Reputation-based state machine replication

Muhong Huang*, Runchao Han†, Zhiqiang Du*‡, Yanfang Fu*, and Liangxin Liu*
*Xi'an Technological University, China
{muhonghuang, duzhiqiang, fuyanfang, liuliangxin}@xatu.edu.cn
†Monash University and CSIRO's Data61, Australia
me@runchao.rocks

***Abstract*—State machine replication (SMR) allows nodes to jointly maintain a consistent ledger, even when a part of nodes are Byzantine. To defend against and/or limit the impact of attacks launched by Byzantine nodes, there have been proposals that combine reputation mechanisms to SMR, where each node has a reputation value based on its historical behaviours, and the node's voting power will be proportional to its reputation. Despite the promising features of reputation-based SMR, existing studies do not provide formal treatment on the reputation mechanism on SMR protocols, including the types of behaviours affecting the reputation, the security properties of the reputation mechanism, or the extra security properties of SMR using reputation mechanisms.**

In this paper, we provide the first formal study on the reputation-based SMR. We define the security properties of the reputation mechanism w.r.t. these misbehaviours. Based on the formalisation of the reputation mechanism, we formally define the reputation-based SMR, and identify a new property *reputation-consistency* that is necessary for ensuring reputation-based SMR's safety. We then design a simple reputation mechanism that achieves all security properties in our formal model. To demonstrate the practicality, we combine our reputation mechanism to the Sync-HotStuff SMR protocol, yielding a simple and efficient reputation-based SMR at the cost of only an extra Δ in latency, where Δ is the maximum delay in synchronous networks.

***Index Terms*—state machine replication, reputation, blockchain.**

I. Introduction

State machine replication (SMR) is a family of protocols that allow a set of nodes to jointly maintain a consistent ledger, even when a certain fraction of nodes are Byzantine. SMR has shown promise in realising many decentralised and fault-tolerant systems, especially blockchains [24], [28].

Despite its wide adoption, there have been many known attacks on SMR protocols, such as *equivocation* where the adversary votes on multiple conflicting blocks, *withholding* where the adversary withholds votes or blocks, and *flash attacks* [11] where the adversary enrols a large number of its nodes to the system to exceed the fault tolerance capacity temporarily. When the adversary does not exceed the protocol's fault tolerance capacity, these attacks can lead to worse communication complexity and/or latency. When the adversary exceeds the protocol's fault tolerance capacity, these attacks can break the protocol's safety and/or liveness.

‡ This research work was partially supported by the Shaanxi Natural Science Basic Research Project (Grant No.2020JM-565), the Shaanxi International Science and Technology Cooperation Program Project (Grant No.2021KW-07). Zhiqiang Du is the corresponding author of this paper.

To defend against and limit the impact of these attacks, many proposals [10], [20], [29] introduce the reputation mechanism to SMR protocols. In the real world, reputation mechanisms are used for increasing and decreasing the weight in making certain decisions for honest and deviating participants, respectively. It has been adopted in many real-world applications, such as E-commerce [1] and user-generated content [3], where people will prefer merchants and online contents with high reputation. When adapted to SMR protocols, the reputation mechanism accumulates nodes' reputations when they follow the protocol honestly, and slashes nodes' reputations when they deviate from the protocol. The voting process in reputation-based SMR protocols is then weighted by nodes' reputations, making the adversary harder to launch attacks. Proof-of-stake (PoS)-based blockchains [26] partially follow the reputation approach, where a node's stake determines its voting power and can be slashed if it behaves maliciously.

A. Related work

Despite the promising features of reputation-based SMR, existing proposals do not provide a formal treatment on the proposed reputation mechanisms, and reputation mechanisms for other protocols and applications cannot be directly adapted to SMR due to the different design goals.

Reputation-based SMR. Existing reputation-based SMR proposals focus on the SMR design, but neglect the formalisation and design of the reputation mechanism. Specifically, they informally discuss some behaviours that increase or reduce the reputation, omitting a formal and complete treatment of these behaviours and the formal security properties of the reputation mechanism.

Guru [10] proposes a reputation mechanism for SMR, where the protocol assigns nodes with reputations based on their behaviours, and elects a subset of nodes (called committee) based on their reputations to execute consensus. RepuCoin [29] is another reputation-based SMR proposal focusing on scalability. Compared to Guru which is built upon traditional BFT-style consensus, RepuCoin is built upon ByzCoin [21] where nodes elect leaders and committees by solving proof-of-work (PoW) puzzles, and the elected committee votes to finalise the proposed blocks. RepuCoin accumulates nodes' reputations if they honestly proposing/voting blocks, and neglects the design of slashing reputations upon misbehaviours. The authors also note that the reputation mechanism can provide resistance against *flash attacks* [11] where the adversary spawns a large

number of new nodes in the system to take over the consensus. Kleinrock et al. [20] combines the reputation mechanism to proof-of-stake (PoS)-based Nakamoto blockchains, where nodes elect committees based on nodes' reputations to finalise blocks, and fall back to the vanilla PoS-based Nakamoto consensus under liveness attacks.

While these proposals focus on the committee election mechanisms that improves the protocol's communication complexity, they omit defining all the behaviours that determine nodes' reputations, and the necessary security properties of such reputation mechanisms. In addition, Guru and RepuCoin study the flash attack resistance provided via empirical experiments rather than formal proofs, and thus cannot show their security guarantee under all protocol parameters and deployment environments. More specifically, we compare some of the existing Reputation-based SMRs and Sync-HotStuff [4] with our work in table I.

Other reputation-based systems. There have been many reputation mechanisms designed for other decentralised protocols and applications, such as leader election [14], multiparty computation [7], privacy-preserving systems [2], [17], [27], and peer-to-peer networks [18], [23]. However, these reputation mechanisms concern different types of behaviours that affect reputations in the targeted protocols and applications, thus cannot be directly adapted to SMR. For example, these protocols do not involve the voting process, while voting is a major behaviour that can lead to attacks and affect reputations in reputation-based SMR. In addition, privacy-preserving systems and peer-to-peer networks do not enforce nodes to have a consistent view on a node's reputation, while such consistency is crucial for reputation-based SMR where a node's reputation determines its voting power.

