ZK-BFT: A Zero-knowledge and Byzantine Fault Tolerant Consensus for Permissioned Blockchain Networks

Wanxin Li
wanxin.li@xjtlu.edu.cn
Xi'an Jiaotong-Liverpool University
China

Collin Meese
Mark Nejad
cmeese@udel.edu
nejad@udel.edu
University of Delaware
USA

Hao Guo
haoguo@nwpu.edu.cn
Northwestern Polytechnical University
China

ABSTRACT

Consensus algorithms play an essential role in blockchains, directly impacting their performance. These algorithms involve validating and ordering pending transactions into new blocks, a process that exposes data to consensus nodes, raising privacy concerns. While existing consensus algorithms focus on network security and performance, data privacy within the consensus layer has received limited attention. This study introduces ZK-BFT, a zero-knowledge and Byzantine fault-tolerant consensus algorithm for permissioned blockchains. ZK-BFT verifies transactions without disclosing original information to consensus nodes, enhancing data privacy within the consensus layer while maintaining Byzantine fault tolerance. Our experiments, using the Hyperledger Ursa cryptographic library and Hyperledger Fabric permissioned blockchain, demonstrate ZK-BFT's potential for integration into existing permissioned blockchain systems that require privacy-by-design and Byzantine fault tolerance.

CCS CONCEPTS

• **Networks** → **Network protocols**; • **Security and privacy** → **Network security**; **Cryptography**.

KEYWORDS

Blockchain, Byzantine Fault Tolerance, Consensus, Privacy, Zero-knowledge Proof

ACM Reference Format:
Wanxin Li, Collin Meese, Mark Nejad, and Hao Guo. 2023. ZK-BFT: A Zero-knowledge and Byzantine Fault Tolerant Consensus for Permissioned Blockchain Networks. In *2023 6th International Conference on Blockchain Technology and Applications (ICBTA) (ICBTA 2023), December 15–17, 2023, Xi'an, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3651655.3651663

1 INTRODUCTION

A permissioned blockchain is a decentralized network where access and data management are restricted to a predefined group of participants or entities, ensuring controlled and secure transactions. The consensus algorithm is a key component of blockchain systems, and the dispersed blockchain nodes utilize it to reach agreements on the order of transactions to be included in the next block. The consensus process ensures the security of the blockchain and the integrity of the associated data by validating all new transactions before they are committed to the global ledger and replicated on all participating nodes. Additionally, consensus algorithms largely impact the overall performance of a blockchain system such as transaction throughput, latency and fault tolerance [1].

While many consensus algorithms prioritize application security and performance, addressing data privacy within the blockchain's consensus layer remains underdeveloped. During consensus processing, nodes validate and order transactions, necessitating the verification of application-specific data within transactions. This exposes data to consensus nodes, raising privacy concerns, particularly in sensitive sectors like healthcare [9, 10] and transportation [11, 14, 15, 17] subject to regional privacy regulations. This shortfall could impede the practicality of deploying valuable blockchain systems. Therefore, blockchain technology and its applications stand to gain significantly from a consensus algorithm that inherently safeguards data privacy within the consensus layer.

Nevertheless, some permissioned blockchain platforms have attempted to solve the privacy problem outside of the consensus layer in creative ways. For example, in the popular Hyperledger Fabric permissioned blockchain platform [1], privacy is primarily preserved either at the channel level through membership and chaincode access permissions, or by using off-chain storage for sensitive data [5]. In the first case, application-specific channels for transactions and smart contracts are deployed and maintained by the governing consortium, and permission to view and transact on a given channel is granted or revoked by the consortium members. However, in this approach the underlying data encapsulated within the blockchain transactions is not protected, and the consensus nodes must have access to it in order to verify the transaction details. In the second case, off-chain storage approaches can be leveraged to preserve the data privacy, but still do not solve the underlying problem of privacy-preservation directly within the consensus process. This motivates our research into the proposed ZK-BFT consensus, which integrates privacy-preserving directly into the consensus layer while still enabling fast processing and

[1]https://www.hyperledger.org/projects/fabric

Byzantine fault tolerance, expanding the potential use-cases for permissioned blockchain technology.

In this study, we address the issue of data privacy within the blockchain consensus layer by proposing a novel privacy-preserving and Byzantine fault tolerant consensus, ZK-BFT, designed for use in permissioned blockchains. The proposed ZK-BFT consensus algorithm integrates zero-knowledge proof (ZKP) for blockchain data directly into the consensus processing, ensuring data privacy while still providing fast performance and Byzantine fault tolerance. Until now, previous works [2, 3, 6] have focused on providing privacy to blockchain data either outside of the consensus process or being applied in the context of permissionless blockchains for cryptocurrency trading, which have different trust and identity assumptions than their permissioned counterparts. To the best of our knowledge, this is the first work to address zero-knowledge privacy within the BFT consensus layer of permissioned blockchains. To summarize, this paper makes the following contributions:

- We propose the ZK-BFT consensus that brings both zero-knowledge privacy and Byzantine fault tolerance to the consensus layer of permissioned blockchain systems.
- We present ZKP-based algorithms and a view change protocol that provide built-in privacy and liveness for ZK-BFT consensus.
- We analyze the properties of ZK-BFT including the correctness proof, privacy-preserving, fault tolerance and communication complexity, and compare its performance with other typical consensus algorithms.
- We conduct experiments to show the performance of ZK-BFT using Hyperledger Ursa cryptographic library and Hyperledger Fabric blockchain platform.

