BLOWN: A Blockchain Protocol for Single-Hop Wireless Networks under Adversarial SINR

Minghui Xu, *Student Member, IEEE,* Feng Zhao, *Member, IEEE,* Yifei Zou, Chunchi Liu,
Xiuzhen Cheng, *Fellow, IEEE,* Falko Dressler, *Fellow, IEEE*

Abstract—Known as a distributed ledger technology (DLT), blockchain has attracted much attention due to its properties such as decentralization, security, immutability and transparency, and its potential of servicing as an infrastructure for various applications. Blockchain can empower wireless networks with identity management, data integrity, access control, and high-level security. However, previous studies on blockchain-enabled wireless networks mostly focus on proposing architectures or building systems with popular blockchain protocols. Nevertheless, such existing protocols have obvious shortcomings when adopted in wireless networks where nodes have limited physical resources, fall short of well-established reliable channels, variable bandwidths impacted by environments or jamming attacks. In this paper, we propose a novel consensus protocol named Proof-of-Channel (PoC) leveraging the natural properties of wireless communications, and a BLOWN protocol (BLOckchain protocol for Wireless Networks) for wireless networks under an adversarial SINR model. We formalize BLOWN with the universal composition framework and prove its security properties, namely persistence and liveness, as well as its strengths in countering against adversarial jamming, double-spending, and Sybil attacks, which are also demonstrated by extensive simulation studies.

Index Terms—Blockchain; Proof-of-Channel; wireless networks; adversarial SINR; jamming; Sybil attacks.

✦

1 INTRODUCTION

Distributed Ledger Technology (DLT) refers to share, replicate, and synchronize a digital ledger across a distributed network without centralized data storage. As a widely used DLT, blockchain technologies intend to organize a digital ledger as a chain of blocks to enable remarkable properties such as decentralization, immutability, and traceability. Since Bitcoin has emerged as the first open cryptocurrency, blockchain has been envisioned as a promising technology that can be used in various practical applications such as finance [1], Internet of Things (IoT) [2], supply chain [3], and security services [4]. In recent years, the popularity of 5G and IoT has arisen more problems of managing devices, sharing information, and carrying on computing tasks among wireless nodes [5]. Such problems become even intractable in a wireless network with small-world and super-dense features [6]. To overcome these challenges, effort has been made to build secure and trusted computing environments such as mobile edge computing (MEC) enabled blockchain [7] and the blockchain empowered 5G [8] in wireless networks taking advantage of blockchain technologies.

Previous studies on blockchain-enabled wireless networks mostly focus on proposing architectures or building systems on top of popular blockchain protocols that are previously deployed on the Internet. Such blockchain protocols make use of consensus algorithms that are based on either proof of resources or message passing. Proof of resources based consensus requires users to compete for proposing blocks by demonstrating their utilization of physical resources such as energy and storage (e.g. Proof-of-Work [9]) or virtual resources such as reputation and weight (e.g., Proof-of-Stake [10]). Message passing based protocols such as PBFT [11], on the other hand, require the participants to reach consensus through message exchanges. Even though these consensus algorithms perform well for existing blockchain protocols, they are not suitable for wireless networks since they are mainly developed for systems with Internet serving as the underlying network infrastructure. More specifically, wireless networks fall short of well-established reliable channels built with physical wires such as fiber as the Internet does – the open free air communications are severely impacted by environments (e.g., interference or contention) resulting in the variable channel bandwidths; and wireless networks are particularly vulnerable to jamming attacks. These two barriers make it very possible for communications to fail, making the traditional consensus algorithms inapplicable. Such problems are not sufficiently addressed by existing blockchain protocols, which motivates our study on blockchain protocols over wireless.

In this paper, we propose BLOWN, a BLOckchain protocol for Wireless Networks, to overcome the above challenges. BLOWN is a two-phase protocol that adopts a new concept, namely Proof-of-Channel (PoC), to seamlessly inte-

---

M. Xu and X. Cheng are with the School of Computer Science and Technology, Shandong University, Qingdao, 266510, P. R. China. E-mail: {mhxu,xzcheng}@sdu.edu.cn; and with the Department of Computer Science, The George Washington University, Washington, DC 20052 USA. E-mail: {mhxu,cheng}@gwu.edu.
F. Zhao (Correspondign Author) is with the Guangxi Colleges and Universities Key Laboratory of Complex System Optimization and Big Data Processing, Yulin Normal University, Yulin, P.R. China. E-mail: zhaofeng@guet.edu.cn.
Y. Zou is with the School of Computer Science and Technology, Shandong University, Qingdao, 266510, P. R. China. E-mail: {yfzou}@sdu.edu.cn.
C. Liu is with the School of Computer Science and Technology, Shandong University, Qingdao, 266510, P. R. China. E-mail: liuchunchi@sdu.edu.cn.
F. Dressler is with the the School of Electrical Engineering and Computer Science, TU Berlin, 10587 Berlin, Germany. E-mail: dressler@ccs-labs.org.
Manuscript created June 20, 2020

grate the procedures of consensus and channel competition. In PoC, nodes compete for available channels to win the rights of proposing blocks. Such a design makes probing the wireless channel conditions part of the consensus procedure, successfully reducing the communication cost while increasing consensus efficiency and effectiveness. On the other hand, we consider that an adversary can make adversarial jamming on the nodes but controls no more than 50% wealth of the network in BLOWN, where wealth is defined to be the total number of coins held by all users. BLOWN is a provably secure system that satisfies two formal security properties: persistence and liveness. Persistence means that if an honest node proclaims a transaction as stable, other honest nodes, if queried, either report the same result or report error messages. Liveness, on the other hand, states that the transactions originated from the honest nodes can eventually be added to the blockchain. To prove BLOWN's properties, we formally model it with a universally composable (UC) framework and analyze it accordingly. Note that it is worthy of emphasizing that PoC can be adapted to multi-hop wireless networks if combined with existing techniques such as distributed spanner construction [12] or supported by an adequate routing layer [13].

Our main contributions are summarized as follows.

1) To the best of our knowledge, BLOWN is the first provably secure protocol that is specifically designed for wireless networks under a realistic adversarial SINR model.
2) A novel, general Proof-of-Channel consensus protocol is proposed in this paper, which leverages the natural properties of wireless networks such as broadcast communications and channel competitions.
3) We develop a UC-style protocol for BLOWN and formally prove BLOWN's persistence and liveness properties by showing that it satisfies concrete chain growth, common prefix, chain quality properties.
4) Finally, extensive simulation studies are conducted to validate our theoretical analysis.

The rest of the paper is organized as follows. Section 2 introduces the most related works on state-of-the-art blockchain protocols. Section 3 presents our models and assumptions. In Section 4, the two-phase BLOWN protocol is explained in detail. Security properties of BLOWN are analyzed in Section 5, which discusses jamming-resistant communications, double-spending attacks, and Sybil attacks. We report the results of our simulation studies in Section 6 and conclude this paper in Section 7.

2 RELATED WORK

Blockchain consensus protocols. We classify blockchain consensus protocols into two categories: proof of resources (virtual or physical) and message passing, and briefly overview them in this section. For a more comprehensive survey we refer the readers to [14].

Proof of physical resources means that users compete for proposing blocks by demonstrating their utilization of physical resources. Proof-of-Work (PoW) is of the most

use in blockchain. The most popular example of PoW-based blockchain is the Bitcoin proposed in 2008, which selects leaders by computational powers (Proof-of-Work) [9]. Ethereum provides the Turning-complete Ethereum Virtual Machine (EVM) and adopts a modified PoW (with Ethash) [15]. Alternatives to PoW include Proof-of-Space [16], Proof-of-Burn (PoB) [17], Proof-of-Elapsed Time (PoET) [18], in which Proof-of-Space, also known as Proof-of-Capacity or Proof-of-Storage, refers to consensus nodes competing by occupied memories or disk spaces, PoB means that a node can destroy coins to virtually earn mining rights, and PoET, proposed by Intel, leverages trusted hardware (e.g., SGX) to determine how long a node has to wait before it is allowed to generate a block.

In contrast to proof of physical resources, proof of virtual resources aims to show the utilization of virtual resources such as reputation, stake, or elaborately defined weight. For example, Proof of Stake (PoS) was developed to address the power consumption issue of PoW and it resorts to stakes as voting rights rather than computational powers. Algorand uses a cryptographic Sortition algorithm to randomly select a verifiable committee according to stakes [19]. IOHK created the Ouroboros family in recent years, which adopts PoS and G.O.D coin tossing to randomly choose a leader according to stakes [10]. Snow White utilizes an epoch-based committee which embodies successful block miners in a specific time period so that all nodes have an identical view of the committee [20]. In Proof-of-Reputation (PoR), each node is assigned a reputation [21], and a node can write blocks only when its reputation meets certain requirements; thus PoR always comes with incentive mechanisms or economic penalty strategies.

In message passing based blockchain protocols, nodes can perform local computations and broadcast messages to each other to reach consensus. This method provides blockchain the robustness to Byzantine failures while ensuring liveness and safety. In Ripple, a transaction that receives more than 80% votes from UNL-recorded servers can step into the next round, and transactions having survived through the whole RPCA process can be added to the blockchain [22]. ELASTICO partitions nodes by their unique identities and a consensus is reached in each shard based on byzantine agreement protocols [23]. Stellar creates overlapped shards, also known as quorum slices, leveraging Federated Byzantine Agreement (FBA) to reach consensus [24]. Omniledger uses lottery-like RandHound and VRF-based leader election algorithms to assign validators to each shard [25]. Other message-passing based protocols utilized in blockchain include PBFT [11], HoneyBadgerBFT [26], Tendermint [27], and Hotstuff [28].

Blockchain for Internet of Things. IoT encompasses devices that are generally connected to a wireless network. Blockchain has been applied for various IoT applications such as access management, security enhancement, and privacy protection. Novo developed a new architecture, which contains six components, for access management in IoT based on blockchain [29]. Dorri et al. optimized blockchain for IoT by introducing a distributed trust model, in which new blocks proposed by the users with high trust can be free from complete transaction validation to decrease the transaction processing overhead. Feng et al. [30] proposed a radio

and computational resource allocation joint optimization framework for blockchain-enabled mobile edge computing. In vehicular ad hoc networks, Malik *et al.* [31] utilized blockchain to achieve secure key management. Guo *et al.* [32] presented a novel endogenous trusted framework for IoT, which integrates blockchain, software-defined networking, and network function virtualization. Guo *et al.* [33] constructed a blockchain-based authentication system to realize trusted data sharing among heterogeneous IoT platforms. In [34], Liu *et al.* developed a tokoin (token+coin) based novel framework to provide fine-grained and accountable access procedure control leveraging blockchain for various IoT applications. Its unique significance lies in that the fine-grained access policy can be defined, modified, and revoked only by the resource owner and the whole access procedure, not just the access action alone, can be accountably and securely controlled. In [35], Liu *et al.* proposed an important idea of extending trust from on-chain to off-chain making use of trusted hardware and blockchain to ensure that the data in digital world is consistent with the truth in physical world and that any change in physical world should be reflected instantly by the data change in digital world.

Consensus protocols for wireless networks. Since consensus is the core of blockchain and our study is closely related to wireless networks, we briefly survey the studies on consensus protocols for wireless networks. The abstract MAC layer [36] is one of the earliest models that can achieve elegant abstraction and precisely capture the fundamental guarantees of the low-level network layer performance in wireless networks. Newport provided the first tight bound for distributed consensus in radio networks [37]. With the abstract MAC layer, Newport and Robinson gave a fault-tolerant consensus algorithm that terminates within $O(N^3 \log N)$, where N is the unknown network size [38]. A pioneering work on the implementation of the abstract MAC layer provides a groundbreaking scheme to adaptively tune the contention and interference in wireless channels [39]. Moniz *et al.* [40] proposed a BFT consensus protocol allowing $k > \lfloor \frac{N}{2} \rfloor$ faulty nodes with time complexity of $O(N^2)$. They assumed an abstract physical layer in wireless ad hoc networks and directly used high-level broadcast primitives. Chockler *et al.* [41] explored fault-tolerant consensus with crashed nodes. Their study reveals the relationship of collision detection and fault-tolerant consensus under a graph-based model. Assuming realistic message delays and a graph model, Scutari and Sergio designed a distributed consensus algorithm for wireless sensor networks [42], making use of a network model that considers the MAC layer with a multipath and frequency-selective channel. Aysal *et al.* [43] studied the average consensus problem with probabilistic broadcasts. They explored the effect of wireless medium on the consensus process and extended the non-sum preserving algorithm to accelerate convergence.

Summary. A common drawback of proof of physical resources lies in their prohibitively large demands of physical resources such as high computational power, storage, energy, or specific hardware, of which devices in wireless networks are notoriously limited; on the other hand, proof of virtual resources might encounter centralization problems caused by the over-powerful validators or authorities. Additionally, message-passing protocols always incur a significant amount of message exchanges leading to non-trivial communication overhead. Most notably, almost all existing works mentioned above were developed for the Internet resting on the close medium with sufficient bandwidth where jamming is not an issue; thus they are insufficient in addressing problems of limited bandwidths and jamming attacks arisen from the open nature of wireless communications. Motivated by these observations, in this paper we propose BLOWN, a wireless blockchain protocol that relies on the newly-developed PoC to seamlessly integrate wireless communications with blockchain consensus while guaranteeing persistence and liveness, the two critical security properties of blockchain, to counter jamming and various blockchain-specific attacks.