PoS-based blockchains with slashing support. Some PoS-based blockchains support *slashing*: if a node is held accountable due to misbehaviour, then its stake, and thus its voting power, will be slashed. It can be considered as a special case of reputation mechanisms where nodes' voting power is proportional to their financial stakes. However, the slashing mechanism is still a new concept with limited formal studies in the field. To the best of our knowledge, Tas et al. [26] is the only formal study on PoS slashing, which is under the permissionless setting.

B. Our contributions

In this paper, we fill this gap by providing the first formal treatment of the reputation-based SMR. We explicitly define the reputation mechanism and its security properties based on malicious behaviours known in SMR, and formalise the concept of reputation-based SMR. Our formalisation identifies a new security property *reputation-consistency* that is necessary for ensuring reputation-based SMR's safety, while being overlooked by existing literature. We then design a simple reputation mechanism that achieves these security properties, and combines it with the Sync-HotStuff [4] SMR protocol to obtain a simple and efficient reputation-based SMR. Compared to Sync-HotStuff, our reputation-based SMR only incurs an extra Δ in latency, where Δ is the maximum delay in synchronous networks.

Formalisation of reputation-based SMR (§III). We provide the first formal treatment of the reputation-based SMR, with an explicit formal definition on the reputation mechanism and flash-attack resistance. For the formalisation of reputation mechanisms, we start from characterising the types of behaviours that will affect the reputation. A node's reputation is then calculated from the protocol transcript that records these behaviours, including the node's historical block proposals and votes. Honest behaviours increase the reputation, and malicious behaviours will reduce the reputation.

We build our formalisation of reputation-based SMR based on existing rich literature in SMR [4]–[6], [22]. Our formalisation identifies and formalises a security property *reputation-consistency*, which is overlooked by previous studies but is necessary for ensuring reputation-based SMR's safety. The reputation-consistency property specifies that all honest nodes agree on the reputation values of other nodes, upon agreeing on a block. Without reputation-consistency, the adversary can make honest nodes to have conflicted views on some nodes' reputations and thus the quorums and leaders, breaking the protocol's safety.

A new reputation mechanism for SMR (§IV). We design a simple reputation mechanism that lies in our formal definition. The reputation function evaluates a node's reputation based on its historical behaviours in proposing and voting blocks, and employs well-understood functions in order to achieve the properties in our formal model while making the reputation value to lie in a certain interval $[\epsilon, 1]$, where ϵ is a sufficiently small value close to zero.

A new reputation-based SMR (§V). By using our reputation mechanism and the Sync-HotStuff SMR [4] as building blocks, we propose a simple reputation-based SMR. Our reputation-based SMR achieves the same performance in the best and worst case and resists flash-attacks, at the cost of only an extra Δ in latency atop Sync-HotStuff. Central to our protocol design, this extra Δ allows all honest nodes to detect and agree on accountable misbehaviours conducted by all Byzantine nodes, achieving the reputation-consistency property. Table I provides a comparison between our proposal and existing reputation-based SMR protocols, as well as Sync-HotStuff that we build upon.

II. Model of State Machine Replication

In this section, we provide the system model of state machine replication (SMR) protocols.

A. System model

Nodes. Let $\mathcal{P} = \{p_1, ..., p_n\}$ be the set of n nodes in the system. Each node p_i owns a key pair (sk_i, pk_i), and is uniquely identified by its public key pk_i in the system. We assume a Public-Key Infrastructure (PKI) who has the

Table I: Comparison of existing reputation-based SMR protocols, Sync-HotStuff and our work.

	System model		Security			Performance	
	Network model	Fault tolerance	Safety	Liveness	Flash attack resistance	Comm. compl.	Latency
Guru-PBFT	Psync.	1/3	✓	✓	✓	$O(mn) \sim O(m^3 + mn)$	$3\Delta \sim t_{\text{ViewChange}}$ †
RepuCoin	Sync.	1/3	✓	✓	✓	$O(mn)$	$3\Delta \sim t_{\text{Nakamoto}}$
Kleinrock et al.	Sync.	1/2	✓	✓	✗	$O(mn)$	$3\Delta \sim t_{\text{Nakamoto}}$
Sync-HotStuff	Sync.	1/2	✓	✓	✗	$O(n^2)$	$3\Delta \sim 4\Delta$
This work	Sync.	1/2	✓	✓	✓	$O(n^2)$	4Δ

† The latency is calculated assuming the global stabilisation time (GST) in partial synchrony has passed;
Guru-PBFT [10]: In the best case when the leader is honest, the communication includes a PBFT within the committee (of which the size is labelled as m) and broadcasting the final block by leader to all nodes, leading to $O(mn + m^2)$ and 3Δ. Since $mn \geq m^2$, $O(mn + m^2)$ asymptotically equals to $O(mn)$. In the worst case when consecutive leaders are Byzantine, the communication includes view change protocols, leading to $O(m^3 + mn)$ and $t_{\text{ViewChange}}$, where $O(m^3)$ comes from view change protocol within the committee, $O(nm)$ comes form the new view broadcast of new leader and $t_{\text{ViewChange}}$ depends on the concrete view change protocol design.
RepuCoin [29]: Similar to Guru, in the best case, the communication includes a PBFT within the committee and broadcasting the final block to all nodes, leading to $O(mn)$ and 3Δ. In the worst case when the leader of microblocks withholds blocks continuously, the t_{Nakamoto} is parameterised by the security level [15], [16].
Kleinrock et al. [20]: In the best case when the reputation system is secure, the communication includes all-to-all communication among two committees and broadcasting the final block to all nodes, leading to $O(mn)$ and 3Δ. In the worst case when the reputation system becomes insecure, the system falls back to PoS-based Nakamoto blockchain, t_{Nakamoto} depends on concrete PoS-based blockchain design.
Sync-HotStuff [4]: The communication includes all-to-all broadcasts among all nodes, leading to $O(n^2)$ communication complexity. In the best case, nodes take 3Δ to commit a block. In the worst case when the leader withholds blocks, nodes take 2Δ to abort the current view, and take another 2Δ to enter the current view and commit a block, leading to 4Δ latency in total.

knowledge of all nodes' public keys in order to prevent identity spoofing.