2 RELATED WORK

Zero-knowledge proofs, initially proposed by Goldwasser et al. [8], allow a prover to convince a verifier of their knowledge of a secret message m while revealing nothing else except their possession of m. These proofs excel in demonstrating possession of a secret without revealing any additional information about it. Research has led to two main types of ZKP protocols: interactive and non-interactive. Interactive ZKP requires a dialogue between the prover and verifier, with multiple challenges for the prover to address, ensuring the verifier's conviction in the prover's knowledge. Non-interactive ZKP allows the prover to generate a proof that can be independently verified, reducing the communication overhead of the proof system.

The inherent properties of ZKP offer a means of ensuring transaction input validity in blockchain systems while safeguarding sensitive information. Notably, ZKP protocols have been introduced to enhance data privacy within cryptocurrency systems. For instance, zk-SNARK [3] was the pioneering ZKP protocol integrated into Zcash [2] cryptocurrency in 2012, ensuring privacy for financial transactions on a PoW-based ledger. Subsequently, Bulletproofs, introduced in 2018 and implemented in Monero cryptocurrency [3], enables efficient proof of a committed value within a range [6]. In Ethereum blockchain [4], zk-STARK [2] is explored for transaction data privacy without relying on a trusted setup, offering enhanced privacy and reduced trust assumptions compared to zk-SNARK.

However, the existing zero-knowledge protocols have either been applied to permissionless blockchain networks to protect the privacy of cryptocurrency transactions [3, 6], or are implemented at the smart contract layer outside of the consensus processing [2]. This motivated our research into the proposed ZK-BFT consensus, which inherently preserves data privacy from consensus layer while also offering Byzantine fault tolerance, designed specifically for use in permissioned blockchain systems.

3 ZK-BFT CONSTRUCTION

3.1 Consensus Overview

There are four entities including the certificate authority and three types of nodes participating in the ZK-BFT consensus. The proposed consensus also consists of ZKP-based algorithms that generates and verifies transaction requests without revealing the message, and a view change protocol that guarantees the network liveness. We define the following entities that take part in the proposed consensus:

- Certificate Authority (CA): The CA verifies clients' digital asset ownership by issuing key pairs. Clients receive private keys for generating zero-knowledge proofs, while public keys are shared with the primary node and replica nodes for validation.
- Client Node: Client nodes handle the generation of zero-knowledge proofs and the transmission of transaction requests.
- Primary Node: The primary node, functioning as a healthy leader, is in charge of receiving and forwarding transaction requests, constructing and publishing blocks, as well as verifying and voting on transactions alongside replica nodes. Each consensus process involves a single primary node.
- Replica Node: Replica nodes are responsible for validating transactions and assessing the leader's status for voting.

In our design, consensus nodes include both primary node and replica nodes. The certificate authority issues private keys to clients for generating zero-knowledge proofs and public keys to consensus nodes for verifying the proofs. Transaction requests are generated by clients in ZKP format that hide the original information. Once receiving the transaction request from a client, the primary node forwards it, and all consensus nodes will verify and confirm the transaction request. The consensus is technically reached when the client receives at least $(N-1)/3+1$ replies from a total number of N consensus nodes. Next, we present the detailed ZK-BFT consensus construction including the consensus processing and the view change protocol. Table 1 shows the variables we use in constructing the proposed consensus algorithm.

3.2 Consensus Processing

Referring to Fig. 1, the ZK-BFT consensus algorithm comprises the following steps: the certificate authority issues key pairs to clients and consensus nodes; the client sends a request; the primary

[2] https://z.cash/
[3] https://www.getmonero.org/
[4] https://ethereum.org/en/

Table 1: Variables in ZK-BFT Consensus Algorithm

Variable	Description
m	Secret message in the transaction (e.g., digital assets)
sk	Private key for generating the zero-knowledge proof
pk	Public key for verifying the zero-knowledge proof
$\mathbb{G}$	Multiplicative cyclic group of prime order p
g	Generator in $\mathbb{G}$
h	Hash digest based on the secret message m
δ	Generated one-time zero-knowledge proof
e	Elliptic curve for bilinear pairing
r	Result for verifying the zero-knowledge proof (true or false)
f	Number of faculty consensus nodes
N	Total number of consensus nodes
id	Identifier for the secret message m
v	View number (period that a given node is the primary)
i	Identifier for the replica node
c	Transaction confirmation result

node forwards the request; all consensus nodes perform *Verify* and *Confirm* processes; the client receives a response confirming consensus. These steps constitute the core of the ZK-BFT consensus algorithm. Consensus is reached when the client obtains at least $f + 1$ replies (f represents the number of faulty nodes) from nodes, validating the final consensus among a total of $3f + 1$ replicas. The proposed consensus process is detailed as follows:

3.2.1 *Setup.* Inherent to a zero-knowledge proof scheme is the necessity for the verifier to precisely understand what is being proven to prevent the prover from producing fraudulent proofs. In this procedure, the certificate authority employs Algorithm 1 to generate key pairs for the certified digital assets of clients. The *Setup* process is a one-time operation, and the certificate authority dispatches the setup message $< SETUP, id, sk, pk >$ to the client node, along with setup messages $< SETUP', id, pk >$ to the consensus nodes. Here, id serves as an identifier to specify the original message m (e.g., digital asset), sk signifies the private key, and pk represents the public key.

$GenerateKey(m) \longrightarrow (sk, pk)$: This algorithm chooses a random $sk \in \mathbb{Z}_p$ and computes $pk = g^{sk} \in \mathbb{G}$. Return the private key sk and the public key pk for the message m.

3.2.2 *Request.* During this procedure, the client employs Algorithm 2 to create a one-time zero-knowledge proof δ using the original message m. Subsequently, the client initiates the transmission of a request $< REQUEST, id, h, \delta >$ to the system, with h representing the hash digest of the original message m.

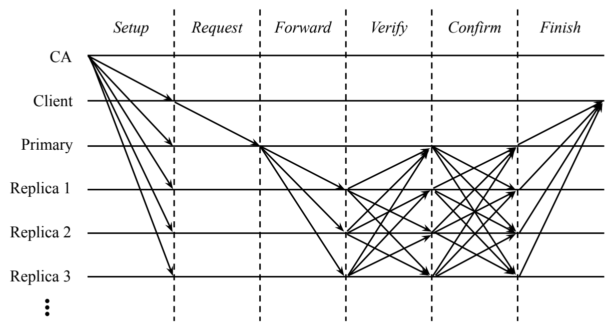

Figure 1: The proposed ZK-BFT consensus processing.

Algorithm 1: *GenerateKey*

```
Input  : secret message $m$
Output : private key $sk$, public key $pk$
1 The certificate authority selects a random $a \in \mathbb{Z}_p$ for
    message $m$;
2 The certificate authority saves the private key as $sk = a$;
3 The certificate authority computes the public key as
    $pk = g^{sk} \in \mathbb{G}$;
4 The certificate authority returns $sk$ and $pk$.
```

Algorithm 2: *GenerateProof*

```
Input  : secret message $m$, private key $sk$
Output : one-time zero-knowledge proof $\delta$
1 The client computes a hash digest $h$ based on the secret
    message $m$, as $h = H(m)$;
2 The client generates the one-time zero-knowledge proof
    $\delta = h^{sk} \in \mathbb{G}$;
3 The client returns $\delta$.
```

$GenerateProof(m, sk) \longrightarrow (\delta)$: This algorithm first computes a one-time hashed messages m, in SHA256 algorithm [19], as $h = H(m)$. Then, it computes a one-time $\delta = h^{sk} \in \mathbb{G}$ and returns the proof δ. Elliptic curves usually have about 2^{256} points [4], and SHA-256 hashing algorithm can offer a 256-bit result.

3.2.3 *Forward.* The primary node publishes a new block and broadcasts the client's request in messages $< FORWARD, id, h, \delta, v >$ to the other replica nodes. The v represents the view number.

3.2.4 *Verify.* Replica nodes receive the forwarded messages and perform the following verifications: (1) The node is currently in view v; (2) The node does not possess other *Forward* messages on the same page (view v, message identifier id). In other words, there is no additional set of (h', δ') that shares the same message identifier id with the set of (h, δ) in the present view v; (3) The node executes Algorithm 3 to validate the one-time zero-knowledge proof δ without accessing the original message m. Following the successful verification, replica nodes dispatch the corresponding verification messages $< VERIFY, id, h, \delta, v, i, r >$ to the other consensus nodes. Here, i denotes the identity of the replica node, and r is a Boolean value ($true$ or $false$) that indicates the verification result.

$VerifyProof(\delta, pk, g, h) \longrightarrow (true/false)$: This algorithm takes the proof δ, the public key pk, the generator g and the hashed identity information h as input and check if $e(\delta, g) = e(h, pk)$. Finally, the function returns a Boolean value either true or false to validate the proof δ without compromising the information of the message m.