3 MODELS AND ASSUMPTIONS

Network Model. In this paper, we consider a network with a set V of N nodes arbitrarily deployed in a communication space. Such a network could contain a group of manipulated Unmanned Aerial Vehicles (UAVs) or intelligent robots in realistic scenarios. A node is equipped with a half-duplex transceiver that can transmit or receive a message, or sense the channel, but cannot transmit and receive or transmit and sense simultaneously. Let $d(u, v)$ be the Euclidean distance between nodes u and v, $D_R(v)$ denote the disk centered at v with a radius R, and $N_R(v)$ denote the set of nodes within $D_R(v)$ including v.

We assume that each node knows the identities, locations and public keys of all other nodes. We further assume that each node can generate key pairs and has access to a secure EUF-CMA digital signature scheme (details of cryptographical tools employed in this paper are presented in protocol analysis and simulation sections). Each node maintains a hash-chain of blocks, and each block contains multiple transactions. We denote frequently-used notations of transaction, block, blokchain, chain of block headers by tx, B, BC, and BH, respectively, and use super/subscript to attach more specific information. A transaction is modeled as a coin exchange process. We adopt the notion of the unspent transaction outputs (UTXOs) accounting method. A UTXO-based account stores coins in a set of UTXOs, and a UTXO defines an output of a blockchain transaction that has not been spent. This UTXO model provides a high level of security since it is convenient to authenticate all transaction sequences using UTXOs, limiting the risk of double-spending attacks.
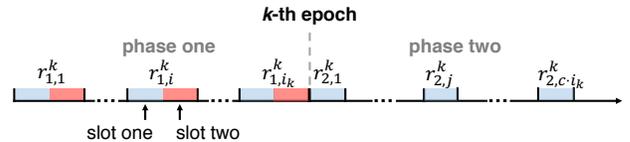

Fig. 1. Epoch-based execution.

Interference Model. We adopt the Signal-to-Interference-plus-Noise-Ratio (SINR) wireless network model, which captures the network interference in a more realistic and precise manner than a graph-based one [44]. A

standard SINR model can be formulated as follows, which states that a message sent by u is correctly received by v if and only if

$$SINR = \frac{\mathcal{S}}{\mathcal{I} + \mathcal{N}} \geq \beta \qquad (1)$$

where $\mathcal{S} = P \cdot d(u,v)^{-\alpha}$ is the received signal power of node v from u, P is the uniform transmit power, $\mathcal{I} = \sum_{w \in W \setminus \{u\}} P \cdot d(w,v)^{-\alpha}$ is the interference at v and W is the set of nodes that transmit in the current round, $\mathcal{N}$ is the ambient noise, $\alpha \in (2,6]$ is the path-loss exponent, and threshold $\beta > 1$ is determined by hardware. To capture a fine-grained noise, we define $\mathcal{N} = \mathcal{ADV}(v)$, where $\mathcal{ADV}(v)$ is the composite noise generated by the environment and adversaries. Assume each node uses a common noise threshold θ, and the distance between any two nodes is bounded by $R_0 = (P/\beta\theta)^{1/\alpha}$. We further assume that each node can perform physical carrier sensing. If at least one neighboring node u of v broadcasts a message, v would either receive a message or sense a busy channel. At each slot, a node v may either (a) sense an idle channel (the measured signal power is lower than θ); (b) receive a message (the measured signal power exceeds θ and $SINR \geq \beta$); or (c) sense a busy channel (the measured signal power exceeds θ but $SINR < \beta$). Let $RSS = \mathcal{S} + \mathcal{I} + \mathcal{N}$ be the total received signal power at a node. Then when the node receives a message, the interference plus noise can be calculated by $\mathcal{I} + \mathcal{N} = RSS - \mathcal{S}$ with a known $\mathcal{S}$ [45]–[49].

Epoch-based Execution. As shown in Fig. 1, the BLOWN protocol is executed in disjoint and consecutive time intervals called *epochs*, and at each epoch no more than one block can be generated. Each epoch e_k consists of two phases with each containing multiple rounds. In e_k, we denote $r_{1,i}^k$ as the i-th round in phase one and $r_{2,j}^k$ as the j-th round in phase two, with $r_{1,i}^k$ consisting of two slots and $r_{2,j}^k$ having only one slot. Besides, i_k is the length of phase one and $c \cdot i_k$ is the length of phase two, where c is a variable constant determined later.

Adversary. We assume that there exists a group of adversaries who can freely join or leave the network, create identities, or make noises to interfere with any honest node at any time. For simplicity, the group of adversaries can be regarded as a powerful adversary $\mathcal{A}$ who controls less than 50% wealth of the entire network. $\mathcal{A}$ can launch jamming attacks. To leave a chance for an honest node to communicate, $\mathcal{A}$ is $((1-\epsilon), T)$-bounded at any time interval of length T rounds, where $T \in \mathbb{N}$ and $0 < \epsilon \leq 1$, indicating that the super adversary can jam nonuniformly at most $(1 - \epsilon)T$ rounds within T. Each node v maintains a variable T_v, which is the estimate of T by v.

In this paper, we say that event E occurs with high probability (w.h.p.) if for any $c \geq 1$, E occurs with probability at least $1 - 1/N^c$, and with moderate probability (w.m.p.) if for any $c \geq 1$, E occurs with probability at least $1 - 1/\log^c N$. A summary of all important notations (including the ones from the BLOWN protocol and the protocol analysis) and their semantic meanings is provided in Table 1.

4 THE BLOWN PROTOCOL

In this section, we present the two-phase BLOWN protocol. We first summarize the BLOWN protocol by providing an overview of BLOWN and its construction primitives, and then detail the protocol itself.

TABLE 1
Summary of Notations

Symbol	Description
B_v^k	block generated by node v in the k-th epoch
BC_v^k	blockchain locally stored at node v
BH_v^k	block headers read from BC_v^k
$P_1(P_2)$	phase one (phase two)
V	set of all nodes
N	network size
c	constant used to determine the length of phase two
c_v	counter variable used to record round information
i_k	length of P_1 of the k-th epoch
tx	a transaction
txp_v	temporary transaction stack of node v
p_v	v's probability of sending a message
p_V	aggregated probability of all nodes
$\hat{p}$	an upper bound of p_v
$r_{1,i}^k(r_{2,j}^k)$	i-th (j-th) round of $P_1(P_2)$ in the k-th epoch
l_v	$l_v \in \{0, 1, \cdots, w_v\}$ as the leader counter of v
l_v^0	initial value of l_v generated by *Sortition*()
T	time window of the adversary
T_v	estimate of T by node v
w_v	deposit of node v
π_v	proof created by *Sortition*()
ϵ	proportion of non-jammed rounds
τ	the parameter to determine the hardness of Sortition

4.1 Overview and Utilities of BLOWN

In this subsection, we present an overview on BLOWN, and describe the construction primitives/utilities to more precisely and concisely illustrate the BLOWN protocol.
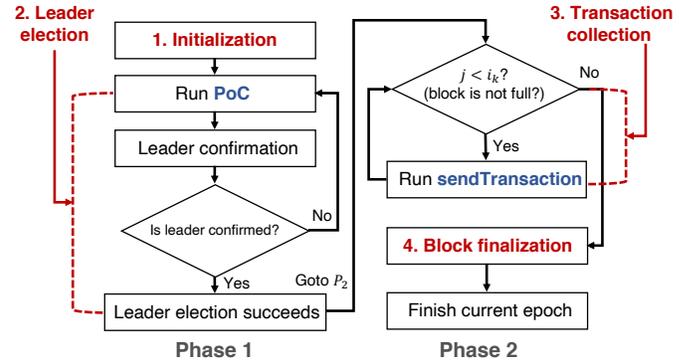
4.1.1 An Overview on BLOWN

Fig. 2. BLOWN protocol overview.

The BLOWN protocol proceeds in epochs, with each constructing no more than one block. Specifically, our protocol has two phases within an epoch, denoted by P_1 and P_2 as shown in Fig. 2. P_1 is responsible for initialization and leader election while P_2 is for transaction collection and block finalization. In our design, nodes contend by broadcasting messages on a wireless channel. In response, we establish a robust jamming-resistant channel by introducing an adaptive transmission mechanism, confronting background noise and jamming simultaneously. Such a channel

Algorithm 1 Utilities for node v

1: **function** $Sortition(sk_v, \text{seed}, \text{role}, \tau, w_v, W)$
2:    $h_v, \pi_v = \text{VRF}(sk_v, \text{seed}||\text{role})$
3:    $p = \tau/W$, $l_v = LeaderCounter(\text{role}, w_v, h_v, p)$
4:    **return** h_v, π_v, l_v
5: **function** $VerifySortition(pk_v, h_v, \pi_v, \text{seed}, \text{role}, \tau, w_v, W, l_v)$
6:    **if** $\text{VerifyVRF}(pk_v, h_v, \pi_v, \text{seed}||\text{role}) = 0$ **then**
7:       **return** 0
8:    $p = \tau/W$, $\hat{l_v} = LeaderCounter(\text{role}, w_v, h_v, p)$
9:    **if** $\hat{l_v} \neq l_v$ **then**
10:      **return** 0
11:    **return** 1
12: **function** $LeaderCounter(\text{role}, w_v, h_v, p)$
13:    $l_v = 0$
14:    **if** role is FOLLOWER **then**
15:      **return** l_v
16:    **if** $\frac{h_v}{2^l} \in \left[0, \sum_{k=0}^{l_v} B(k; w_v, p)\right]$ **then**
17:      **return** l_v
18:    $l_v = 1$
19:    **while** $\frac{h_v}{2^l} \notin \left(\sum_{k=0}^{l_v-1} B(k; w_v, p), \sum_{k=0}^{l_v} B(k; w_v, p)\right]$ **do**
20:      $l_v = l_v + 1$
21:    **return** l_v
22: **function** $MSG(r_{1,i}^k, l_v)$
23:    $m.r_{1,i}^k \leftarrow r_{1,i}^k$
24:    $m.l_v \leftarrow l_v$
25:    **return** m
26: **function** $MSGT(tx, r_{2,j}^k, l_v)$
27:    $m_T.tx \leftarrow tx$
28:    $m_T.r_{2,j}^k \leftarrow r_{2,j}^k$
29:    $m_T.l_v \leftarrow l_v$
30:    **return** m_T
31: **function** $MSGB(BC_v^k, B_v^k, r_{2,j}^k, l_v, \text{role}, w_v, h_v, \pi_v, l_v')$
32:    $m_B.BH_v^k \leftarrow$ read BH_v^k from BC_v^k
33:    $m_B.B_v^k \leftarrow B_v^k$
34:    $m_B.r_{2,j}^k \leftarrow r_{2,j}^k$
35:    $m_B.l_v \leftarrow l_v$
36:    $m_B.\text{sort} \leftarrow \{\text{role}, w_v, h_v, \pi_v, l_v'\}$
37:    **return** m_B
38: **function** $Pack(txp_v)$ ▷ Pack txp to form a block
39:    **return** B_v^k
40: **function** $Append(BC_v^{k-1}, B_u^k)$ ▷ append the new block
41:    **return** $BC_v^k \leftarrow BC_v^{k-1} + B_u^k$

---

ceiving a message, l_v is decremented. The sole survivor with non-zero l_v at the end of P_1 is appointed as the leader. This essentially integrates leader election and channel contention into a single process, namely the phase one of BLOWN. In P_2, the leader is responsible for collecting and verifying transactions, assembling them into a new block, and then broadcasting the block to the whole network. If the new block is valid, it is admitted by all honest nodes.

4.1.2 Utilities

Algorithm 1 lists the following frequently used functions for any node v in BLOWN: *Sortition*(), *VerifySortition*(), *LeaderCounter*(), *MSG*(), *MSGT*(), *MSGB*(), *Pack*(), and *Append*(). It also presents the following data structures employed by the above functions: transaction tx, transaction stack txp_v, block B_v^k, block header BH_v^k, blockchain BC_v^k, basic message m, transaction message m_T, and block message m_B.

Concretely, we modify the crytographic Sortition algorithm proposed by Algorand [19] to make it suitable for our BLOWN protocol. The Sortition algorithm is based on a verifiable random function (VRF), which takes as inputs a private key sk_v, a random seed and a role, and outputs a hash h_v as well as its corresponding proof π_v. There are two types of roles: a FOLLOWER who can only be a follower during an epoch and a LEADER who is a potential leader. Besides, W is the accumulated number of coins of all users in the network, w_v is the deposit of node v, $l_v \in \{0, 1, \cdots, w_v\}$ is the leader counter of the node v, and $p = \frac{\tau}{W}$ is the probability based on which each coin is used to increment the counter value where τ determines the hardness. The probability of $l_v = k$ follows the binomial distribution $B(k; w_v, p) = \binom{w_v}{k} p^k (1-p)^{w_v-k}$ with $\sum_0^{w_v} B(k; w_v, p) = 1$. To determine l_v, the *LeaderCounter*(role, w_v, h_v, p) divides $[0, 1]$ into consecutive intervals as $I(l_v) = \left[0, \sum_{k=0}^{l_v} B(k; w_v, p)\right]$ for $l_v = 0$ and $I(l_v) = \left(\sum_{k=0}^{l_v-1} B(k; w_v, p), \sum_{k=0}^{l_v} B(k; w_v, p)\right]$ for $I(l_v) \in \{1, \cdots, w_v\}$. If v's role is FOLLOWER, $I(l_v) = 0$; otherwise, if the normalized hash $\frac{h_v}{2^l}$ (l is the hash length) falls in the interval $I(l_v)$, l_v is returned as the value of the leader counter. The function *VerifySortition*() intends to check if h_v, π_v, l_v are valid by calling *VerifyVRF*() and recomputing *LeaderCounter*().