Ledger. All nodes jointly maintain a ledger formed as a blockchain, i.e., a chain of blocks. Transactions sent from users are packaged in each block. A block B_r at height r is formed as $B_r = (h^-, pk, txs)$, where $h^- = H(B_{r-1})$ is the hash of the predecessor block B_{r-1}, pk is the public key of the block producer, txs is a set of transactions. We call two blocks (B_i, B_j) at the same height, i.e., $i = j$, are conflicted blocks. We call a block B_r is a certified block, provided that there is a set of sign votes on this block to form a quorum certificate of nodes, a quorum certificate QC consist of more than $1/2$ voting power from nodes. We assume transactions submitted from users are valid.

Protocol transcripts. Apart from the ledger that concerns agreed transactions and protocol metadata, each node also maintains the protocol transcripts locally in order to determine other nodes' reputations. Specifically, the protocol transcripts include the set M_j of node p_j's proposed blocks and the set V_j of p_j's votes on blocks.

Network model. We adopt the synchronous network model where the adversary can delay the message delivery up to a known upper bound Δ.

Adversary model. The adversary can corrupt f out of n nodes, where $n \geq 2f + 1$. Corrupted nodes can behave arbitrarily (including deviating from the protocol), while the other honest nodes always follow the protocol. A corrupted node is also called *Byzantine* node. The adversary is computationally bounded, i.e., can only solve polynomial algorithms and thus cannot break standard cryptographic primitives such as hash functions and digital signatures. The adversary is static, i.e., it chooses set of f nodes to corrupt at the beginning of the protocol and cannot choose to corrupt other nodes afterwards.

B. Security properties

A Byzantine fault tolerant (BFT) state machine replication (SMR) protocol allows a set of nodes to maintain a linearisable and consistent ledger of blocks akin to a single non-faulty server. A BFT SMR protocol provides the following properties.

- **Safety:** Every two honest nodes do not commit different blocks at the same height.
- **Liveness:** If a transaction is received by an honest node, then the transaction will be eventually included in every honest node's ledger.

We briefly summarise possible attacks against these properties below.

Safety attacks. Typical attacks on SMR's safety include equivocation and private chain attacks. Equivocation means that the adversary directs Byzantine nodes to send different messages to different nodes, which may lead to disagreements among nodes. Private chain attacks mean that the adversary directs Byzantine nodes to work on a separate blockchain privately while following the protocol. There are different variants of private chain attacks, where grinding attacks are a notable one. In the grinding attack [8], [19], the adversary directs Byzantine nodes to "grind" (i.e., attempt to produce blocks after) all known blocks.

Liveness attacks. Typical attacks on SMR's liveness include the aforementioned equivocation and withholding attacks. Apart from safety, equivocation may prevent honest nodes to make decisions forever, breaking the liveness. Withholding means that the adversary directs Byzantine nodes to withhold messages to certain nodes, which may also prevent them to make decisions forever.

C. Performance metrics

BFT SMR concerns two performance metrics, namely communication complexity and latency.

- **Communication complexity:** The total amount of data (in bits) transferred to commit a block.
- **Latency:** The total amount of time taken to commit a block.

We stress that the protocol may achieve different values on these two metrics in best-case and worst-case executions.

III. Reputation-based state machine replication

In this section, we extend the SMR model to provide a formal definition of reputation-based SMR. We start from analysing misbehaviours in SMR protocols (§III-A), then formalise the reputation function that gives each node a reputation based on its historical (mis)behaviours (§III-B), and finally combines the reputation function into the SMR model to obtain the definition of the reputation-based SMR (§III-C).

A. Misbehaviours in SMR

A node's reputation depends on its historical behaviours, in which honest behaviours accumulate the reputation and misbehaviours reduce the reputation. While honest behaviours are specified by the protocol, we review misbehaviours in synchronous SMR protocols based on existing works on accountable Byzantine consensus [12], [13], [25]. We concern two aspects of bad behaviours, namely the types of behaviours (including block proposal and vote) and their reasons of being malicious (including equivocation, withholding, and other behaviours deviating from the protocol specification).

Equivocating blocks/votes. Equivocation means that a node performs behaviours conflicted with each other. Equivocating blocks mean that a node, who is the leader of an epoch, proposes multiple different blocks at the same height. Equivocating votes mean that a node votes for multiple different blocks at the same height.

In order to detect equivocating blocks, a node needs to receive two proposed blocks at the same height signed by the same node. In order to detect equivocating votes, a node needs to receive two votes on two different blocks at the same height signed by the same node.

Withholding blocks/votes. Withholding means that a node withholds messages that it should send according to the protocol. Withholding blocks mean that a node, who is the leader of an epoch, does not propose any block at this epoch. Withholding votes mean that a node does not vote blocks proposed by the leader.

In order to detect withholding blocks, a node needs to ensure no other honest node has received any block in an epoch, which usually requires an all-to-all broadcast. Detecting withholding votes is challenging. There are two scenarios that a node does not vote on time: 1) the node is Byzantine, and 2) the leader withholds its proposed block to a certain honest node, making this node unable to vote for any block. An honest node cannot distinguish between the two scenarios, where the node is Byzantine in the first scenario and is honest in the second scenario.

Malicious blocks/votes. Malicious blocks mean a non-leader node proposes a block. Malicious votes mean a node votes for a block that does not exist. Malicious blocks/votes will not affect the normal execution of the SMR protocol, and can be detected since the proposed block and vote carry the signature from the Byzantine node.

B. Definition of Reputation function

Given the malicious behaviours, we then define the reputation function $f_{\mathsf{Rep}}(\cdot)$ for SMR, including its syntax and security properties.