3.2.5 *Confirm.* Each consensus node needs to receive at least $2f$ verification messages from other consensus nodes (a total of

Algorithm 3: *VerifyProof*

```
Input  : one-time zero-knowledge proof $\delta$, public key $pk$,
         generation $g$, hashed secret message $h$
Output : verification result $r$
1 The consensus node checks if $e(\delta, g) == e(h, pk)$ then
2 |   $r = true$ ;
3 else
4 |   $r = false$ ;
5 end
6 The consensus node returns $r$.
```

$2f + 1$ including its own) and validates if the id, h, δ, v of these verification messages are all consistent. Once validation is completed, the consensus node will set the confirmation message $< CONFIRM, id, h, \delta, v, i, c >$ to true, and broadcast it to the other consensus nodes. The c is a Boolean value that indicates its confirmation result.

3.2.6 Finish. Once in the *Finish* phase, each node commits the block for which they have the corresponding *Forward*, $2f + 1$ *Verify*, and $2f + 1$ *Confirm* messages. After the block is successfully committed to the chain, each node sends a confirmation message to the client to indicate consensus has been achieved for its request.

In the event the client fails to collect $f+1$ *Finish* messages within a time period t, the client resends the request to the primary node for a retry. Upon receiving the same request, if consensus has already been reached during the *Confirm* phase, replicas retransmit the *Finish* messages. If consensus has not been reached, the network restarts the protocol.

3.3 View Change Protocol

A view represents the duration during which a specific node acts as the primary. Consequently, a view change entails transitioning to a different primary node. Our proposed ZK-BFT consensus ensures network liveness through a view change protocol, as illustrated in Fig. 2. When a replica node (e.g., replica node 1) identifies a fault in the current view v, such as the primary node sending an invalid message or failing to produce a valid block in a timely manner, it broadcasts a view change request for $v + 1$ to the other nodes in the network. Upon receiving the request, the other nodes validate it through communication with the current primary node. If indeed the primary is found to be faulty, all non-faulty nodes broadcast confirmation messages for the view change.

When replica node 1 receives $2f$ confirmation messages from other nodes (a total of $2f + 1$ including its own), it will broadcast a new view message for view $v + 1$ to all nodes including the client node. When the other nodes receive the new view message, they will switch to the new view, and the new primary will start publishing blocks, receiving and forwarding clients' request messages.

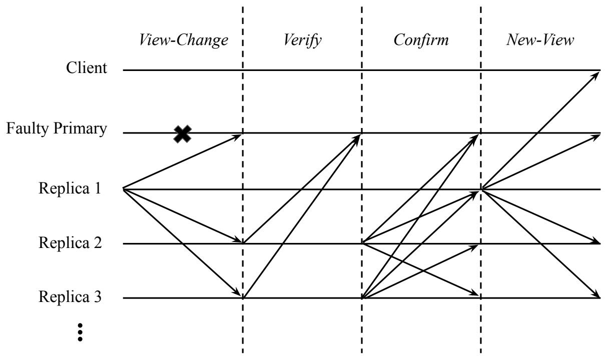

Figure 2: View change protocol in ZK-BFT. (The primary node is faulty and the replica node 1 will become the new primary node.)

4 THEORETICAL ANALYSIS

In this section, we first give a formal proof under the proposed ZK-BFT consensus design. Next, we discuss how privacy-preserving and Byzantine fault tolerance are addressed in ZK-BFT consensus. We then analyze the performance of ZK-BFT consensus from the perspective of communication complexity, and compare ZK-BFT with other state-of-the-art consensus algorithms.

4.1 Correctness Proof

Proposition 1. *The proposed ZK-BFT consensus can accurately validate transaction requests without disclosing the original message m.*

Assuming a client generates the zero-knowledge proof δ for the message m and transmits this proof δ along with the transaction request to the primary node. We demonstrate the validation of proof δ through Algorithm 3. Initially, the public key pk is computed as follows:

$$pk = g^{sk}, \tag{1}$$

Then, a one-time zero-knowledge proof is generated as:

$$(m, sk) \longrightarrow \delta = H(m)^{sk} = h^{sk}, \tag{2}$$

Next, verifying the proof δ is done by checking that if:

$$e(\delta, g) = e(h, pk), \tag{3}$$

Now, we prove the Equation 3 based on bilinear pairing property:

$$\begin{aligned} e(\delta, g) &= e(h^{sk}, g) \\ &= e(h, g^{sk}) \\ &= e(h, pk). \end{aligned} \tag{4}$$

Bilinear Pairing Property: *Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p with generator g. Let $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ be a computable, bilinear and non-degenerate pairing into the group $\mathbb{G}_T$. Then, we have $e(x^a, y^b) = e(x, y)^{ab}$ for all $x, y \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$ because $\mathbb{G}$ is cyclic.*

4.2 Privacy-preserving

Proposition 2. *The proposed ZK-BFT consensus can protect transaction privacy during consensus processing.*

As shown in Equation 2, a one-time zero-knowledge proof δ is generated based on the hashed message m and private key sk. Instead of sending the original message m to consensus processing, peer nodes in ZK-BFT will use Equation 3 to validate the proof δ and reach into consensus. In this way, a transaction will be processed without revealing any details about the original message m to peer nodes. If an attacker intercepts the zero-knowledge proof δ, it still cannot decrypt the original message m. This is because the hash digest $H(m)$ is one-way generated using the SHA-256 algorithm, and as a result, the generated proof δ from Equation 2 is a unique and one-time proof that cannot be reversed.