Three functions, namely *MSG*(), *MSGT*(), and *MSGB*(), generate messages that can respectively be used for leader election, transaction collection, and block finalization. Specifically, *MSG*() creates a basic message m for leader election in P_1, *MSGT*() produces a message m_T embodying a transaction, which is sent during the transaction collection process in P_2, and *MSGB*() outputs a message m_B which contains a B_v^k generated by the leader v, a BH_v^k read from BC_v^k, the current value l_v of the leader counter, and a string $\{\text{role}, w_v, h_v, \pi_v, l_v'\}$ used to verify Sortition where l_v' is the original value of the leader counter. To reduce communication cost, we send BH_v^k embodied in m_B for a simplified verification. Finally, *Pack*(txp_v) is adopted to validate and pack transactions to form a new block, and *Append*(BC_v^{k-1}, B_u^k) appends the new block B_u^k to the local blockchain BC_v^{k-1}.

is realized by dynamically adjusting the transmission probability p_v of each node v according to its sensed contention in the network. Concretely, we first adopt the Sortition algorithm to assign v a weight l_v based on its account balance. Sortition ensures that splitting coins to generate massive identities cannot break our protocol. After initialization, the protocol starts the process of leader election. We utilize the nature of contention in a wireless network to design our proof-of-channel consensus mechanism (PoC). To achieve usability and efficiency, PoC allows nodes to compete on the channel right-of-use to obtain opportunities of proposing blocks rather than rely on extra physical resources or introduce communication overhead. More specifically, upon re-

4.2 The BLOWN Protocol Specifications

In a nutshell, BLOWN is a two-phase protocol. As shown respectively in Algorithm 3 and Algorihtm 5, phase P_1 is employed for initialization and leader election while phase P_2 is for transaction collection and block finalization.

4.2.1 Phase P_1

Algorithm 2 PoC subroutine

1: **if** v decides to send a message based on p_v **then**
2:    $m \leftarrow MSG(r_{1,i}^k, l_v)$, v broadcasts (m, σ)
3: **else**
4:    **if** channel is idle **then**
5:       $p_v = \min\{(1+\gamma)p_v, \hat{p}\}$
6:       $T_v = \max\{1, T_v - 1\}$
7:    **else**
8:       **if** v receives a message (m, σ) **then**
9:          $p_v = (1+\gamma)^{-1}p_v$
10:          $l_v = l_v - 1$
11: $c_v = c_v + 1$
12: **if** $c_v \geq T_v$ **then**
13:    $c_v = 1$
14:    **if** there is no idle rounds in the past T_v rounds **then**
15:       $p_v = (1+\gamma)^{-1}p_v,$
16:       $T_v = T_v + 2$

Algorithm 3 BLOWN P_1 protocol

1: ▷ Initialization
2: $h_v, \pi_v, l_v^0 = Sortition(sk_v, \text{seed} || \text{role}, \tau, w_v, W)$
3: $p_v = \hat{p}, c_v = 0, T_v = 1, i = 1, l_v = l_v^0$
4: ▷ Leader election
5: **while** TRUE **do**
6:    **if** $l_v > 0$ **then**   ▷ As a potential leader
7:       ▷ slot one of $r_{1,i}^k$
8:       run **PoC**
9:       ▷ slot two of $r_{1,i}^k$
10:       **if** v broadcasts a message in slot one **then**
11:          v listens on the channel
12:          **if** v senses an idle channel **then**
13:             Goto P_2 with $i_k = i$ ▷ run P_2 as a leader
14:       **else**
15:          $m \leftarrow MSG(r_{1,i}^k, l_v)$, and v broadcasts (m, σ)
16:    **else**   ▷ As a follower
17:       ▷ slot one of $r_{1,i}^k$
18:       v listens on the channel to receive a message
19:       ▷ slot two of $r_{1,i}^k$
20:       **if** in slot one v receives (m, σ) from u and has $\mathcal{I} + \mathcal{N} < \theta$ **then**
21:          **if** v senses an idle channel **then**
22:             v recognize u as the leader
23:             Goto P_2 with $i_k = i$ ▷ run P_2 as a follower
24:       **else**
25:          $m \leftarrow MSG(r_{1,i}^k, l_v)$, and v broadcasts (m, σ)
26:    $i = i + 1$

Let's examine the details of the BLOWN P_1 protocol. Lines 2-3 of Algorithm 3 constitute the initialization process. First, $Sortition()$ takes as inputs sk_v, seed$||$role, τ, w_v and W

(see Line 2), and outputs h_v, π_v and l_v^0, where h_v and π_v are respectively a hash and its corresponding proof, and $l_v^0 \in \{0, 1, \cdots, w_v\}$ stands for the initial leader counter. Note that $l_v > 0$ indicates that v remains to be a potential leader while $l_v = 0$ indicates that v is a follower. Let $\hat{p}$ be the maximum transmission probability, which can be initialized to any small number in $(0, 1)$. Since the absence of followers might lead to a bad case in which all nodes are potential leaders and simultaneously broadcast messages in slot one[1], we prevent this from occurring by ensuring that there always exist at least one follower after initialization. A simple approach to achieving this goal is to artificially and randomly add followers (with a FOLLOWER rule) to the network. Second, we set $p_v = \hat{p}$, $c_v = 0$, $T_v = 1$, $i = 1, l_v = l_v^0$ (Line 3), where p_v is the probability on which node v decides to send a message, and is upper-bounded by $\hat{p}$, $c_v = 1$ is a counter variable used to record round information, T_v is the estimate of the time window of the adversary by node v, i is the round counter used in P_1, and l_v is the leader counter variable initialized to l_v^0. After initialization, P_1 proceeds round by round with each containing two slots, and a node v's activity at each slot depends on its role.

Before proceed any further, we need to explain the PoC subroutine described in Algorithm 2 to adjust l_v, p_v and T_v, the leader counter, transmission probability and adversary's time window estimate, according to the sensed channel condition at the first slot of each round in P_1. Specifically, v with $l_v > 0$ (a potential leader) performs the following actions: it either broadcasts a message (m, σ) with probability p_v (Lines 1-2), where σ is the signature of m, or senses the channel with probability $1 - p_v$ (Lines 3-10). One can see that v adapts its p_v in a multiplicative increase or decrease manner by a factor of $(1 + \gamma)$, where $\gamma = O(1/(\log T + \log \log N))$ is a small number that is loosely determined by N and T (see the proof of Theorem 3). Particularly, p_v is multiplicatively increased (Line 5) when the channel is sensed idle or decreased (Line 9) when a message is received[2]. Such a mechanism ensures that honest nodes can cooperatively adjust their transmission probabilities to help reduce contention on the channel. Meanwhile, we decrease T_v by 1 if the channel is idle (Line 6) as the estimate of adversary's time window seems to be too large when the channel is idle, and decrease l_v by 1 if a neighbor of v successfully broadcasts a message (Line 10) as the neighbor seems to have a better chance of being the leader. On the other hand, if the number of rounds in P_1 is no less than T_v (Line 12), the estimate of the adversary's time window, we further check whether or not there is an idle round in the past T_v rounds (Line 14), and if not, p_v is decreased (Line 15) and T_v is increased (Line 16) to further adjust p_v and T_v. One can see that a successful broadcast causes the decrements of the l_v values of the receivers. When $l_v = 0$, v becomes a follower who can only sense the channel in slot one of the next round.

Now we are back to continue explaining Phase P_1 of the

---

1. Such a bad case only occurs with a small probability, which is less than $\hat{p}^n$

2. Receiving a message indicates the message has a valid signature, and we do not explicitly present the signature verification process for conciseness.

BLOWN protocol, which contains multiple rounds. At slot one of each round, if v is a potential leader, which means $l_v > 0$, v runs the PoC subroutine described in Algorithm 2 (Line 8); otherwise, v listens on the channel for message reception (Line 18). At slot two of each round, v behaves according to its actions in slot one. If v as a potential leader broadcasts a message in slot one and senses an idle channel in slot two, it can set itself as a leader and goto P_2 (Lines 10-13); otherwise it broadcasts a message in slot two. A follower v recognizes u as the leader only when v believes u is the only transmitter in slot one and senses an idle channel in slot two (Lines 20-23); otherwise v transmits in slot two (Line 25). In Theorem 4, we prove that slot two is capable of letting the leader and the followers mutually recognize each other.

At the end of P_1, there should be only one survivor with $l_v > 0$, who then becomes the leader. Note that i_k denotes the length of P_1, which is used to determine the length of P_2. We will prove in Theorem 4 of Section 5 that Algorithm 3 ensures a successful leader election.

Algorithm 4 sendTransaction subroutine

1: **if** v decides to send a message based on p_v **then**
2: $\quad m_T \leftarrow MSGT(tx, r_{2,j}^k, l_v)$, and v broadcasts (m_T, σ_T)
3: **else**
4: $\quad$ **if** channel is idle **then**
5: $\quad\quad p_v = \min\{(1+\gamma)p_v, \hat{p}\}$
6: $\quad\quad T_v = \max\{1, T_v - 1\}$
7: $\quad$ **else**
8: $\quad\quad$ **if** receives a message (m_T, σ_T) **then**
9: $\quad\quad\quad p_v = (1+\gamma)^{-1}p_v$
10: $c_v = c_v + 1$
11: **if** $c_v \geq T_v$ **then**
12: $\quad c_v = 1$
13: $\quad$ **if** there is no idle round in the past T_v rounds **then**
14: $\quad\quad p_v = (1+\gamma)^{-1}p_v$,
15: $\quad\quad T_v = T_v + 2$

4.2.2 Phase P_2

Phase P_2 of BLOWN performs transaction collection and block finalization, as shown in Algorithm 5. It proceeds by a fixed amount of $c \cdot i_k$ rounds where each round contains only one slot, and c is a constant to directly determine the length of P_2 and indirectly the maximum block size, which can be adjusted according to specific implementations. We refer to j as the round counter in P_2. If $j < c \cdot i_k$, a leader selected in P_1 should listen to the channel to receive signed transaction messages, which are recorded in the stack txp_v, while other nodes continuously broadcast signed transaction messages (Lines 2-8). After $c \cdot i_k$ rounds, the leader serializes all transactions to form a new block denoted by $B_v^k \leftarrow Pack(txp_v)$, and broadcasts the (m_B, σ_B) (Lines 12-15). Once receiving a (m_B, σ_B) from u, a node v should append the new block to its local blockchain only if σ_B is valid and *VerifySortition* $(pk_v, seed, \tau, W, m_B.sort)$ = 1 (Lines 17-18). Note that the sendTransaction subroutine presented in Algorithm 4 is employed by P_2 to broadcast transactions and the parameters p_v, c_v, T_v are utilized to ensure jamming-resistant communications as they function in the PoC subroutine shown in Algorithm 2.

Algorithm 5 BLOWN P_2 protocol

1: $\triangleright$ Transaction collection
2: **while** $j < c \cdot i_k$ **do**
3: $\quad$ **if** $l_v > 0$ **then** $\qquad\qquad\qquad\quad \triangleright$ As a leader
4: $\quad\quad v$ listens on the channel to receive a (m_T, σ_T)
5: $\quad\quad$ **if** receives $m_T.\text{tx} \neq \perp$ **then**
6: $\quad\quad\quad txp_v[j] = m_T.tx$
7: $\quad$ **else** $\qquad\qquad\qquad\qquad\qquad\quad \triangleright$ As a follower
8: $\quad\quad$ run **sendTransaction**
9: $\quad j = j + 1$
10: $\triangleright$ Block finalization
11: **if** $j = c \cdot i_k$ **then**
12: $\quad$ **if** $l_v > 0$ **then** $\qquad\qquad\qquad\quad \triangleright$ As a leader
13: $\quad\quad B_v^k \leftarrow Pack(txp_v)$
14: $\quad\quad BC_v^k \leftarrow Append(BC_v^{k-1}, B_v^k)$
15: $\quad\quad m_B \leftarrow MSGB(BC_v^k, B_v^k, r_{2,j}^k, l_v, \text{role}, w_v, h_v, \pi_v, l_v^0)$, and broadcasts (m_B, σ_B)
16: $\quad$ **else** $\qquad\qquad\qquad\qquad\qquad\quad \triangleright$ As a follower
17: $\quad\quad$ **if** receives (m_B, σ_B) && *VerifySortition* $(pk_v, seed, \tau, W, m_B.sort) = 1$ **then**
18: $\quad\quad\quad Append(BC_v^{k-1}, m_B.B_u^k)$

---

5 PROTOCOL ANALYSIS

Proving security properties of a complex protocol such as BLOWN is very challenging. Thus we leverage the universally composable (UC) framework proposed by Canetti *et al.* [50]. The UC framework captures the security of a protocol via emulating an idealized protocol $\mathcal{F}$ (often referred to as an ideal functionality), which satisfies strong security properties. Then a real protocol π specifying concrete implementations is said to be secure if it is indistinguishable from $\mathcal{F}$. The main feature of the UC framework is the universal composability that allows one to perform analysis on a complex protocol, whose security properties can be derived from the security of its components.