Definition 1 (Reputation function)**.** *Reputation function* $f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, V_j) \rightarrow \mu_j$ *takes input*

- ϵ *is initial reputation value for all nodes;*
- pk_j *is node* p_j*'s identity;*
- M_j *is the set of* p_j*'s block proposals so far; and*
- V_j *is the set of* p_j*'s votes so far,*

and outputs the reputation value $\mu_j \in [\epsilon, 1]$ *of node* j*. Function* f_{Rep} *satisfies the following properties:*

- ***(Initial reputation value distribution)*** *Giving any node* p_j *that has just entered the system an initial reputation value* ϵ*, which is a sufficiently small value close to* 0*.*
- ***(Malicious votes reduces reputation)*** *For any tuple* $(V_j, \hat{V}_j)$*, if* V_j *records more malicious votes of node* j *than* $\hat{V}_j$*, then* $f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, V_j) < f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, \hat{V}_j)$*.*
- ***(Equivocating/malicious/withholding blocks reduces reputation)*** *For any tuple* $(M_j, \hat{M}_j)$*, if* M_j *records more equivocating blocks of node* j *than* $\hat{M}_j$*, or* M_j *records more malicious blocks of node* j *than* $\hat{M}_j$*, or* M_j *records fewer valid blocks of node* j*, then* $f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, V_j) < f_{\mathsf{Rep}}(\epsilon, pk_j, \hat{M}_j, V_j)$*.*
- ***(Honest votes increase reputation)*** *For any tuple* $(V_j, \hat{V}_j)$*, if* V_j *records more votes on blocks than* $\hat{V}_j$*,then* $f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, V_j) > f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, \hat{V}_j)$*.*
- ***(Honest blocks increase reputation)*** *For any tuple*$(M_j, \hat{M}_j)$*, if* M_j *records more valid blocks than* $\hat{M}_j$*,then* $f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, V_j) > f_{\mathsf{Rep}}(\epsilon, pk_j, \hat{M}_j, V_j)$*.*

C. Definition of reputation-based SMR

We adapt the definition of SMR [4]–[6] to define a reputation-based SMR. Compared to the traditional SMR, reputation-based SMR replaces the one-man-one-vote design with weighted voting by reputation. In addition, reputation-based SMR requires a third property *reputation-consistency*, where honest nodes should have the same view on other nodes' reputations. Existing reputation-based SMR protocols [10], [20], [29] provide proofs on reputation-consistency implicitly within their security proofs, omitting its explicit definition.

Definition 2 (Reputation-based SMR)**.** *A reputation-based SMR protocol driven by the majority of voting power (reputation) in the system, and the voting power of each node*

The procedures of $f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, V_j) \rightarrow \mu_j$:

$$S(M_j) = \max(0, |M_j| - \xi_{\text{w}}|\text{Withholding}(M_j)| - \xi_{\text{e}}|\text{Equivocating}(M_j)|) - \xi_{\text{mp}}|\text{Malicious}(M_j)|)$$

$$S(V_j) = \max(0, |V_j| - \xi_{\text{mv}}|\text{Malicious}(V_j)|)$$

$$\mu_j = \epsilon + \tanh\left[\gamma(S(V_j) + S(M_j))\right]$$

where

- ξ_{w} is the penalty factor of withholding blocks
- ξ_{e} is the penalty factor of equivocating blocks
- ξ_{mp} is the penalty factor of malicious blocks
- ξ_{mv} is the penalty factor of malicious votes
- γ is the factor of increasing reputation
- $S(V_j)$ is the scoring function for node voting behavior
- $S(M_j)$ is the scoring function for node block proposal behavior

Figure 1: reputation function $f_{\mathsf{Rep}}(\epsilon, pk_j, M_j, V_j) \rightarrow \mu_j$

determined by their behavior. When the adversary's nodes that have less than half of the total reputation (which is always true when the majority of nodes are honest), the reputation-based SMR provides the following three properties.

- ***Safety:*** *Every two honest nodes do not commit different blocks at the same height.*
- ***Reputation-consistency:*** *At the same height, every two honest nodes do not have different reputation values for a node.*
- ***Liveness:*** *If a transaction is received by an honest node, then the transaction will be eventually included in every honest node's ledger.*

Flash attack resistance. In addition, we consider SMR protocols that resist against flash attacks [11], [29], where the adversary manages to enrol a large number of its nodes into the system, so that it can temporarily exceed the fault tolerance capacity and break some security properties. Guru [10] and RepuCoin [29] observe that the reputation mechanism can provide resistance against flash attacks for SMR protocols.

Definition 3 (Flash attack resistance)**.** *A BFT SMR protocol resists against flash attacks if it can ensure safety, reputation-consistency and liveness for a certain period of time under an adversary who can corrupt more than f out of n nodes, where $n \geq 2f + 1$.*

IV. Construction of the Reputation Function

In this section, we provide a simple construction of the reputation function that satisfies all properties defined in §III-C.

A. Construction

Figure 1 shows our construction of the reputation function. Intuitively, the function scores a node's proposing and voting behaviours, combines them to a single value, and finally normalises it to $[\epsilon, 1]$. The function takes node p_j's public key pk_j, block proposal set M_j and vote set V_j as input, and computes as follows to output the reputation value μ_j of p_j[1]

Scores of block proposal/voting behaviours. All misbehaviours can be obtained from parsing M_j and V_j. We define score functions $S(M_j)$ and $S(V_j)$ for scoring proposing and voting behaviours, respectively. $S(M_j)$ considers the number $|\text{Withholding}(M_j)|$, $|\text{Equivocating}(M_j)|$, $|\text{Malicious}(M_j)|$ of withholding, equivocating and malicious blocks, respectively, and applies a penalty factor ξ_w, ξ_e and ξ_{mp} for each of them. $S(V_j)$ considers the number $|\text{Malicious}(V_j)|$ of malicious votes, and applies a penalty factor ξ_{mv} as well. We use the total number of block proposals/votes to subtract the number of each misbehaviours multiplied by its penalty factor. $S(M_j)$ and $S(V_j)$ specifies the smallest possible reputation value of zero by using function $max(\cdot)$.

Combining block proposal/voting scores together. We then combine the scores $S(M_j)$ and $S(V_j)$ to a single value, and normalise it to the final reputation value μ_j. Specifically, we add $S(M_j)$ and $S(V_j)$ together and multiply it with a factor γ. where γ controls the slope of the reputation function w.r.t. behaviours. After that, we use function $\tanh(\cdot)$ to normalise the value $\gamma(S(V_j)+S(M_j))$ to internal $[\epsilon, 1]$, yielding the final reputation value μ_j.