4.3 Crash Fault Tolerance

Proposition 3. *The proposed ZK-BFT consensus can tolerate up to f nodes that fail for communication ($f = (N-1)/3$). In other words, ZK-BFT consensus can offer $1/3$ crash fault tolerance.*

Crash fault tolerance (CFT) is one level of resiliency, where the distributed system can still correctly reach consensus if certain nodes fail for communication. Given a total number of N consensus nodes ($N = 3f+1$), each node needs to receive at least $2f$ verification messages from other nodes to successfully process the *Confirm* phase resulting in $(N-1)/3$ crash fault tolerance.

4.4 Byzantine Fault Tolerance

Proposition 4. *The proposed ZK-BFT consensus can tolerate up to f nodes that send malicious messages ($f = (N-1)/3$). In other words, ZK-BFT consensus can offer $1/3$ Byzantine fault tolerance.*

Byzantine fault tolerance (BFT) builds a more complex level of resiliency and deals with distributed systems that certain nodes could have malicious actors. BFT is enabled during ZK-BFT consensus processing. Given a total number of N consensus nodes ($N = 3f+1$), the ZK-BFT can still reach a consensus when a maximum number of f nodes send malicious messages. We will prove this under the worst circumstance.

First, in the *Verify* phase, we divide the non-fault nodes into three groups: (1) f non-fault nodes with decision A; (2) f non-fault nodes with decision B; and (3) one remaining non-fault node. Then f malicious nodes tell the first non-fault node group that they support decision A, and tell the second non-fault node group that they support decision B. Now, in the *Confirm* phase, decision A has gathered $2f$ votes from the first group's point of view, and decision B has gathered $2f$ votes from the second group's point of view. Then after the remaining one non-fault node broadcasts its decision, consensus will be reached: one of the groups becomes the majority and the other becomes the minority. In conclusion, the ZK-BFT consensus can offer $(N-1)/3$ Byzantine fault tolerance.

4.5 Communication Complexity

Proposition 5. *For a total node count of N, comprising one primary node and n replica nodes ($N = n+1$), the communication complexity for achieving consensus in the proposed ZK-BFT is $O(N^2)$.*

The communication complexity of ZK-BFT is on the order of N^2 due to the peer-to-peer and all-to-all communications during the *Verify* and *Confirm* phases, as illustrated in Fig. 1. To elaborate, in the *Verify* phase, each replica node sends its verification results to all other consensus nodes, including the primary node, resulting in a total of $(N-1)\cdot N$ messages. In the *Confirm* phase, every consensus node, including the primary node, checks the consistency of verification results from other nodes and broadcasts confirmation messages to all other consensus nodes, generating a total of $N \cdot N$ messages.

4.6 Comparison Analysis

In this subsection, we compare the properties and performance among the proposed ZK-BFT and other typical consensus algorithms for blockchain networks. To the best of our knowledge, the proposed ZK-BFT and its previous version P-CFT [16] are the first two mechanisms that have built-in zero-knowledge privacy from blockchain consensus layer. In comparison with our previous P-CFT study, the ZK-BFT consensus is re-designed and further improves the fault tolerance level against Byzantine attacks.

Proof of Work (PoW) [12] and Proof of Stake (PoS) [20] are two commonly used consensus algorithms in permissionless blockchain. In PoW, miners consume huge amounts of computing power to compete with each other, add blocks of transactions to the blockchain, and get rewarded with more coins. As a result, PoW can only process transactions in low throughput and high latency, whose communication complexity is $O(N^2)$. The communication complexity of PoS is the same as PoW. However, the performance was slightly improved in PoS because mining power is given based on the amount of coins held by a miner. Both PoW and PoS can offer $(N-1)/2$ Byzantine fault tolerance, but at the cost of greatly increased resource consumption and reduced throughput in comparison with permissioned blockchain consensus algorithms.

In permissioned blockchain, consensus algorithms are designed to be computationally inexpensive, and performance are distinctly improved because only a smaller group of nodes are needed for validation purposes. In Raft consensus [18], every node in the replicated state machine can participate in any three states: follower, candidate, and leader. The leader is responsible for log replication to the followers, and a candidate can be elected as the new leader when the current leader fails for communication. Raft can provide higher transaction throughput and lower transaction latency because all followers trust the leader node and the communication complexity of Raft is only $O(N)$. However, Raft can not tolerate Byzantine failure when some nodes send malicious information during consensus processing.