5.1 UC Composition of BLOWN

We formulate two UC-style protocols (or hybrid experiments), which are presented in Fig. 3 and Fig. 4. The $\pi_\text{B}[\mathcal{F}_\text{SIG}, \mathcal{F}_\text{SORT}]$ conducts a hybrid experiment for BLOWN using an ideal hybrid functionality $[\mathcal{F}_\text{SIG}, \mathcal{F}_\text{SORT}]$ where $\mathcal{F}_\text{SIG}$ is an ideal digital signature scheme and $\mathcal{F}_\text{SORT}$ is an ideal functionality, performing three sortition-related functions as shown in Fig. 5. BLOWN is denoted as $\pi_\text{B}[\pi_\text{SIG}, \pi_\text{SORT}]$, which implements real protocols π_SIG and π_SORT. Besides, $\pi_\text{SORT}[\mathcal{F}_\text{VRF}]$ is a protocol that realizes sortition-related functionalities, consisting of Leader-Counter, Sortition, and VerifySortition. These functionalities are consistent with the corresponding ones specified in Algorithm 1 except that $\pi_\text{SORT}[\mathcal{F}_\text{VRF}]$ uses an ideal functionality $\mathcal{F}_\text{VRF}$ in Sortition and VerifySortition. In contrast, Algorithm 1 adopts a realistic VRF implementation. Let $\mathcal{A}, \mathcal{Z}, \mathcal{S}$ be respectively the adversary, environment, simulator, whose specific meanings should depend on the context. We first show that the following lemma 1 holds for $\pi_\text{SORT}[\mathcal{F}_\text{VRF}]$.

Lemma 1. *With the same security parameter λ, for each probabilistic polynomial-time (PPT) $\{\mathcal{A}, \mathcal{Z}\}$, it holds that the protocol*

Protocol $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$

π_{B} is a protocol run by all nodes interacting with the ideal functionalities $\mathcal{F}_{\mathrm{SIG}}$ and $\mathcal{F}_{\mathrm{SORT}}$.

Initialization: Send (sk_v, $seed$, $role$, τ, w_v, W) to $\mathcal{F}_{\mathrm{SORT}}$, which returns ($h_v, \pi_v, l_v^0$). Next, initialize the remaining local parameters as $p_v = \hat{p}$, $c_v = 0$, $T_v = 1$, $i = 1$, $l_v = l_v^0$.

Leader election: For each round $r_{1,i}^k$ of P_1 during the k-th epoch, perform the following (1) or (2) according to the value of l_v:

(1) If $l_v > 0$, run PoC in slot one. If broadcasting a message in slot one, listen on the channel in slot two and if the channel is idle, goto P_2 with $i_k = i$ at the end of slot two; otherwise, send $m \leftarrow MSG(r_{1,i}^k, l_v)$ to $\mathcal{F}_{\mathrm{SIG}}$, which returns a signed message (m, σ), i.e., (m, σ) is obtained by querying $\mathcal{F}_{\mathrm{SIG}}$, then broadcast (m, σ) in slot two.

(2) If $l_v = 0$, listen on the channel in slot one. If receiving a valid (m, σ) from u with $\mathcal{I} + \mathcal{N} < \theta$ in slot one, and sensing an idle channel in slot two, recognize u as the leader and goto P_2 with $i_k = i$ at the end of slot two; otherwise, generate $m \leftarrow MSG(r_{1,i}^k, l_v)$, send m to $\mathcal{F}_{\mathrm{SIG}}$, which returns (m, σ), then broadcast (m, σ) in slot two. Note that a valid m holds when $\mathcal{F}_{\mathrm{SIG}}$ returns 1 upon being queried with (m, σ).

Transaction collection: At each round $r_{2,j}^k$, if $l_v > 0$, listen on the channel for a possible signed transaction message m_T, add the transaction to the local stack as $txp_v[j] = m_T.tx$ if receiving (m_T, σ_T) and $\mathcal{F}_{\mathrm{SIG}}$ returns 1 when being queried with (m_T, σ_T). If $l_v = 0$, send $m_T \leftarrow MSGT(tx, r_{2,j}^k, l_v)$ to $\mathcal{F}_{\mathrm{SIG}}$, which returns a signed message (m_T, σ_T), then broadcast (m_T, σ_T).

Block finalization: During the round $r_{2,c \cdot i_k}^k$, if $l_v > 0$, execute $B_v^k \leftarrow Pack(txp_v)$ and $BC_v^k \leftarrow Append(\mathrm{BC}_v^{k-1}, B_v^k)$; then generate $m_B \leftarrow MSGB(BC_v^k, B_v^k, r_{2,j}^k, l_v, role, w_v, h_v, \pi_v, l_v^0)$ and send it to $\mathcal{F}_{\mathrm{SIG}}$, which returns (m_B, σ_B). If $l_v = 0$, listen on the channel for a possible block message (m_B, σ_B); if receiving a valid (m_B, σ_B), which means $\mathcal{F}_{\mathrm{SIG}}$ returns 1 upon being queried with (m_B, σ_B), and $\mathcal{F}_{\mathrm{SORT}}$ returning 1 upon being queried with $(pk_v, seed, \tau, W, m_B.sort)$, execute $Append(\mathrm{BC}_v^{k-1}, m_B.B_u^k)$.

Fig. 3. The protocol (hybrid experiment) $\pi_{\mathrm{B}}[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$.

Protocol $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$

LeaderCounter: When activated with input (role, w_v, h_v, p), first initialize $l_v = 0$. If role is FOLLOWER, output $l_v = 0$ and exit. If role is LEADER, compute $\frac{h_v}{2^l}$; if $\frac{h_v}{2^l}$ falls in $\left[0, \sum_{k=0}^{l_v} B(k; w_v, p)\right]$, output $l_v = 0$ and exit; otherwise, increase l_v until it satisfies that $\frac{h_v}{2^l} \in \left(\sum_{k=0}^{l_v-1} B(k; w_v, p), \sum_{k=0}^{l_v} B(k; w_v, p)\right]$, then send l_v to v and output l_v.

Sortition: When activated with input (sk_v, $seed$, $role$, τ, w_v, W), first feed (sk_v, $seed\|role$) to $\mathcal{F}_{\mathrm{VRF}}$, which returns ($h_v, \pi_v$); then compute $p = \tau/W$ and input (role, w_v, h_v, p) to LeaderCounter, which returns l_v; finally, output (h_v, π_v, l_v).

VerifySortition: When activated with input (pk_v, h_v, π_v, $seed$, $role$, τ, w_v, W, l_v), first feed (pk_v, h_v, π_v, $seed\|role$) to $\mathcal{F}_{\mathrm{VRF}}$, which returns ($pk_v$, h_v, π_v, $seed\|role$, f). If $f = 0$, output FALSE, which means that verification fails; if $f = 1$, compute $p = \tau/W$ and feed (h_v, π_v) to LeaderCounter to obtain $\hat{l}_v$. Following that, if $\hat{l}_v \neq l_v$, output FALSE; otherwise output TRUE, which means that verification succeeds.

Fig. 4. The protocol (hybrid experiment) $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$.

$\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ *securely realizes* $\mathcal{F}_{\mathrm{SORT}}$ *under the* $\mathcal{F}_{\mathrm{VRF}}$*-hybrid model.*

Proof. Let $\mathcal{A}$ be an adversary that interacts with the nodes running $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ under the $\mathcal{F}_{\mathrm{VRF}}$-hybrid model. We need to construct an ideal simulator $\mathcal{S}$ such that the view of any environment $\mathcal{Z}$ of an interaction with $\mathcal{A}$ and $\pi_{\mathrm{SORT}}[\mathcal{F}_{\mathrm{VRF}}]$ is exactly the same as that of an interaction with $\mathcal{S}$ and $\mathcal{F}_{\mathrm{SORT}}$. In our construction, the simulator $\mathcal{S}$ runs $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$ (under the name of $\mathcal{F}_{\mathrm{VRF}}$) and simulates other possibly involved nodes. Here, the $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$ who is attacking the VRF function is identically defined as the one attacking the ideal functionality $\mathcal{F}_{\mathrm{VRF}}^{Praos}$ presented in [51]. $\mathcal{S}$ is responsible for forwarding messages from $\mathcal{Z}$ and $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$. Besides,

$\mathcal{S}$ performs the following operations:

1) Simulating value and proof generation: When $\mathcal{S}$ receives a message (sk_v, $seed\|role$) in the ideal process from $\mathcal{F}_{\mathrm{SORT}}$, it simulates for $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$ (under the name of $\mathcal{F}_{\mathrm{VRF}}$) the process of generating (Evaluated, s_{id}, h_v, π_v), where s_{id} represents a session id which is not explicitly presented in this paper for simplicity. $\mathcal{S}$ then forwards (h_v, π_v) to $\mathcal{F}_{\mathrm{SORT}}$.

2) Simulating verification: When $\mathcal{S}$ receives a message (pk_v, h_v, π_v, $seed\|role$) in the ideal process from $\mathcal{F}_{\mathrm{SORT}}$ meaning a verificaiton query is received, it simulates for $\mathcal{A}_{\mathcal{F}_{\mathrm{VRF}}}$ the process of VRF verification. Once receiving (Verified, s_{id}, h_v, π_v, f), $\mathcal{S}$ forwards (pk_v, h_v, π_v, $seed\|role$, f) to $\mathcal{F}_{\mathrm{SORT}}$.

Functionality $\mathcal{F}_{\text{SORT}}$

LeaderCounter: Upon receiving (role, w_v, h_v, p) from some node v, verify if role is `FOLLOWER`. If so, send $l_v = 0$ to v; otherwise, compute $\frac{h_v}{2^l}$. Next if $\frac{h_v}{2^l}$ falls in $\left[0, \sum_{k=0}^{l_v} B(k; w_v, p)\right]$, sends $l_v = 0$ to v; otherwise increase l_v until it satisfies that $\frac{h_v}{2^l} \in \left(\sum_{k=0}^{l_v-1} B(k; w_v, p), \sum_{k=0}^{l_v} B(k; w_v, p)\right]$, then send l_v to v.

Sortition: Upon receiving (sk_v, seed, role, τ, w_v, W) from some node v, send (sk_v, seed$\|$role) to the adversary, who returns (h_v, π_v).

    (1)   If there is no entry (sk_v, seed$\|$role, h_v, π_v) recorded, record (sk_v, seed$\|$role, h_v, π_v); if there is an existing entry (sk_v, seed$\|$role, h'_v, π'_v) that satisfies $h'_v = h_v$ and $\pi'_v = \pi_v$, do nothing. Next compute $p = \tau/W$ and send (role, w_v, h_v, p) to LeaderCounter, which returns l_v. Finally, send (h_v, π_v, l_v) to v.

    (2)   If there is an entry (sk_v, seed$\|$role, h'_v, π'_v) recorded but $h'_v \neq h_v$ or $\pi'_v \neq \pi_v$, send an error message to v.

VerifySortition: Upon receiving (pk_v, h_v, π_v, seed, role, τ, w_v, W, l_v), send (pk_v, h_v, π_v, seed$\|$role) to the adversary, who returns (pk_v, h_v, π_v, seed$\|$role, f).

    (1)   If $f = 0$ or there is no entry (sk_v, seed$\|$role, h_v, π_v) recorded, send 0 to v, which means that verification fails.

    (2)   If $f = 1$ and there is an existing entry (sk_v, seed$\|$role, h_v, π_v), compute $p = \tau/W$ and send (h_v, π_v) to LeaderCounter, which returns $\hat{l_v}$. If $\hat{l_v} \neq l_v$, sends 0 to v, i.e., verification fails; otherwise send 1 to v meaning that verification succeeds.

Fig. 5. The ideal functionality $\mathcal{F}_{\text{SORT}}$

It is straightforward to verify that $\mathcal{S}$ perfectly simulates the adversary and other components. That is, for any PPT $\{\mathcal{A}, \mathcal{Z}\}$, $\mathcal{Z}$ cannot distinguish between its interaction with $\mathcal{A}$ and $\pi_{\text{SORT}}[\mathcal{F}_{\text{VRF}}]$ or $\mathcal{S}$ and $\mathcal{F}_{\text{SORT}}$. Thus one can draw a conclusion that $\pi_{\text{SORT}}[\mathcal{F}_{\text{VRF}}]$ securely realizes $\mathcal{F}_{\text{SORT}}$ under the $\mathcal{F}_{\text{VRF}}$-hybrid model. $\square$

In the setting of [51], the authors elegantly proved that there exists a realistic implementation of π_{VRF} that can securely realize the ideal $\mathcal{F}_{\text{VRF}}$ under the Computational Diffie-Hellman (CDH) assumption in the random oracle model. Therefore with such a secure real-world implementation, our protocol $\pi_{\text{SORT}}[\pi_{\text{VRF}}]$, abbreviated as π_{SORT}, is computationally indistinguishable from $\pi_{\text{SORT}}[\mathcal{F}_{\text{VRF}}]$, and thus securely realizes $\mathcal{F}_{\text{SORT}}$ according to Lemma 1. Then for the analysis of the complicated BLOWN protocol, one can get rid of the repeated reduction proofs by conducting a hybrid experiment $\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$, where $\mathcal{F}_{\text{SORT}}$ is the ideal signature scheme presented in [52]. In Section 5.2, we report the salient features that can be realized by $\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$ with the ideal combinatorial functionalities $[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$. Thus we need to show that the real BLOWN protocol $\pi_{\text{B}}[\pi_{\text{SIG}}, \pi_{\text{SORT}}]$ (π_{SIG} is a secure EUF-CMA digital signature scheme) and $\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$ are computationally indistinguishable so that $\pi_{\text{B}}[\pi_{\text{SIG}}, \pi_{\text{SORT}}]$ can inherit all features of $\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$.