B. Security analysis

We show that our reputation function achieves all the properties defined in §III-B.

Initial reputation value distribution. Since a newly joined node p_j does not have any block proposal or vote, $S(M_j)$ and $S(V_j)$ will be zero, and thus $\tanh[\gamma(S(V_j) + S(M_j))] = 0$.

Malicious votes reduces reputation. If V_j records more malicious votes of node j than $\hat{V}_j$, then $|\text{Malicious}(V_j)| > |\text{Malicious}(\hat{V}_j)|$. Since $\xi_{\text{mv}} > 1$, $S(V_j) < S(\hat{V}_j)$, leading to smaller reputation.

Equivocating/malicious/withholding blocks reduces reputation. Similar to above, if M_j records more equivocating blocks of node j than $\hat{M}_j$, or M_j records more malicious blocks of node j than $\hat{M}_j$, or M_j records fewer valid blocks of node j, then $S(M_j) < S(\hat{M}_j)$ with $\xi_{\text{w}}, \xi_{\text{e}}, \xi_{\text{mp}} > 1$, leading to smaller reputation.

Honest block proposals/votes increase reputation. If V_j records more votes on blocks than $\hat{V}_j$ or M_j records more valid blocks than $\hat{M}_j$, then $S(V_j) > S(\hat{V}_j)$ or $M(V_j) > M(\hat{V}_j)$, respectively. Either case will lead to a larger reputation.

V. Construction of the Reputation-based SMR Protocol

In this section, we provide a reputation-based SMR protocol. We build our protocol upon the reputation mechanism in §IV. Our protocol achieves the same performance in the

[1] In this paper, we do not specify whether to remove nodes with 0 reputation, because their presence does not affect the security properties of our protocol.

best and worst case and resists flash-attacks, at the cost of only an extra Δ in latency atop Sync-HotStuff. Central to our protocol design, this extra Δ allows all honest nodes to detect and agree on accountable misbehaviours conducted by all Byzantine nodes, achieving the reputation-consistency property.

A. Design overview

Given the rich literature in the field of synchronous SMR protocols, we consider building our reputation-based SMR based on one of them. For simplicity, we choose Sync-HotStuff [4], a synchronous SMR protocol that is specialised for simplicity and achieves practical performance, i.e., latency of 3Δ and communication complexity of $O(n^2)$.

The first challenge of building a reputation-based SMR is to ensure the reputation-consistency property, which is not considered in traditional SMR protocols. To ensure reputation-consistency, nodes have to share a consistent view on all nodes' historical behaviours, especially misbehaviours. Traditional SMR protocols usually aim at identifying misbehaviours that affect protocol execution, neglecting other misbehaviours such as malicious blocks and votes. For example, Sync-HotStuff only allows nodes to identify equivocating blocks and withholding blocks, but not the others summarised in §III-A.

Another challenge is to distinguish between honest behaviours and misbehaviours. We observe that only certain misbehaviours can be irrefutably identified before all honest nodes have committed the block in an epoch. If there are some honest nodes that have not committed a block, then these nodes cannot distinguish between two equivocating blocks, until one of them reaches a quorum of votes.

Given the above two challenges, it is necessary to add an extra phase after the Byzantine broadcast. This phase will be dedicated for allowing nodes to consistently identify misbehaviours. Specifically, for those misbehaviours that can be irrefutably identified by the adversary's messages (such as equivocating blocks and malicious blocks/votes), nodes will broadcast the evidences of these misbehaviours in this phase.

Last, since reputation-based SMR demands reputation-weighted voting, we adapt the Byzantine broadcast of Sync-HotStuff to such a weighted setting.

B. Primitives

Our protocol use the Sync-HotStuff as the building block. Sync-HotStuff employs two primitives, namely round-robin leader election and Byzantine broadcast.

Round-robin leader election. Round-robin leader election allows nodes to elect a leader for each epoch. The round-robin leader election algorithm takes the last block as input, and outputs a random leader from all nodes excluding last f leaders and misbehaving leaders.

Byzantine broadcast. Byzantine broadcast is a protocol that allows a certain node (called sender) to consistently distribute a message among n nodes, in which up to f nodes are Byzantine.

Definition 4 (Byzantine broadcast [5])**.** *Byzantine broadcast satisfies the following properties:*

- ***(Agreement)*** *If two honest nodes commit value B and B' respectively, then $B = B'$.*
- ***(Termination)*** *All honest nodes eventually commit a value.*
- ***(Validity)*** *If the designated sender is honest, then all honest nodes commit on the sender's value.*

We use the Byzantine Broadcast (BB) of the Sync-HotStuff [4], the state-of-the-art synchronous SMR protocol with $O(n^2)$ communication complexity and 3Δ latency. To apply the reputation mechanism, we adapt it to the weighted setting, where voting power of a node is in proportional to its reputation in the system. That is, committing a block requires more than half of the total voting power.