In terms of Byzantine fault tolerance for permissioned blockchains, these BFT-type consensus algorithms, such as pBFT [7] and Tendermint [13], are able to deal with Byzantine nodes that behave arbitrarily. In pBFT, Byzantine fault tolerance is achieved by requiring multiple rounds of voting by the set of verifiers to arrive at a mutual agreement, which is recorded as a collection of signatures on the block content. Tendermint is an extension of original pBFT with optimised gossip-based communication and designed for high number of nodes. As we analyze above, our proposed ZK-BFT consensus can offer the same Byzantine fault tolerant rate and communication complexity with pBFT and Tendermint. Moreover, the proposed ZK-BFT can verify transactions without revealing details due to the zero-knowledge algorithm design. In Table 2, we provide a summary of the comparisons for the consensuses discussed in this subsection.

Table 2: Comparisons of the proposed and other typical consensus algorithms.

Consensus	Blockchain Type	Zero-knowledge	Byzantine Fault Tolerance	Communication Complexity	Throughput	Latency
PoW [12]	Permissionless	×	✓, $(N-1)/2$	$O(N^2)$	low	high
PoS [20]	Permissionless	×	✓, $(N-1)/2$	$O(N^2)$	medium	medium
Raft [18]	Permissioned	×	×	$O(N)$	high	low
pBFT [7]	Permssioned	×	✓, $(N-1)/3$	$O(N^2)$	high	low
Tendermint [13]	Permssioned	×	✓, $(N-1)/3$	$O(N^2)$	high	low
P-CFT [16]	Permissioned	✓	×	$O(N^2)$	high	low
ZK-BFT (proposed)	Permissioned	✓	✓, $(N-1)/3$	$O(N^2)$	high	low

5 EXPERIMENTS AND EVALUATION

5.1 Experiment Setup

The zero-knowledge proof is the core component of the ZK-BFT that brings privacy-preserving into consensus processing. We first present the implementation of the proposed zero-knowledge proof algorithms, which are written in Rust programming language utilizing the Hyperledger Ursa library. Then, we conduct a series of experiments to measure the performance of the ZK-BFT consensus algorithm. We also compare ZK-BFT with other existing zero-knowledge protocols that provide privacy for blockchain. The implementation and experiments are deployed on Ubuntu 18.04 operating system with 2.8 GHz Intel i5-8400 processor and 32GB DDR4 memory. The initial consensus group consists of 20 replica nodes, and we increase the nodes group from 20, to 40, 60, 80 and 100 in experiments.

5.2 Performance

In this subsection, we present the experimental results that measure the performance of the proposed ZK-BFT consensus. The results include transaction size, transaction generation time, transaction verification time and transaction latency by varying the number of transactions, and message counts from consensus processing and view change operation by varying the number of replica nodes in the network:

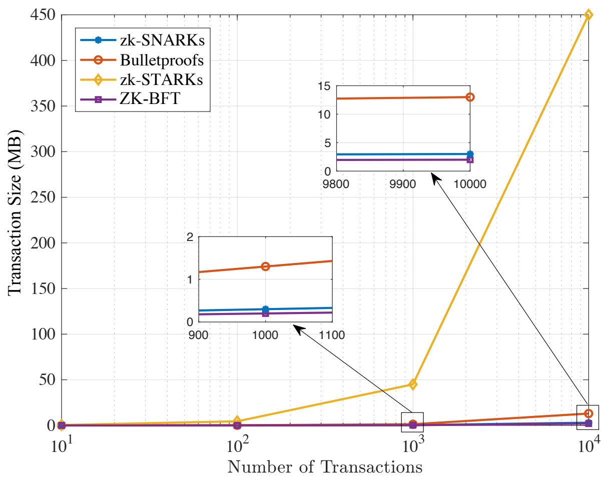

Figure 3: The relationship between total transaction size and number of transactions from the proposed ZK-BFT and other state-of-the-art ZKP protocols.

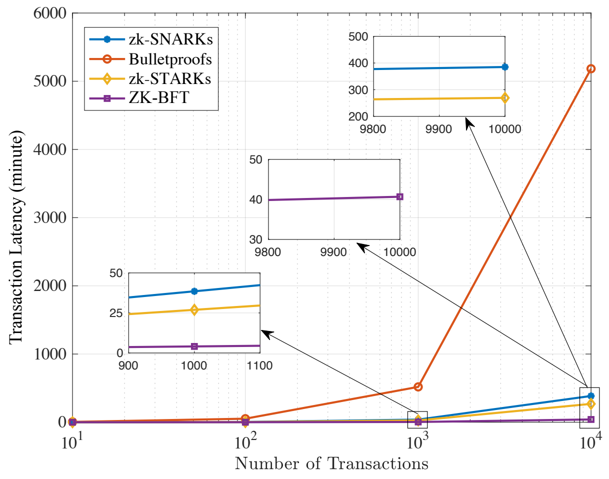

Figure 4: The relationship between transaction latency and number of transactions from the proposed ZK-BFT and other state-of-the-art ZKP protocols.