Theorem 1. *With the same security parameter λ, for each PPT $\{\mathcal{A}, \mathcal{Z}\}$, it holds that there is a PPT $\mathcal{S}$ such that*

$$\text{EXEC}^{\mathcal{A}, \mathcal{Z}}_{\pi_{\text{B}}[\pi_{\text{SIG}}, \pi_{\text{SORT}}]} \approx \text{EXEC}^{\mathcal{S}, \mathcal{Z}}_{\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]}, \qquad (2)$$

where "$\approx$" means computationally indistinguishable.

Proof. With a real digital signature protocol π_{SIG}, we obtain $\pi_{\text{B}}[\pi_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$, which is a protocol under the $\mathcal{F}_{\text{SORT}}$-hybrid model. From Lemma 1, one can see that it holds for each PPT $\mathcal{A}$ and $\mathcal{Z}$, the protocol π_{SORT} securely realizes $\mathcal{F}_{\text{SORT}}$. According to the universal composition theorem,

it holds that for any adversary $\mathcal{A}_{\mathcal{F}_{\text{VRF}}}$, there exists an adversary $\mathcal{A}_{\mathcal{F}_{\text{SORT}}}$ such that for any environment $\mathcal{Z}$, we have

$$\text{EXEC}^{\mathcal{A}_{\mathcal{F}_{\text{VRF}}}, \mathcal{Z}}_{\pi_{\text{B}}[\pi_{\text{SIG}}, \pi_{\text{SORT}}]} \approx \text{EXEC}^{\mathcal{A}_{\mathcal{F}_{\text{SORT}}}, \mathcal{Z}}_{\pi_{\text{B}}[\pi_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]}, \qquad (3)$$

Let $\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$ be a protocol under the $\mathcal{F}_{\text{SIG}}$-hybrid model with a fixed $\mathcal{F}_{\text{SORT}}$. Making use of an EUF-CMA digital signature scheme π_{SIG} that securely realizes $\mathcal{F}_{\text{SIG}}$, we have

$$\text{EXEC}^{\mathcal{A}_{\mathcal{F}_{\text{SIG}}}, \mathcal{Z}}_{\pi_{\text{B}}[\pi_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]} \approx \text{EXEC}^{\mathcal{A}_{\mathcal{F}_0}, \mathcal{Z}}_{\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]}, \qquad (4)$$

where $\mathcal{A}_{\mathcal{F}_0}$ is a dumb adversary. Combining (3) and (4), one can construct the simulator $\mathcal{S}$ that can run $\mathcal{A}_{\mathcal{F}_{\text{SORT}}}, \mathcal{A}_{\mathcal{F}_{\text{VRF}}}, \mathcal{A}_{\mathcal{F}_0}$ and forward messages between the adversary and $\mathcal{Z}$ so that $\mathcal{Z}$ cannot distinguish the interactions with $\pi_{\text{B}}[\pi_{\text{SIG}}, \pi_{\text{SORT}}]$ from those with $\pi_{\text{B}}[\mathcal{F}_{\text{SIG}}, \mathcal{F}_{\text{SORT}}]$. $\square$

5.2 Persistence and liveness

We first formulate a state machine $\mathbb{S}$ with the following four states: `START`, `LEADER`, `COMMIT`, `FINAL`.

Definition 1. *(`START` State). The system is in `START` state when the following conditions hold: (1) $|\{v | v \in V, l_v > 0\}| > 1$; (2) the honest nodes that accepted B_v^{k-1} in the last epoch have finished initialization.*

Definition 2. *(`LEADER` State). The system is in `LEADER` state when the following conditions hold: (1) there is a node v with $l_v > 0$ and $\forall u \in V \setminus \{v\}, l_u = 0$; (2) $j = 0$; (3) the size of v's transaction stack $|txp_v| = 0$.*

Definition 3. *(`COMMIT` State). The system is in `COMMIT` state when the following conditions hold: (1) there is a node v with $l_v > 0$ and $\forall u \in V \setminus \{v\}, l_u = 0$; (2) $0 < j < c \cdot i_k$.*

Definition 4. *(`FINAL` State). The system is in `FINAL` state if one of the following two conditions holds: (1) each honest node v*

has received a valid m_B and accepted the block B_u^k; (2) honest nodes did not receive a block in the $(c+1)i_k$-th round.

Garay *et al.* [53] proved that a secure distributed ledger should satisfy persistence and liveness properties. Let tx_i^j be the j-th transaction of the i-th block (the 0-th block is the genesis block). We say tx_i^j is *t-stable* when the current block index is larger than $i + t$, where $t > 0$. Then the persistence and liveness properties that BLOWN should guarantee can be defined as follows.

Definition 5. *Persistence and liveness. Persistence states that if an honest node v proclaims a transaction tx_i^j as t-stable, other honest nodes, if queried, either report the same result or report error messages. On the other hand, the liveness property states that if an honest node generates the transaction tx_i^j and contends to broadcast it in phase two, the BLOWN protocol can add it to the blockchain within finite epochs w.h.p.*

Kiayias and Panagiotakos [54] showed that persistence and liveness can be derived from the following three more concrete properties: chain growth, common prefix, and chain quality.

Definition 6 (Chain growth property). *Consider two chains $\mathcal{C}_1, \mathcal{C}_2$ possessed by two honest nodes at the onset of two epochs $e_1 < e_2$ with e_2 at least k epochs ahead of e_1. It holds that $len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq \tau \cdot k$, where τ is the speed coefficient with $\tau \in (0,1]$ and $k \in \mathbb{N}$.*

Definition 7 (Common prefix property). *The chains $\mathcal{C}_1, \mathcal{C}_2$ possessed by two honest nodes at the onset of the epoch $e_1 < e_2$ satisfy $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$, where $k \in \mathbb{N}$ and $\mathcal{C}_1^{\lceil k}$ denotes the chain obtained by removing the last k blocks from $\mathcal{C}_1$, and $\preceq$ denotes the prefix relation.*

Definition 8 (Chain quality property). *Consider any portion of length at least l of the chain possessed by an honest party at the onset of an epoch. The ratio of the blocks originated from the adversary is at most $1 - \mu$, where $\mu \in (0,1]$ is the chain quality coefficient.*

In the remainder of this section, we prove that $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ satisfies chain growth, common prefix, and chain quality properties, indicating that BLOWN guarantees persistence and liveness.

5.2.1 Chain Growth

$\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ meets chain growth as claimed in Theorem 2. We prove Theorem 2 by two steps: 1) each epoch must be terminated within a finite time (or $\mathbb{S}$ never enters a deadlock); 2) the chain growth property should quantify the blockchain growing speed such that new blocks are added to a chain with a speed coefficient $\tau \in (0,1]$. Concretely, we first prove that $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ ensures robust communication channels, as without which the protocol can hardly proceed. With such a communication channel, $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ supports a successful leader election, which provides correctness, efficiency, and practicality. Then we perform an analysis on the $\mathbb{S}$ of BLOWN to end the proof of Theorem 2.

Theorem 2. *It holds for $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ that each epoch can terminate in $O(cw_{max}\lambda)$, and there are $O(cw_{max})$ transactions*

added to the blockchain at each epoch w.h.p., at the speed coefficient (following Definition 6) $\tau = 0.5$.

To start with, we need to prove Theorem 3, which states that $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ can ensure a robust communication channel. Recall that the distance between any two nodes is bounded by $R_0 = (P/\beta\theta)^{1/\alpha}$. Therefore for $\forall v \in V$, $D_{R_0}(v)$ can cover all the neighbors of node v so that if at least one node $u \in N_{R_0}(v)$ transmits a message, v would either receive the message or sense a busy channel.

Theorem 3. *If $N_{R_0}(v) \neq \varnothing$, it holds true for BLOWN that runing at least $F = \Omega((T \log N)/\epsilon + (\log N)^4/(\gamma\epsilon)^2)$ rounds leads to at least $(1 - \epsilon\beta')\rho e^{\frac{-\rho}{1-\hat{p}}} F$ rounds of successful transmissions against any $((1 - \epsilon), T)$-bounded adversary w.h.p., where $\gamma = O(1/(\log T + \log \log N))$ and ρ is a constant.*

Proof. To prove Theorem 3, we divide $D_{R_0}(v)$ into six sectors of equal angles centered at v, and denote an arbitrary sector as S. Then we refer to $\bar{p}_v = \sum_{w \in S \setminus \{v\}} p_w$ as the aggregated transmission probability of the neighbors of v, and p_S denotes the aggregated transmission probability of all the nodes in S. Lemma 4 can be proved utilizing Lemma 2 and 3, whose proofs can be found in [55]. We divide the F into $(c \log N)/\epsilon$ consecutive subframes, with each consisting of $c(T + (\log N)^3/(\gamma^2\epsilon))$ rounds.

Lemma 2. *Consider any node v in S. If $\bar{p}_v > 5 - \hat{p}$ during all rounds of a subframe I' of I and at the beginning of I', $T_v \leq \sqrt{F}$, then p_v is at most $1/N^2$ at the end of I', w.h.p.*

Lemma 3. *For any subframe I in F and any initial value of p_S in I there is at least one round in I with $p_S \leq 5$ w.h.p.*

Lemma 4. *For any subframe I_k in I, if $p_S \leq 5$ occurs during the past subframe I_{k-1}, $p_S \leq 5e^2$ holds throughout I_k w.m.p.*

Proof. Let p_S^t be the cummulative transmission probability of nodes in S at round t. Assume the probability that all nodes in S are not transmitting is q_0, the probability that only one node in S is transmitting is q_1, and the probability that at least two nodes in S are transmitting is q_2. Then one can obtain the upper bound of the expectation of p_S^{t+1} as follows:

$$E[p_S^{t+1}] \leq q_0(1+\gamma)p_S^t + q_1(1+\gamma)^{-1}p_S^t + q_2 \cdot p_S^t. \quad (5)$$

This upper bound holds true even if we consider the rounds when $c_v > T_v$, which decreases p_S. Let E_2 be the event when at least two nodes in S transmit. If E_2 does not happen, $q_2 = 0$ and Eq. (5) becomes

$$E[p_S^{t+1}] = \frac{q_0}{q_0 + q_1}(1+\gamma)p_S^t + \frac{q_1}{q_0 + q_1}(1+\gamma)^{-1}p_S^t. \quad (6)$$

If $p_S > 5$, we have $q_1 \geq p_S \cdot q_0 \geq 5q_0$. Hence,

$$E[p_S^{t+1}] \leq [\frac{(1+\gamma)}{6} + \frac{5(1+\gamma)^{-1}}{6}]p_S^t \leq (1+\gamma)^{-1/2}p_S^t. \quad (7)$$

Considering the case where E_2 might happen, one can rewrite $E[p_S^{t+1}]$ as

$$E[p_S^{t+1}] \leq [q_2 + (1-q_2)(1+\gamma)^{-1/2}]p_S^t. \quad (8)$$

Since $q_2 = 1 - q_0 - q_1 < 1 - p_S e^{\frac{p_S}{1-\hat{p}}}$, we have

$$E[p_S^{t+1}] \leq [1 - p_S e^{\frac{p_S}{1-\hat{p}}} + (1+\gamma)^{-1/2}p_S e^{\frac{p_S}{1-\hat{p}}}]p_S^t. \quad (9)$$

Suppose in the subframe I_{k-1} there is a round t with $p_S > 5$. One can find a time interval $I' \subseteq I_{k-1}$, which satisfies $5 < p_S < 5e$ during I', $p_S < 5$ just before I', and $p_S > 5e$ at the end of I'. We intend to bound the probability at which such I' happens. Let $\phi = \log_{1+\gamma}[(1 - p_S e^{\frac{p_S}{1-\bar{p}}} + (1+\gamma)^{-1/2} p_S e^{\frac{p_S}{1-\bar{p}}})^{-1}]$. Since γ is sufficiently small, we have $\phi \in (0.5, 1)$ and $E[p_S^{t+1}] \leq (1 + \gamma)^{-\phi}$. On the other hand, $p_S^{t+1} \leq (1 + \gamma)p_S^t \leq (1 + \gamma)^{2\phi} p_S^t$. Then let $X_S^t = \log_{(1+\gamma)} p_S^t + \sum_{i=0}^{t-1} \phi_k$ and $X_S^0 = \log_{(1+\gamma)} p_S^0$, it is easy to verify that $E[X_S^{t+1}] = X_t$ and $X_S^{t+1} \leq X_S^t + c_{t+1}$, where $c_{t+1} = 3\phi_t$. Leveraging the Azuma–Hoeffding Inequality, it holds that

$$P[X_S^T - X_S^0 > \delta] \leq e^{\frac{-\delta^2}{2\sum_{k=1}^T c_k^2}}, \tag{10}$$

for $\delta = 1/\gamma + \sum_{k=0}^{T-1} \phi_k$. Therefore

$$P[\log_{(1+\gamma)} p_S^T/p_S^0 > 1/\gamma] \leq e^{\frac{-\delta^2}{2\sum_{k=0}^{T-1}(3\phi_k)^2}}. \tag{11}$$

Let $\psi = \sum_{k=0}^{T-1}(\phi_k)^2$, we have $e^{\frac{-\delta^2}{2\sum_{k=0}^{T-1}(3\phi_k)^2}} = \frac{(\psi+1/\gamma)^2}{18\psi} \geq \frac{1}{9\gamma}$. Hence,

$$P[\log_{(1+\gamma)} p_S^T/p_S^0 > 1/\gamma] \leq e^{-1/9\gamma} \leq \frac{1}{\log^c N}, \tag{12}$$

for any constant c if $\gamma = O(1/(\log T + \log \log N))$. Note that $\log_{(1+\gamma)} p_S^T/p_S^0 > 1/\gamma$ indicates $p_S^T/p_S^0 > e$. Considering $p_S^0 > 5$ at the beginning of a subframe I', $P[\log_{(1+\gamma)} p_S^T/p_S^0 > 1/\gamma]$ is the probability at which the aggregated probability of the nodes in S exceeds $5e$ at the end of I'. Hence we prove that if $p_S < 5$ holds at the beginning of I_{k-1}, $p_S < 5e$ holds throughout I_{k-1} w.m.p. Also, it is analogous to prove that if $p_S < 5e$ is true at the beginning of I_{k-1}, $p_S < 5e^2$ holds throughout I_{k-1} w.m.p. Hence, if $p_S \leq 5$ happens during the past subframe I_{k-1}, $p_S < 5e$ holds throughout I_k w.m.p. Since $p_S < 5e$ holds at the beginning of I_k, $p_S < 5e^2$ holds throughout I_k w.m.p., which proves the lemma. $\square$

Lemma 5. $(1 - \epsilon\beta')$-fraction of subframes in F satisfy $p_V \leq \rho$ w.h.p, where $p_V = \sum_{v \in V} p_v$ is the aggregated probability of all nodes, and ϵ, β', ρ are constants.