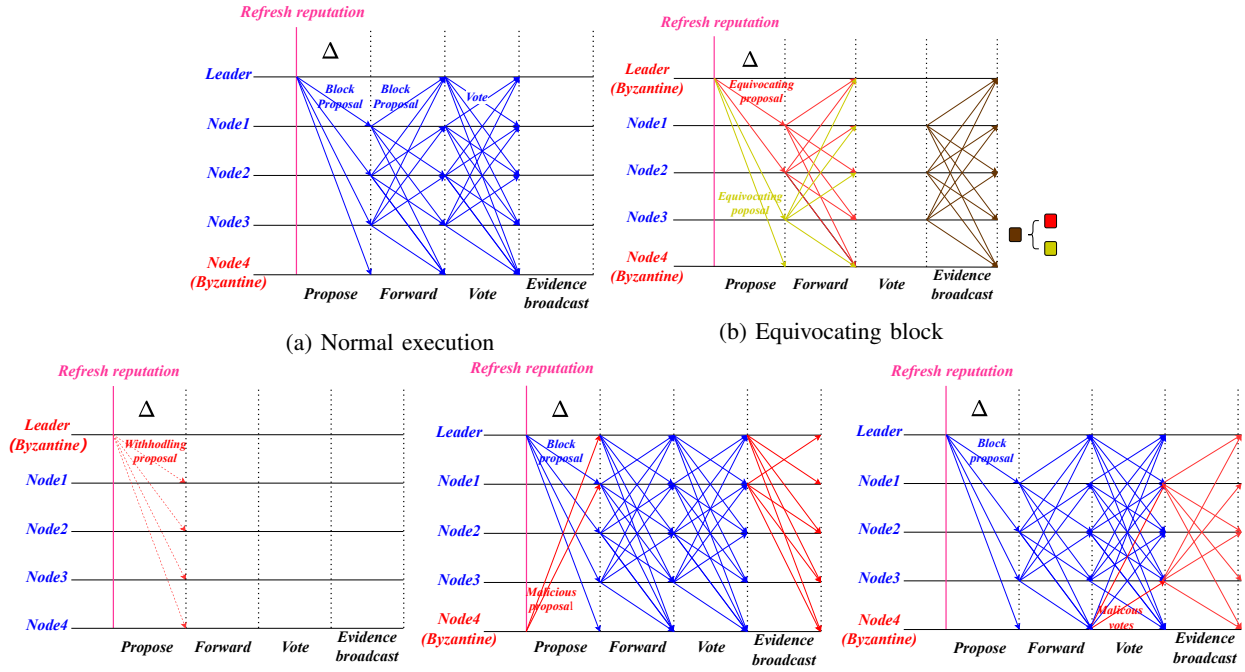
C. Our protocol

Based on the above primitives, we provide our reputation-based SMR protocol. Figure 3 provides the full protocol specification. Figure 2 depicts the protocol execution under normal case and under misbehaviours. The protocol consists of three phases, namely *refresh reputation*, *generate block*, *broadcast evidence*. We describe their processes through an epoch r in a node p_i's view.

Refresh reputation. At the beginning of each epoch, for every node p_j, node p_i refreshes its reputation by $\mu_j \leftarrow f_{\mathsf{Rep}}(pk_j, M_j, V_j)$, where (M_j, V_j) are determined by node p_j's block proposal and voting behaviors, and åre updated though the (non-blocking) evidence collection in the previous epoch.

Generate block. This phase is identical to the Sync-HotStuff protocol, where nodes elect a leader to propose a block and trigger a BB over this block. Specifically, nodes execute the round-robin leader election function to elect a leader pk_L. If node p_i is the leader (i.e., $pk_i = pk_L$), then it proposes block B_r and triggers BB over B_r. Otherwise, when node p_i receives the block B_r from leader pk_L, p_i verifies block B_r's validity. If B_r is valid, then p_i broadcasts it, votes for block B_r following the BB protocol. When the timer expires, if detecting misbehaviours of malicious/equivocating/withholding blocks, then node p_i commits an empty block $\perp$, and counts no vote for any node in this epoch. Otherwise, node p_i commits block B_r and counts one vote every node. Note that we set the timer to be 4Δ and Sync-HotStuff's BB protocol has the latency of 3Δ. The last Δ will be used for broadcasting evidences for misbehaviours, which will be explained below.

Evidence broadcast. This phase happens when the BB protocol terminates and Timer has a Δ remaining. It focuses on broadcasting evidences of nodes' misbehaviours, such that nodes have a consistent view on all nodes' historical behaviours. The evidences include those of malicious votes/blocks, equivocating blocks and withholding blocks, each described below. Note that equivocating and withholding votes cannot be detected as analysed in §III-A, and withholding block can be detected without an evidence.



(a) Normal execution

(b) Equivocating block

(c) Withholding block

(d) Malicious block

(e) Malicious vote

Figure 2: Different execution scenarios of our reputation-based SMR.

- **(Equivocating block, Figure 2b)** If node p_i receive two different blocks (B'_r, B_r) with signatures (σ', σ) both signed by the leader pk_L, then node p_i considers them as equivocating blocks and informs other nodes by broadcasting

$$\langle \texttt{Blame-Equivocating-Blocks}, pk_L, B_r, B'_r, \sigma, \sigma' \rangle_i$$

  where $\langle\cdot\rangle_i$ means a message signed by node p_i. Since the leader is Byzantine, node p_i will commit an empty block $\perp$ when Timer expires.
- **(Withholding block, Figure 2c)** If node p_i receives no block proposed by the leader after the vote phase, then this means that all honest nodes do not receive the leader and the leader is withholding blocks. In this case, p_i commit empty block $\perp$ as leader's block when Timer expires.
- **(Malicious block, Figure 2d)** If node p_i receives a block B'_r proposed by a non-leader node p_j with signature σ_j, then p_i considers this block as a malicious block and informs other nodes by broadcasting

$$\langle \texttt{Blame-Malicious-Block}, pk_j, B'_r, \sigma_j \rangle_i$$

- **(Malicious vote, Figure 2e)** If node p_i receives a signature σ_j associated with block $B'_r \neq B_r$ where B_r is the honest block, then p_i considers B'_r as a malicious vote and informs other nodes by broadcasting

$$\langle \texttt{Blame-Malicious-Vote}, pk_j, B'_r, \sigma_j \rangle_i$$

Note that, these two misbehaviours cannot affect the normal operation of our protocol, thus node p_i will commit the block proposed by leader.

Non-blocking evidence collection. When timer Timer expires, node p_i updates the reputations of each node p_j by processing their block proposal set $M_{i,j}$ and vote set $V_{i,j}$ as follows.

- **(Malicious vote/block)** If p_i receives an evidence of a malicious block (resp. vote) from node p_j, then p_i records this malicious block (resp. vote) in M_{ij} (resp. $V_{i,j}$).
- **(Equivocating block)** If p_i receives an evidence of equivocating blocks from leader node p_j, then p_i records these equivocating blocks in $M_{i,j}$ and ignores all votes received in this epoch.
- **(Withholding block)** If p_i does not receive the block from leader node p_j throughout this epoch, then it records an empty block ($\perp$) in $M_{i,j}$ and ignores all votes received in this epoch.