5.2.1 *Transaction Size.*

Transaction size is an important consideration for blockchain networks. Smaller transactions are easier to validate; larger transactions take more work, and take up more space in the block. To evaluate the transaction size of the proposed ZK-BFT consensus, we conduct multiple groups of experiments with the number transactions varying in this section. As shown in Fig. 3, ZK-BFT takes only 2MB for processing 10,000 transactions in the network , which is more efficient than zk-SNARKs (3MB) and saves tens of megabytes compared to Bulletproof and hundreds of megabytes compared to zk-STARKs.

5.2.2 *Transaction Latency.*

By considering the transaction generation and verification time together, Fig. 4 displays the total transaction latency by varying the number of transactions. The proposed ZKP takes only 40 minutes to process 10,000 transactions in ZK-BFT consensus, which saves hundreds of minutes from zk-SNARKs and zk-STARKs and even thousands of minutes from Bulletproofs.

5.2.3 *Number of Messages.*

In ZK-BFT, consensus nodes reach into agreements by rounds of communication. Therefore, the number of consensus nodes in the network will impact the number of exchanged messages during consensus processing and view change operation. We increase the number of replica nodes from 20, to 40, 60, 80 and 100, and record the number of exchanged messages for nodes communication. By considering varying the number of replica nodes in the network, Fig. 5 shows the number of exchanged messages from each step during ZK-BFT consensus processing, and Fig. 6 shows the number exchanged messages from each step in ZK-BFT view change operation.

During ZK-BFT consensus processing, the primary node is the only one representative from the network to receive the transaction request from the client. Consequently, there is only one message sent at the *Request* step even we increase the number of replica nodes. In *Setup*, *Forward* and *Finish* steps, the number of exchanged messages will increase linearly. The $Verify$ and $Confirm$ messages will have exponential growth due to the N-to-N communications.

In ZK-BFT view change operation, the $Verify$ messages becomes linear growth because each replica node only needs to communicate once with the current faulty primary node. The $View-Change$ request and $New - View$ result messages show linear growth because of the one-to-N communications by the future new primary node. The $Confirm$ messages will stay exponential growth due to the N-to-N communications to reach into an agreement on the new primary node.

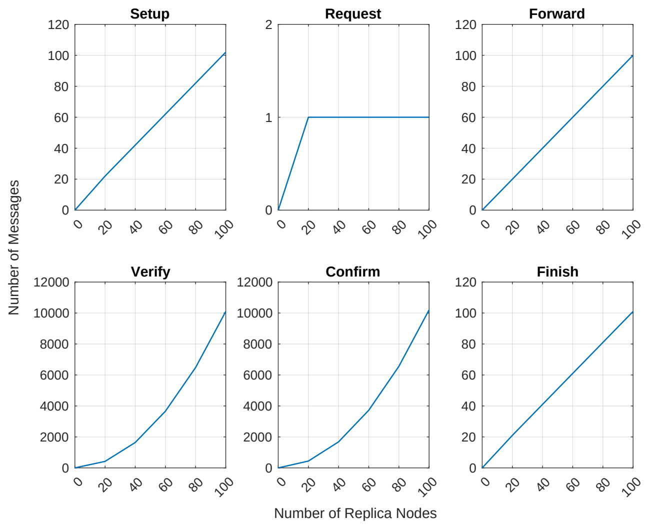

Figure 5: The relationship between number of exchanged messages and number of replica nodes from each step in ZK-BFT consensus processing.

6 CONCLUSION

This paper proposes ZK-BFT, a zero-knowledge and Byzantine fault tolerant consensus algorithm for permissioned blockchains. The ZKP-based algorithms and the view change protocol are proposed to support the operation of ZK-BFT and provide its privacy-preservation feature. In theoretical analysis of the proposed ZK-BFT consensus, we discussed its correctness proof, privacy-preserving feature, fault tolerance threshold, communication complexity, and compare ZK-BFT with other typical consensus algorithms. We conduct experiments to measure the performance of ZK-BFT, and the results show that ZK-BFT can provide lower transaction latency with smaller transaction size in comparison with the existing ZKP protocols.

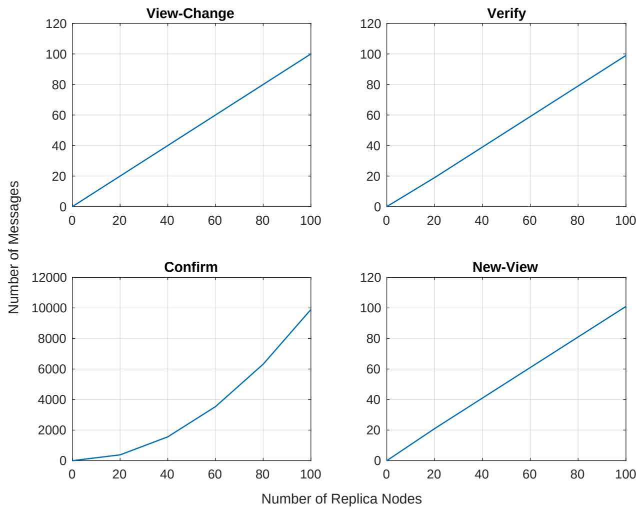

Figure 6: The relationship between number of exchanged messages and number of replica nodes from each step in ZK-BFT view change protocol.