Proof. Let us focus on a fixed subframe I_k and its previous subframe I_{k-1}. Lemma 3 indicates that there is at least one round in I_{k-1} with $p_S \leq 5$ w.h.p. Then it follows from Lemma 4 that if there is at least one round in I_{k-1} with $p_S \leq 5$, $p_S < 5e^2$ holds throughout I_k w.m.p. Define a subframe I to be *good* if $p_S \leq 5e^2$ holds throughout I, and otherwise I is *bad*. Then it follows from the Chernoff bounds that at most $\epsilon\beta'/6$ of the subframes in F are bad w.h.p. Since D_{R_0} consists of six sectors and covers all nodes in V, there is at least $(1-\epsilon\beta')$-fraction of subframes in which the aggregated probability $p_V = \sum_{v \in V} p_v = \sum_{v \in N_{R_0}(v)} p_v$ is bounded by $\rho = 6 \times 5e^2 = 30e^2$, which completes the proof. $\square$

Then, the probability on which there exists one successful transmission is given by

$$\begin{aligned}
\sum_{v \in V} p_v \prod_{w \in V \setminus v} (1 - p_w) &\geq \sum_{v \in V} p_v \prod_{w \in V} (1 - p_w) \\
&\geq \sum_{v \in V} p_v \prod_{w \in V} e^{\frac{-p_w}{1-\bar{p}}} \\
&= \sum_{v \in V} p_v e^{\frac{-p_V}{1-\bar{p}}} \\
&= p_V e^{\frac{-p_V}{1-\bar{p}}} \\
&\geq \rho e^{\frac{-\rho}{1-\bar{p}}}.
\end{aligned} \tag{13}$$

$\square$

With the robust communication guarantee, we next prove that BLOWN can support a successful leader election, which is the core of the protocol. Most leader election algorithms in wireless networks are only responsible for reaching the state at which one node is the leader and others are followers. Our algorithm goes one-step further by ensuring that all nodes have an identical view of the network after leader election, which is crucial to the our protocol, as shown in Theorem 4.

Theorem 4. *(Successful leader election). Let w_{max} be the maximum weight among all nodes and λ be a constant to be determined. $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ ensures a successful leader election while satisfying the following three properties: 1) **Correctness**: only one node is left as the leader with a positive l_v at the end of P_1; 2) **Efficiency**: the success of leader election can be achieved within $O(w_{max}\lambda)$ rounds w.h.p.; 3) **Practicality**: the leader and the followers should have the knowledge regarding who is the leader and at which round the leader is elected.*

Proof. We prove the three properties in order. During a leader election process, all nodes contend for broadcasting messages in P_1 until only one node is left with a positive l_v, which can always be achieved inevitably. This can be proved by contradiction. Without loss of generality, we assume that there are two nodes left with a positive l_v. If these two nodes broadcast messages at the same round, they can not receive messages from each other simultaneously. Therefore, there is no chance for two nodes to receive messages in the same round, and there must be only one node surviving at the end. One can trivially expand this result to the cases with $3, 4, \cdots, N$ nodes left with positive l_v values, thus proving that the protocol can always lead to the state when only one node survives as the leader with a positive l_v.

To prove the efficiency property, we resort to Theorem 3, which shows that a constant fraction of the rounds have successful transmissions w.h.p. Concretely, a successful communication should happen once every $\lambda = (1 - \epsilon\beta)^{-1} \rho^{-1} e^{\frac{\rho}{1-\bar{p}}}$ rounds on average w.h.p. Then leader election can be finished in $O(w_{max}\lambda)$ rounds w.h.p. This indicates that $O(1)$ number of rounds can lead to a successful leader election and the communication complexity is not directly related to the network size.

To prove the practicality, we denote E_v as the event that v broadcasts a message in slot one and senses an idle channel in slot two. In this case, v would know itself as the leader. Let p_v be the probability that v broadcasts a

message in slot one, $p_v^{(0)}$ be the probability that v broadcasts a message and there is also at least one node u with $l_u > 0$ broadcasting a message in slot one, $p_v^{(1)}$ be the probability that v broadcasts a message and there exists at least one node u with $l_u > 0$ sensing the channel in slot one, and $p_v^{(2)}$ be the probability that v broadcasts a message in slot one and $l_u = 0, \forall u \in V \setminus \{v\}$. Certainly, $p_v = p_v^{(0)} + p_v^{(1)} + p_v^{(2)}$. If E_v happens, v senses an idle channel in slot two. Then $p_v^{(0)} = 0$ since if u broadcasts a message in slot one, a follower f senses interference and thus broadcasts an m in slot two so that v senses interference in slot two, which contradicts our assumption. Also, $p_v^{(1)} = 0$ because if there exists a node u with $l_u > 0$ sensing the channel in slot one, u has to broadcast a message in slot two which also contradicts the assumption. Therefore, we obtain the result that if E_v happens, v can confirm itself as the unique leader.

Correspondingly, we denote E_f as the event that a follower f recognizes v as the leader when f receives a message from v and obtains $\mathcal{I} + \mathcal{N} < \theta$ in slot one, then senses an idle channel in slot two. Let p_f be the probability that f receives a message from v and obtains $\mathcal{I} + \mathcal{N} < \theta$ in slot one, $p_f^{(0)}$ be the probability that there is at least one node $u \in V \setminus \{v\}$ with $l_u > 0$ sensing the channel in slot one, $p_f^{(1)}$ be the probability that v is the unique leader; then we have $p_f = p_f^{(0)} + p_f^{(1)}$. Assume E_v happens, we have $p_f^{(0)} = 0$ since if $p_f^{(0)} \neq 0$, u has to broadcast a message in slot two and thus a follower senses interference, which contradicts our assumption. As a result, E_f indicates that v is the unique leader. Additionally, the round at which a successful leader election happens can be found when E_v and E_f occur simultaneously, which ends the proof of the third property. $\square$

Utilizing Theorem 3 and 4 as intermediate conclusions, we are finally ready to prove Theorem 2.

Proof. The time between LEADER and COMMIT is fixed to $j = c \cdot i_k$ rounds, where c is an adjustable constant parameter according to different implementation scenarios. If the leader does not broadcast a block in the $(c+1)i_k$-th round, the state transits to the final state since $\mathbb{S}$ satisfies the second condition of a FINAL state. Then $\mathbb{S}$ starts the next epoch. According to Theorem 4, each epoch can be terminated in $O(cw_{max}\lambda)$ w.h.p., and there should be $O(cw_{max})$ transactions added to the blockchain in each epoch w.h.p.

Assume an honest node v generates the transaction tx_i^j and contends to broadcast it in P_2. The transaction can be received by an honest leader with probability at least $p = cw_{max}/N$ in each epoch. By applying the Chernoff bound, we obtain that tx_i^j can be added to the blockchain within n epochs with probability at least $1 - e^{-\frac{(np-1)^2}{2}}$, where n is the number of epochs when v broadcasts tx_i^j. The above analysis indicates that $\mathbb{S}$ has no chance of staying at a deadlock in any epoch. Considering the assumption that honest nodes control more than 50% coins, $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ with ideal functionalities $[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ can ensure a fair sortition based on the nodes' coin distribution. Thus, with probability at least 50%, an honest node can be selected as a leader to propose a new block. When two chains $\mathcal{C}_1, \mathcal{C}_2$ possessed by two honest nodes at the onset of two epochs

$e_1 < e_2$ with e_2 at least k epochs ahead of e_1, it holds that $len(\mathcal{C}_2) - len(\mathcal{C}_1) \geq \tau \cdot k$, where $\tau = 0.5$. This completes the proof of Theorem 2.

$\square$

5.2.2 Common Prefix

Theorem 5. $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ *satisfies the common prefix property (following Definition 7).*

Proof. In $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ a node can directly append a new block B_u^k to its local blockchain only when $\mathcal{F}_{SIG}$ and $\mathcal{F}_{SORT}$ answer with 1 when being queried. Therefore, an adversary who intends to disguise itself as a leader to propose a block should fail since it cannot break $[\mathcal{F}_{SIG}$ and $\mathcal{F}_{SORT}]$. However, a malicious leader (an adversary who wins the leader election) can still diverge the global distributed ledger to cause a Δ-fork defined in Definition 10.
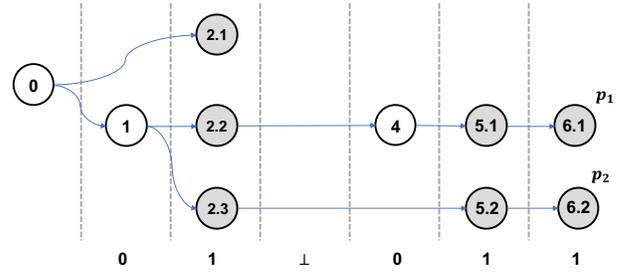

Fig. 6. A simple example to illustrate the concepts of string, Δ-fork, and divergence. Specifically, we have $s_6 = \{0, 1, \perp, 0, 1, 1\}$, $l(p_1) = 5$, and $l(p_2) = 4$. The tree presented here is a Δ-fork with $\Delta = 1$, and $|div(p_1, p_2)| = 4$.

Definition 9 (String). *Consider an epoch e_k during the execution of functionality $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ with adversary $\mathcal{A}$ and environment $\mathcal{Z}$. Let $S = \{e_1, \cdots, e_k\}$ denote a sequence of epochs of length k. The string $s = \{0, 1, \perp\}^k$ of S is defined so that $s_i = 1$ if the adversary controls the epoch leader of e_i and broadcasts a block, $s_i = 0$ if an honest node controls the epoch and broadcasts a block, and $s_i = \perp$ if no block is broadcast. We say that the index i is adversarial if $s_i = 1$ and honest otherwise.*

W.l.o.g., let $s_0 = 0$ for e_0 meaning that the genesis block has an honest index.

Definition 10 (Δ-Fork). *Let string $s = \{0, 1, \perp\}^k$ of S and Δ be a non-negative interger. A Δ-fork is a directed, acyclic, rooted tree $F = (V, E)$ in which the two longest paths p_1 and p_2 satisfy $|l(p_1) - l(p_2)| \leq \Delta$, where a path p refers to a road from the root to a leaf and $l(p)$ is the hop-count (length) of the path p.*

Definition 11 (Divergence). *Denote the divergence of two paths p_1 and p_2 in a Δ-Fork as $div(p_1, p_2)$, which is defined as*

$$div(p_1, p_2) = \max\{l(p_1), l(p_2)\} - l(p_1 \cap p_2), \quad (14)$$

where $l(p_1 \cap p_2)$ is the legnth of the common path of p_1 and p_2, and $div(p_1, p_2)$ is non-negative.