D. Security analysis

Lemma 1 (Reputation consistency)**.** *If BB satisfies agreement and termination, then when the* Timer *expires, for any node p_z and any two honest nodes (p_i, p_j), $(M_{i,z}, V_{i,z}) = (M_{j,z}, V_{j,z})$.*

Proof. For the sake of contradiction, assuming there exists two honest nodes p_i and p_j such that $(M_{i,z}, V_{i,z}) \neq (M_{j,z}, V_{j,z})$ when the Timer expires. As BB satisfies agreement and termination, in each epoch, any two honest nodes either commit the same block proposed by a leader, or commit an empty block (i.e., $\perp$).

Let r be the current epoch. Upon timer Timer_{r-1} of last epoch $r-1$ expires, node p_i executes as follows.

1) **Refresh reputations and set timer.** For every node p_j, node p_i executes $\mu_j \leftarrow f_{\mathsf{Rep}}(pk_j, M_j, V_j)$ to refreshes node p_j's reputation μ_j. Node p_i sets a timer Timer_r of 4Δ.
2) **Generate block.** Node p_i executes leader election, Byzantine broadcast and commit steps as follows.
   a) **Leader election.** Node p_i executes the round-robin leader election to elect a leader pk_L, then checks if it is elected as leader L by checking whether $pk_i = pk_L$.
   b) **Byzantine broadcast.** If p_i is the elected leader, then p_i locally samples a set of transactions txs and proposes block $B_r = (H(B_{r-1}), pk_i, txs)$, and triggers BB over B_r with other nodes. Otherwise, p_i waits for the leader block proposal and follows BB.
   c) **Block commit.** When the Timer_r expires, if detecting misbehaviours of equivocating/withholding blocks, then node p_i commits an empty block $\perp$, and counts no vote for any node in this epoch. Otherwise, node p_i commits block B_r and counts one vote every node.
3) **Evidence broadcast.** When Timer_r remains Δ, node p_i executes as follows.
   - **Detect malicious vote/block:** If node p_i receives a signature σ_j associated with block $B'_r \neq B_r$, then p_i broadcasts $\langle \texttt{Blame-Malicious-Vote}, pk_j, B'_r, \sigma_j \rangle_i$ to other nodes. In addition, if node p_i receives a block B'_r proposed by p_j with signature σ_j is not the leader of epoch r, then broadcasts $\langle \texttt{Blame-Malicious-Block}, pk_j, B'_r, \sigma_j \rangle_i$.
   - **Detect equivocating block.** If node p_i receives two different blocks (B_r, B'_r) with signatures (σ_r, σ'_r) signed by the leader pk_L of epoch r, node p_i broadcasts $\langle \texttt{Blame-Block-Equivocation}, pk_L, B_r, B'_r, \sigma_r, \sigma'_r \rangle_i$ to all nodes.
   - **Detect withholding block.** if node p_i receives no block, then $M_{i,L}$ will not record any block proposed by the current leader L.
4) **(Non-blocking) Evidence collection** Upon receiving any signature σ_j, node p_i adds it to $V_{i,j}$. Upon receiving any block $B_r \neq \perp$ proposed by node p_j, node p_i adds it to $M_{i,j}$.

Figure 3: Specification of our reputation-based SMR protocol.

If the committed block is $\perp$, then this means that the leader equivocates or withholds the block, both can be detected by at least one honest node when Timer has Δ remaining. After this Δ, all honest nodes will be informed of the equivocation or withholding and thus update their local sets, contradicting to our assumption.

If the committed block is not empty, then there can be two possible scenarios: 1) there exists no misbehaviour, 2) there exists misbehaviours that do not affect protocol execution, which can only be malicious block or malicious vote.

- **Case 1:** when Timer expires, all honest nodes update their local sets consistently without recording any misbehaviour, contradicting to our assumption.
- **Case 2:** At the last Δ, all honest nodes will be informed of the malicious block and/or vote and update their local sets respectively, contradicting to our assumption.

Thus, if the adversary can break reputation-consistency of our reputation-based protocol, then it has to break the agreement or termination of the BB protocol, which only happens with negligible probability. □

Lemma 2 (Safety)**.** *If the BB protocol satisfies agreement, then our reputation-based SMR protocol satisfies safety.*

Proof. For the sake of contradiction, assuming the reputation-based SMR protocol does not satisfy safety. That is, there exists an epoch r where two honest nodes commit different blocks $B \neq B'$. Given the reputation mechanism, for each honest node and Byzantine node, the honest node's reputation is no less than that of the Byzantine node. As there are more than $\frac{1}{2}$ honest nodes in the system, honest nodes control more than $\frac{1}{2}$ voting power in the system. Under this setting, the BB protocol satisfies agreement. Therefore, if they commit block B and B' in epoch r, then $B = B'$, contradicting to our assumption. Thus, if the adversary can break the safety of our our reputation-based SMR protocol, then it has to corrupt voting power in network more than $\frac{1}{2}$ or break BB's agreement, which only has negligible probability. □

Lemma 3 (Liveness)**.** *If the BB protocol satisfies termination, then our reputation-based SMR protocol satisfies liveness.*

Proof. For the sake of contradiction, assuming the protocol does not satisfy liveness. That is, there exists a transaction that is received by an honest node in epoch r, but eventually is not committed in every honest node's ledger. As the voting power of byzantine nodes less than $\frac{1}{2}$. The leader is either honest or Byzantine. If the leader is honest, then the leader will propose a block including his transaction to all nodes, as BB protocol satisfies termination, more than $\frac{1}{2}$ honest voting power will commit this block. If the leader is Byzantine, then it either 1) follows the protocol, 2) withholds the block, and

3) proposes multiple conflicting blocks.

- **Case 1:** This case is similar with the honest leader case;
- **Case 2/3:** by termination, nodes will commit an empty block and enter the next epoch.