ACKNOWLEDGMENTS

This manuscript is a revised and extended version of "P-CFT: A Privacy-preserving and Crash Fault Tolerant Consensus Algorithm for Permissioned Blockchains" [16], which was published in the proceeding of 2021 4th IEEE International Conference on Hot Information-Centric Networking (HotICN 2021). The research is supported by Xi'an Jiaotong-Liverpool University Research Development Fund under the Grant No. RDF-22-02-106, 2023-2026, the Fundamental Research Funds for the Central Universities under the Grant No. G2021KY05101, 2021-2024, Guangdong Basic and Applied Basic Research Foundation under the Grant No. 2021A1515110286, 2021-2024, and the Basic Research Programs of Taicang under Grant No. TC2022JC23, 2022-2024.

REFERENCES

[1] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. 2020. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications* 154 (2020), 113385.
[2] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2018. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.* 2018 (2018), 46.
[3] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. 2014. Succinct non-interactive zero knowledge for a von Neumann architecture. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 781–796.
[4] Joppe W Bos, J Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow. 2014. Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security*. Springer, 157–175.
[5] Sotirios Brotsis, Nicholas Kolokotronis, Konstantinos Limniotis, Gueltoum Bendiab, and Stavros Shiaeles. 2020. On the Security and Privacy of Hyperledger Fabric: Challenges and Open Issues. In *2020 IEEE World Congress on Services (SERVICES)*. 197–204. https://doi.org/10.1109/SERVICES48979.2020.00049
[6] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 315–334.

[7] Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.

[8] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1989. The knowledge complexity of interactive proof systems. *SIAM Journal on computing* 18, 1 (1989), 186–208.

[9] Hao Guo, Wanxin Li, Ehsan Meamari, Chien-Chung Shen, and Mark Nejad. 2020. Attribute-based Multi-Signature and Encryption for EHR Management: A Blockchain-based Solution. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 1–5. https://doi.org/10.1109/ICBC48266.2020.9169395

[10] Hao Guo, Wanxin Li, Mark Nejad, and Chien-Chung Shen. 2023. A Hybrid Blockchain-Edge Architecture for Electronic Health Record Management With Attribute-Based Cryptographic Mechanisms. *IEEE Transactions on Network and Service Management* 20, 2 (2023), 1759–1774. https://doi.org/10.1109/TNSM.2022.3186006

[11] Hao Guo, Collin Meese, Wanxin Li, Chien-Chung Shen, and Mark Nejad. 2023. B2SFL: A Bi-Level Blockchained Architecture for Secure Federated Learning-Based Traffic Prediction. *IEEE Transactions on Services Computing* (2023), 1–15. https://doi.org/10.1109/TSC.2023.3318990

[12] Markus Jakobsson and Ari Juels. 1999. Proofs of work and bread pudding protocols. In *Secure information networks*. Springer, 258–272.

[13] Jae Kwon. 2014. Tendermint: Consensus without mining. *Draft v. 0.6, fall* 1, 11 (2014).

[14] Wanxin Li, Collin Meese, Hao Guo, and Mark Nejad. 2020. Blockchain-Enabled Identity Verification for Safe Ridesharing Leveraging Zero-Knowledge Proof. In *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*. 18–24. https://doi.org/10.1109/HotICN50779.2020.9350858

[15] Wanxin Li, Collin Meese, Hao Guo, and Mark Nejad. 2023. Aggregated Zero-Knowledge Proof and Blockchain-Empowered Authentication for Autonomous Truck Platooning. *IEEE Transactions on Intelligent Transportation Systems* 24, 9 (2023), 9309–9323. https://doi.org/10.1109/TITS.2023.3271436

[16] Wanxin Li, Collin Meese, Mark Nejad, and Hao Guo. 2021. P-CFT: A Privacy-preserving and Crash Fault Tolerant Consensus Algorithm for Permissioned Blockchains. In *2021 4th International Conference on Hot Information-Centric Networking (HotICN)*. 26–31. https://doi.org/10.1109/HotICN53262.2021.9680829

[17] Wanxin Li, Collin Meese, Zijia Gary Zhong, Hao Guo, and Mark Nejad. 2021. Location-aware Verification for Autonomous Truck Platooning Based on Blockchain and Zero-knowledge Proof. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 1–5. https://doi.org/10.1109/ICBC51069.2021.9461116

[18] Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*. 305–319.

[19] D Rachmawati, JT Tarigan, and ABC Ginting. 2018. A comparative study of Message Digest 5 (MD5) and SHA256 algorithm. In *Journal of Physics: Conference Series*, Vol. 978. 012116.

[20] Fahad Saleh. 2021. Blockchain without waste: Proof-of-stake. *The Review of financial studies* 34, 3 (2021), 1156–1190.