Lemma 6. *The common prefix property is satisfied if and only if for any pair of paths $p_i, p_j, i \neq j$, in a Δ-fork, $div(p_i, p_j) \leq k$.*

Proof. For the "only if" direction, we assume that there exits a path p_1, p_2 (w.l.o.g., $l(p_1) > l(p_2)$) such that $div(p_1, p_2) > k$. That is $\max\{l(p_1), l(p_2)\} - l(p_1 \cap p_2) = l(p_1) - l(p_1 \cap p_2) >$

k. Let V_1 (V_2) be the set of honest nodes that store the distributed ledger as the path p_1 (p_2). Once querying a local blockchain, any $v_1 \in V_1$ ($v_2 \in V_2$) responds with $\mathcal{C}_1$ ($\mathcal{C}_2$). Denote the latest point of the common path $p_1 \cap p_2$ as $\hat{v}$, which is also called a bifurcation point. The path $p_1^{\lceil k}$ that is obtained by truncating the last k vertices of p_1 still covers $\hat{v}$, which is not the endpoint of p_1 since $l(p_1) - k > l(p_1 \cap p_2)$. Denote the endpoint of p_1 as $end(p_1)$. Then the blocks corresponding to the points from $\hat{v}$ to $end(p_1)$ are included in $\mathcal{C}_1^{\lceil k}$, but the block mapped to $end(p_1^{\lceil k})$ is not included in $\mathcal{C}_2$, thus violating the common prefix property. For the "if" direction, assuming that the common prefix is violated, there exists a pair of ledgers $\mathcal{C}_1$ and $\mathcal{C}_2$ for $e_1 < e_2$ such that $\mathcal{C}_1^{\lceil k} \npreceq \mathcal{C}_2$. Mapping such blockchains to two distinct paths p_1, p_2, the endpoint $end(p_1^{\lceil k})$ corresponding to the latest block in $\mathcal{C}_1^{\lceil k}$ is not covered by p_2 and comes after $\hat{v}$. By the definition of divergence, $div(p_1, p_2) = \max\{l(p_1), l(p_2)\} - l(p_1 \cap p_2) > k$. $\square$

Here one can define a common prefix violation as the case when there exit two paths p_1, p_2 in a Δ-fork with $|div(p_1, p_2)| > k$. To prove Theorem 5, we need to show that a common prefix violation happens with an extremely small probability. Generally speaking, $\Delta \leq k$, and p_1, p_2 can be regarded as the respective paths that the honest nodes and adversary go through. This is based on the assumption that all honest nodes strictly follow the longest chain rule, while the adversary focuses on increasing the length of an illegal chain (e.g., including a double-spend transaction). Therefore, a common prefix violation can also be interpreted as a race between honest nodes and the adversary that lasts for more than k blocks, but their view paths still follow $|l(p_1) - l(p_2)| \leq \Delta$. Let $X_i \in \{\pm 1\}$ (for $i = 1, 2, \cdots$) denote a series of independent random variables for which $Pr[X_i = 1] = (1 - \epsilon)/2$. Note that $\epsilon \in (0, 1)$ is satisfied in functionality $\pi_B[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORsT}}]$ since the adversary controls less than 50% coins and the protocol adopts a hybrid $[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ to ensure that the probability of the adversary being a leader is less than 1/2. Consider k epochs of the biased walk beginning at the bifurcation point. The resulting value is tightly concentrated at $-\epsilon k$. By applying the Chernoff bound, for each k random walk hitting problem, we have

$$Pr[X_k < \Delta] \leq e^{-(1 - \Delta/\epsilon k)^2 \epsilon k/2} = e^{-O(k)}, \quad (15)$$

where $\Delta \ll k$. This indicates that $\pi_B[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ satisfies the common prefix property w.h.p., which completes the proof. $\square$

5.2.3 Chain Quality

The chain quality property requires that a certain fraction of the blocks should satisfy high quality standards (high-quality blocks are the ones generated absolutely by honest nodes). Chain quality can be threatened by Sybil attacks which are particularly harmful in wireless networks [56]. In a Sybil attack, an attacker can behave as many nodes by illegitimately claiming massive identities or impersonating others. A successful attacker chosen as a leader can deny to broadcast a new block or broadcast an invalid block. Since honest nodes can neither wait for more than $c \cdot i_k$ rounds

in P_2 nor accept invalid blocks, the attacker cannot hinder the system from changing from LEADER state to the FINAL state. However, an attacker can make an epoch wasted without any new block being added to the blockchain, thereby harming the liveness. Our BLOWN protocol prevents Sybil attacks and ensures liveness under the assumption that all malicious nodes control no more than 50% coins of the entire network.

Consider one epoch. $\mathcal{F}_{\mathrm{SORT}}$ provides a binomial distribution as $B(k; w_v, p) = \binom{w_v}{k} p^k (1 - p)^{w_v - k}$, which has a salient property that splitting coins into multiple sub-users does not give attackers any advantage. In particular, suppose an attacker splits its account balance $w_{\mathcal{A}}$ into $w_{\mathcal{A}}^1, w_{\mathcal{A}}^2, \cdots, w_{\mathcal{A}}^n$, thus each sub-user has a binomial distribution as $X_{\mathcal{A}}^i \sim B(w_{\mathcal{A}}^i, p)$. However, splitting coins does not increase the sum of the values of the leader counter controlled by the attacker since $(X_{\mathcal{A}}^1 + X_{\mathcal{A}}^2 + \cdots + X_{\mathcal{A}}^n) \sim B(w_{\mathcal{A}}^1 + w_{\mathcal{A}}^2 + \cdots + w_{\mathcal{A}}^n, p)$. Also, splitting coins decreases the maximum of the leader counter of the sub-users, which makes it harder for a sub-user to survive in P_1. Without loss of generality, suppose each node has an equal value of balance. Then at each epoch, the probability of a malicious node being chosen as a leader is no more than 50%.

Theorem 6. *Given that the ratio of the adversarial coins $\alpha < 1/2$, $\pi_B[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ satisfies the chain quality property with $\mu = 1 - (1 + \delta)\alpha$, where $\delta \in (0, 1)$.*

Proof. Let X_i denote the event where the ith epoch has an adversarial leader. We have $\mathbb{E}[X_i] \leq \alpha l$. Applying the Chernoff bound we obtain

$$Pr[X \geq (1 + \delta)\alpha l] \leq e^{-O(l)}. \quad (16)$$

Then the probability that the ratio β for the blocks originated from the adversary is at most $(1 + \delta)\alpha$ is given as

$$Pr[\beta \leq (1 + \delta)\alpha] = 1 - Pr[X \geq (1 + \delta)\alpha l] \geq 1 - e^{-O(l)}. \quad (17)$$

When l is sufficiently large, $\beta \leq (1 + \delta)\alpha$ w.h.p. Thus we complete the proof of the chain quality property with $\mu = 1 - (1 + \delta)\alpha$. Note that even though $\mu = 1 - (1 + \delta)\alpha$ blocks can be proposed by the adversary, these blocks only contain a small fraction of malicious ones (jointly ensured by the chain growth and common prefix properties). $\square$

Therefore we can conclude that $\pi_B[\mathcal{F}_{\mathrm{SIG}}, \mathcal{F}_{\mathrm{SORT}}]$ satisfies the chain growth, common prefix, and chain quality properties, thus guaranteeing persistence and liveness. By applying Theorem 1, BLOWN (i.e., $\pi_B[\pi_{\mathrm{SIG}}, \pi_{\mathrm{SORT}}]$) naturally ensures persistence and liveness.

6 SIMULATION STUDY

In this section, we implement a simulator to study how various parameters impact the performance of our BLOWN protocol. Specifically, in Section 6.1, we analyze the one-epoch execution of BLOWN to show its correctness and efficiency. Then we present the performance of BLOWN under various jamming and Sybil attack scenarios in Section 6.2.

In our simulation, we use the crypto library of golang[3] and adopt ed25519 for digital signatures, with 64-byte private key, 32-byte public key, and 64-byte signature. Public

3. https://github.com/golang/crypto

keys are broadcast to all nodes on the preset of our simulations. Besides, The Sortition algorithm is implemented with the VRF provided by CONIKS[4]. We employ two types of 2-dimensional planes of size $d \times d$ units, where $d = 10$ or $d = \sqrt{N}$ with N known as the network size. Nodes are randomly generated and distributed in the plane and no two nodes can have the same coordinates. The unit for an epoch length is round. If not stated otherwise, we adopt the following parameters $\alpha = 4$, $\beta = 2$, $\theta = 2$, $P = \beta\theta(\sqrt{2}d)^{\alpha}$, $\hat{p} = 0.1$, $T = 60$, $\epsilon = 0.3$, $\gamma = 0.1$, $w_v = 20$, $\tau = W/2$, and $c = 10$. Besides, nodes are uniformly distributed and the percentage of the Sybil nodes is 0% by default. Without loss of generality, all parameter values are chosen carefully to reflect various real-world cases, but not to aim to optimize the performance. All the experiments are performed under a CentOS 7 operating system running on a machine with an Intel Xeon 3.4 GHz CPU, 120 GB RAM and 1 TB SATA Hard Drive. All the reported results are the average of 100 runs, unless stated otherwise.
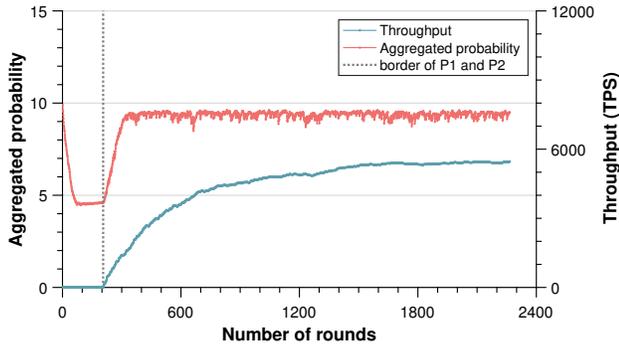
6.1 Correctness and Efficiency

Fig. 7. Aggregated probability and throughput *vs.* number of rounds, to demonstrate the convergence behavior, where $density = 1, d = 10, N = 100$, and nodes are uniformly distributed.

One-Epoch Execution. We denote by $|txp_t|$ the number of transactions received by the leader within $t = i + j$ rounds, with i and j respectively being the number of rounds in P_1 and P_2. Given that the unit slot time for IEEE 802.11 is set to be $50\mu s$, we have throughput as

$$\text{Throughput} = \frac{|txp_t|}{i \times 100\mu s + j \times 50\mu s} \quad (18)$$

since $r_{1,i}^k$ has two slots while $r_{2,j}^k$ has only one slot. Fig. 7 presents a typical example to illustrate the convergence of the aggregated probability $p_V = \sum_{v \in V} p_v$ and throughput during one-epoch execution, where $p_V = N \times \hat{p} = 10$ in the outset. There is a gray dash borderline distinguishing P_1 and P_2. Since BLOWN can rapidly adjust the initial parameters by multiplicatively increasing or deceasing p_v, p_V adapts rapidly to reduce the noise in the channel to help achieve successful communications. Therefore, it only takes 206 rounds (corresponding to 0.206s in a real-world setting) to complete P_1. Such a quick adaptation contributes to the throughput of the entire protocol. In P_2, nodes all become

4. https://github.com/coniks-sys/coniks-go/tree/master/crypto/vrf

active to broadcast transactions enabling p_V to grow. The leader collects transactions from the 207th to the 2265th round, and a block is finalized at the 2266th round. Note that p_V and throughput respectively converge to 5399 TPS and 9.37, which are mean values calculated from the last 500 rounds. Besides, we evaluate cryptographic overhead (in ms, an average of 1000 repeated trials), including the overhead of signing a transaction (0.09 ms), verifying a transaction [0.21 ms], signing a block (1.20 ms), confirming a block (930.14 ms), Sortition(3.02 ms), and VerifySortition(4.57 ms).

Network Size. Next we simulate the performance as a function of the network size (or N), where nodes are scattered in the plane of size $d = \sqrt{N} \times \sqrt{N}$ following a uniform or Gauss distribution. As shown in Fig. 8(a), the epoch length slowly increases with a larger N with both uniform and Gauss distributions, which also means that the leader election costs more time for a larger N. On the other hand, throughput decreases with a larger N since the added nodes lead to heavier contention. However, because of the resiliency of our jamming resistant channel, throughput can converge to about 6000 TPS and 2000 TPS for the uniform and Gauss distribution, respectively. Compared with the uniform distribution, Gauss distribution always has a larger epoch length and lower throughput since denser nodes centrally aggregate, leading to stronger contention.

Network Density. We also investigate how the network density impacts on the performance of the BLOWN protocol. Nodes are uniformly distributed in a 10×10 plane, and $density = 0.2, 0, 3, \cdots, 2$. As shown in Fig. 8(b), the epoch length slowly increases from the 1867 to the 2464 rounds, with the density rising tenfold. The throughput decreases for larger density and approximately converges to 6000 TPS.

6.2 Jamming Attacks and Sybil Attacks

Jamming Attacks. Here we present our protocol's performance when confronting jammers who can choose different strategies with the constraint of $(1 - \epsilon)T$. We consider two types of jammers: *random* jammers that can randomly jam $(1 - \epsilon)T$ rounds at any interval of length T and *bursty* jammers who would jam $(1 - \epsilon)T$ consecutive rounds at any interval of length T. We test the epoch length and throughput when $\epsilon = 0.1, 0.15, \cdots, 0.5$, with a higher ϵ implying a lower attack frequency. The results are demonstrated in Fig. 9(a), which indicate that the epoch length decreases with the increasing ϵ due to the lower frequency of jamming attacks. Besides, ϵ does not significantly impact the throughput for both kinds of jammers. The epoch length increases faster with lower ϵ considering random jammers, indicating that random jammers are more powerful than bursty ones in BLOWN. This is because the introduction of T_v makes it easier to address continuous heavy contentions.

Sybil Attacks. In a Sybil attack, an attacker can control massive malicious nodes that compete for being a leader but refuse to collect transactions and propose blocks. In this circumstance, the epoch with a malicious leader would be abandoned so that there is no valid block to be accepted within such an epoch. Even though we already show in our protocol analysis that BLOWN can defend against Sybil attackers who control less than 50% wealth of the entire network, such attackers can harm the liveness of our protocol.
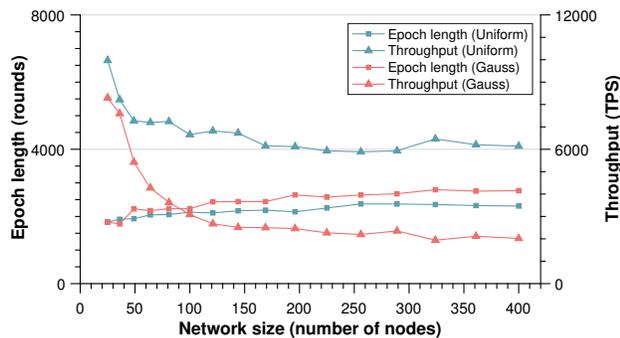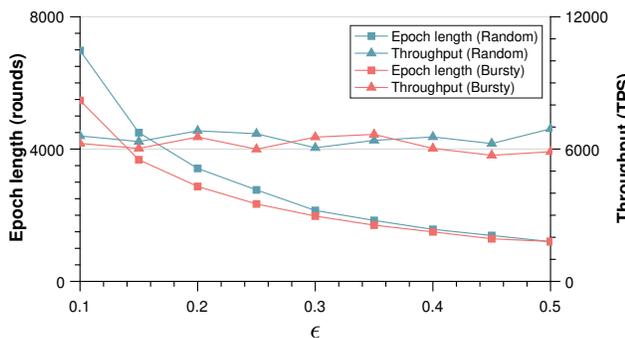
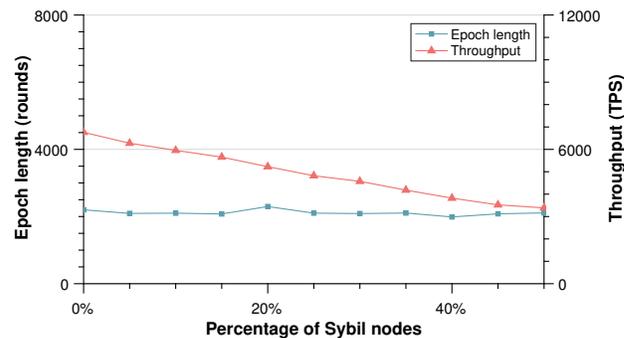(a) Epoch length and throughput *vs.* the network size N, where $density = 1, d = \sqrt{N} \times \sqrt{N}$.
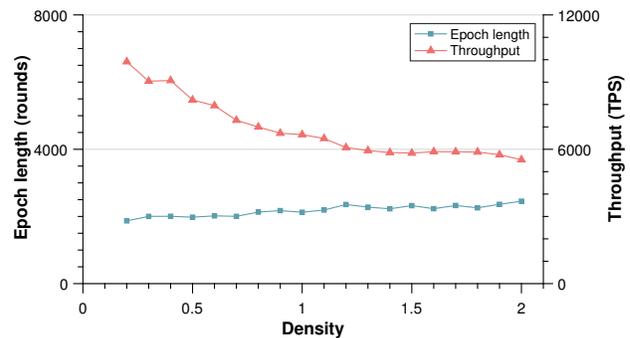
(b) Epoch length and throughput *vs.* the density, where $d = 10, N = 100$.

Fig. 8. The performance of BLOWN *vs.* network size (in a uniform or Gauss distribution) and density.

(a) Epoch length and throughput *vs.* ϵ, where $density = 1, d = 10, N = 100$.

(b) Epoch length and throughput with the percentage of Sybil nodes, where $density = 1, d = 10$, and $N = 100$.

Fig. 9. The performance of BLOWN when confronting jamming attacks (random jammers or bursty jammers) and Sybil attacks.

In Fig. 9(b), the percentage of Sybil nodes does not impact the epoch length since Sybil nodes are not absent from competing in the leader election. However, the throughput has an evident linear decline for a larger percentage of Sybil nodes. Compared to the setting without Sybil nodes, $50\%N$ Sybil nodes would decrease the throughput by 49.90%.

7 CONCLUSION AND FUTURE RESEARCH

In this paper, we propose a 2-phase blockchain protocol, namely BLOWN. BLOWN establishes a jamming-resistant communication channel and combines the Sortition algorithm and our newly proposed PoC consensus algorithm for efficient and secure leader election. Besides, BLOWN prevents double-spending attacks and Sybil attacks. Analysis and simulation results demonstrate the efficiency, effectiveness, and security properties of the BLOWN protocol. In our future research, we will investigate the multi-hop version of BLOWN, as well as the Byzantine fault-tolerant BLOWN in wireless ad hoc or fading channel settings. Also, it is neccessary to explore how practical attacks such as eclipse attacks, nothing-at-stake attacks, selfish-mining attacks can be mitigated by our protocol.

REFERENCES

[1] A. Tapscott and D. Tapscott, "How blockchain is changing finance," *Harvard Business Review*, vol. 1, no. 9, pp. 2–5, 2017.

[2] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1676–1717, 2018.

[3] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

[4] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 858–880, 2018.

[5] B. Soret, K. I. Pedersen, N. T. Jørgensen, and V. Fernández-López, "Interference coordination for dense wireless networks," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 102–109, 2015.

[6] W. Cheng, J. Yu, F. Zhao, and X. Cheng, "Ssdnet: Small-world super-dense device-to-device wireless networks," *IEEE Network*, vol. 32, no. 1, pp. 186–192, 2017.

[7] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018.

[8] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5g beyond," *IEEE Network*, vol. 33, no. 3, pp. 10–17, 2019.

[9] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," https://downloads.coindesk.com/research/whitepapers/bitcoin.pdf, White Paper, 2008.

[10] 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.

[11] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[12] D. Yu, L. Ning, Y. Zou, J. Yu, X. Cheng, and F. C. Lau, "Distributed spanner construction with physical interference: constant stretch and linear sparseness," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2138–2151, 2017.

[13] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *Proceedings of the 1st ACM workshop on Wireless security*, 2002, pp. 21–30.

[14] 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. [Online]. Available: https://doi.org/10.1109/COMST.2020.2969706

[15] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

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

[17] B. Wiki. Proof of burn. [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_burn

[18] Hyperledger. [Online]. Available: https://www.hyperledger.org/

[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 51–68.

[20] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake." *IACR Cryptology ePrint Archive*, vol. 2016, p. 919, 2016.

[21] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 475–490.

[22] D. Schwartz, N. Youngs, A. Britto *et al.*, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, 2014.

[23] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 17–30.

[24] Stellar. [Online]. Available: https://www.stellar.org/[Accessed: 2019-09-27]

[25] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.

[26] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 31–42.

[27] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, p. 11, 2014.

[28] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[29] O. Novo, "Blockchain meets iot: An architecture for scalable access management in iot," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.

[30] J. Feng, F. R. Yu, Q. Pei, J. Du, and L. Zhu, "Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems," *IEEE Transactions on Wireless Communications*, 2020.

[31] N. Malik, P. Nanda, X. He, and R. P. Liu, "Vehicular networks with security and trust management solutions: proposed secured message exchange via blockchain technology," *Wirel. Networks*, vol. 26, no. 6, pp. 4207–4226, 2020. [Online]. Available: https://doi.org/10.1007/s11276-020-02325-z

[32] S. Guo, Y. Qi, P. Yu, S. Xu, and F. Qi, "When network operation meets blockchain: An artificial-intelligence-driven customization service for trusted virtual resources of iot," *IEEE Network*, vol. 34, no. 5, pp. 46–53, 2020.

[33] S. Guo, X. Hu, S. Guo, X. Qiu, and F. Qi, "Blockchain meets edge computing: A distributed and trusted authentication system," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1972–1983, 2019.

[34] C. Liu, M. Xu, H. Guo, X. Cheng, Y. Xiao, D. Yu, B. Gong, A. Yerukhimovich, S. Wang, and W. Lv, "Tokoin: A coin-based accountable access control scheme for the internet of things," *IEEE Transactions on Mobile Computing*, 2021.

[35] C. Liu, H. Guo, M. Xu, S. Wang, D. Yu, J. Yu, and X. Cheng, "Extending on-chain trust to off-chain – a trustworthy vaccine shipping example," *IEEE Transactions on Computers*, 2021.

[36] F. Kuhn, N. Lynch, and C. Newport, "The abstract mac layer," in *International Symposium on Distributed Computing*. Springer, 2009, pp. 48–62.

[37] C. Newport, "Consensus with an abstract mac layer," in *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, 2014, pp. 66–75.

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

[39] D. Yu, Y. Zhang, Y. Huang, H. Jin, J. Yu, and Q.-S. Hua, "Exact implementation of abstract mac layer via carrier sensing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1196–1204.

[40] H. Moniz, N. F. Neves, and M. Correia, "Byzantine fault-tolerant consensus in wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2441–2454, 2012.

[41] G. Chockler, M. Demirbas, S. Gilbert, C. Newport, and T. Nolte, "Consensus and collision detectors in wireless ad hoc networks," in *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, 2005, pp. 197–206.

[42] G. Scutari and S. Barbarossa, "Distributed consensus over wireless sensor networks affected by multipath fading," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 4100–4106, 2008.

[43] T. C. Aysal, A. D. Sarwate, and A. G. Dimakis, "Reaching consensus in wireless networks with probabilistic broadcast," in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2009, pp. 732–739.

[44] D. Yu, Y. Zou, M. Xu, Y. Xu, Y. Zhang, B. Gong, and X. Xing, "Competitive age of information in dynamic iot networks," *IEEE Internet of Things Journal*, 2020.

[45] J. C. Ikuno, S. Pendl, M. Šimko, and M. Rupp, "Accurate sinr estimation model for system level simulation of lte networks," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 1471–1475.

[46] Y. Zhao, J. Wu, and S. Lu, "Efficient sinr estimating with accuracy control in large scale cognitive radio networks," in *2011 IEEE 17th International Conference on Parallel and Distributed Systems*. IEEE, 2011, pp. 549–556.

[47] K. J. Olszewski, "Sinr measurement method for ofdm communications systems," Aug. 21 2007, uS Patent 7,260,054.

[48] D. R. Jeske and A. Sampath, "Method of estimating a signal-to-interference+ noise ratio (sinr)," Feb. 27 2007, uS Patent 7,184,497.

[49] A. Abu-Dayya, "Determining sinr in a communications system," Nov. 23 1999, uS Patent 5,991,273.

[50] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 136–145.

[51] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 66–98.

[52] R. Canetti, "Universally composable signature, certification, and authentication," in *Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004.*, 2004, pp. 219–233.

[53] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[54] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols." *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 1019, 2015.

[55] A. Richa, C. Scheideler, S. Schmid, and J. Zhang, "A jamming-resistant mac protocol for multi-hop wireless networks," in *International Symposium on Distributed Computing*. Springer, 2010, pp. 179–193.

[56] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Third international*

symposium on information processing in sensor networks, 2004. IPSN 2004. IEEE, 2004, pp. 259–268.

8 APPENDIX

Lemma 7. *(Chernoff Bound). Given a set of independent binary random variables $X_1, X_2, \cdots, X_n$, let $X = \sum_1^n X_i$ and $\mu = \sum_1^n p_i$, where $X_i = 1$ with probability p_i. If $\mathbb{E}\left[\prod_{i \in S} X_i \leq \prod_{i \in S} q_i\right]$, where $S \subseteq \{0, 1, \cdots, n\}$, then it holds for any $\delta > 0$ that*

$$P_r[X \geq (1 + \delta)\mu] \leq e^{-\frac{\delta^2 \mu}{2(1 + \delta/3b)}}.$$

If $\mathbb{E}\left[\prod_{i \in S} X_i \geq \prod_{i \in S} q_i\right]$, where $S \subseteq \{0, 1, \cdots, n\}$, then for any $\delta \in (0, 1]$, we have

$$P_r[X \leq (1 - \delta)\mu] \leq e^{-\frac{\delta^2 \mu}{2}}.$$

Lemma 8. *(Azuma–Hoeffding Inequality). Suppose X_k ($k = 0, 1, \cdots$) is a martingale (or super-martingale) and $|X_k - X_{k-1}| \leq c_k$. Then for all positive integers N and $\delta > 0$, we have*

$$P[X_N - X_0 > \delta] \leq e^{\frac{-\delta^2}{2 \sum_{k=1}^N c_k^2}}.$$

Lemma 9. *(Impacts of α, β in SINR on BLOWN's properties). The SINR model formulated by Eq. (1) has two coefficients, namely α (path-loss exponent) and β (noise threshold). We say that Theorem 5 holds for $\alpha \in (\alpha_{min}, \alpha_{max}), \beta \in (\beta_{min}, \beta_{max})$, which indicates that nodes can have various α and β.*

Proof. To prove this lemma, we need to show that with $\alpha \in (\alpha_{min}, \alpha_{max})$ and $\beta \in (\beta_{min}, \beta_{max})$, $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ satisfies persistence and liveness. This is equivalent to showing that with $\alpha \in (\alpha_{min}, \alpha_{max})$ and $\beta \in (\beta_{min}, \beta_{max})$, $\pi_B[\mathcal{F}_{SIG}, \mathcal{F}_{SORT}]$ satisfies the chain growth, common prefix, and chain quality properties. In Section 5.2, the chain growth property is the first one to be proved and it affects the correctness of the other two properties. Also, the distributions of α and β only affect the chain quality property. Thus we need to show that Lemma 2, 3, 4, and 5 hold for $\alpha \in (\alpha_{min}, \alpha_{max})$ and $\beta \in (\beta_{min}, \beta_{max})$. First of all, it is straightforward to verify that Lemma 2 and 3 hold for any constant $\alpha \in \{\alpha_{min}, \alpha_{max}\}$. For Lemma 4, α and β only affect the length of the subframe I_k. However, for each value of $\alpha \in (\alpha_{min}, \alpha_{max})$ and $\beta \in (\beta_{min}, \beta_{max})$, we can adapt $\gamma = O(1/(\log T + \log \log N)$ to make Eq. (12) hold. Therefore, Lemma 4 still holds in this case. If Lemma 3 holds, Lemma 5 holds, which completes the proof. $\square$