With round-robin, the probability that consecutive leaders are Byzantine decreases exponentially with the number of consecutive epochs, as analysed in [9]. Thus, eventually, there will be an honest leader who will commit all transactions received by honest nodes, contradicting to the assumption. Thus, if the adversary can break the liveness of our reputation-based SMR protocol, then it has to corrupt voting power in network more than $\frac{1}{2}$ or break BB's termination, which only has negligible probability. □

E. Flash attack resistance analysis

Recall that the adversary is static, which means that the set of Byzantine nodes does not change throughout the execution. After a sufficient long execution, the average reputation of the honest nodes can be infinitely close to 1. Base on this, we assume a sufficient long execution with $f+1$ honest nodes and f Byzantine nodes, and assume the average reputation of Byzantine nodes is $\eta < 1$. Thus, the total voting power of honest nodes (resp. Byzantine nodes) is $f+1$ (resp. ηf). In order to break safety or liveness, the adversary needs $f+1-\eta f = (1-\eta)f+1$ extra voting power.

Lemma 4 (Reputation accumulation)**.** *Let $f_{\mathsf{Rep}}(\cdot)$ be our reputation function defined in §IV-A. Assuming all existing n nodes behave honestly. If x nodes newly join the system and honestly follow the protocol for r epochs, then the reputation value μ_j for each p_j of these nodes will be*

$$\mu_j = \epsilon + \tanh\left[\frac{(n+x+1)\gamma r}{n+x}\right]$$

Proof. When a node p_j newly joins the system, its reputation is 0. During r epochs, it will vote for r times, and on average will be elected as leader and propose a block for $\frac{r}{n+x}$ times, where n is the number of existing nodes and x is the number of newly joined nodes. Thus, $S(V_j)$ will become r and $S(M_j)$ will become $\frac{r}{n+x}$, leading to reputation value

$$\mu_j = \epsilon + \tanh\left[\gamma(r+\frac{r}{n+x})\right] = \epsilon + \tanh\left[\frac{(n+x+1)\gamma r}{n+x}\right]$$

. □

Lemma 5 (Flash attack resistance, quantified in epochs)**.** *In order to launch a flash attack, the adversary needs*

$$x = \frac{(1-\eta)f+1}{\epsilon + \tanh\left[\frac{(n+x+1)\gamma r}{n+x}\right]}$$

new nodes to join the system and behave honestly for r epochs.

Proof. In order to launch the flash attack, the adversary needs to gain extra voting power of $(1-\eta)f+1$. By Lemma 4, to do so in r epochs, the adversary needs

$$\frac{(1-\eta)f+1}{\mu_j} = \frac{(1-\eta)f+1}{\epsilon + \tanh\left[\frac{(n+x+1)\gamma r}{n+x}\right]}$$

new nodes to behave honestly for r epochs. □

Given that r consecutive epochs take $t = 4\Delta r$, we have $r = \frac{t}{4\Delta}$, and thus the following corollary.

Corollary 1 (Flash attack resistance, quantified in time)**.** *In order to launch a flash attack, the adversary needs*

$$x = \frac{(1-\eta)f+1}{\epsilon + \tanh\left[\frac{(n+x+1)\gamma t}{4\Delta(n+x)}\right]}$$

new nodes to join the system and behave honestly for a time period of t.

F. Performance analysis

Communication complexity. Recall that Sync-HotStuff has the communication complexity of $O(n^2)$. Since the extra evidence broadcast phase costs at most $O(n^2)$ (when malicious blocks/votes happen), our protocol also achieves $O(n^2)$. That is, our protocol incurs no asymptotic communication complexity compared to Sync-HotStuff.

Latency. Recall that Sync-hotStuff commits a block for every 3Δ. With the extra evidence broadcast phase, our protocol commits a block for every 4Δ, leading to the latency of 4Δ.

VI. Conclusion

This paper has provided the first formal treatment on the reputation-based SMR, a proven secure reputation mechanism for SMR, and an efficient reputation-based SMR protocol based on Sync-HotStuff.

There are still several open challenges in this field. First, we consider the implementation and experimental evaluation of our proposed reputation-based SMR as a future work. Another promising direction is to extend our block to partially synchronous networks. In addition, while this work considers the permissioned setting with a fixed set of nodes, the reputation mechanism in the permissionless setting remains as an open challenge.

References

[1] Amazon. https://www.amazon.com/
[2] Anonrep
[3] Youtube. https://www.youtube.com/
[4] Abraham, I., Malkhi, D., Nayak, K., Ren, L., Yin, M.: Sync hotstuff: Simple and practical synchronous state machine replication. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 106–118. IEEE (2020)
[5] Abraham, I., Nayak, K., Ren, L., Xiang, Z.: Byzantine agreement, broadcast and state machine replication with near-optimal good-case latency. arXiv preprint arXiv:2003.13155 (2020)
[6] Abraham, I., Nayak, K., Ren, L., Xiang, Z.: Good-case latency of byzantine broadcast: A complete categorization. In: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing. pp. 331–341 (2021)
[7] Asharov, G., Lindell, Y., Zarosim, H.: Fair and efficient secure multiparty computation with reputation systems. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 201–220. Springer (2013)
[8] Bagaria, V., Dembo, A., Kannan, S., Oh, S., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Proof-of-stake longest chain protocols: Security vs predictability. arXiv preprint arXiv:1910.02218 (2019)

[9] Bhat, A., Shrestha, N., Luo, Z., Kate, A., Nayak, K.: Randpiper–reconfiguration-friendly random beacons with quadratic communication. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 3502–3524 (2021)

[10] Biryukov, A., Feher, D., Khovratovich, D.: Guru: Universal reputation module for distributed consensus protocols. Cryptology ePrint Archive (2017)

[11] Bonneau, J., Felten, E.W., Goldfeder, S., Kroll, J.A., Narayanan, A.: Why buy when you can rent? bribery attacks on bitcoin consensus (2016)

[12] Civit, P., Gilbert, S., Gramoli, V.: Polygraph: Accountable byzantine agreement. In: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS). pp. 403–413. IEEE (2021)

[13] Civit, P., Gilbert, S., Gramoli, V., Guerraoui, R., Komatovic, J., Milosevic, Z., Serendinschi, A.: Crime and punishment in distributed byzantine decision tasks (extended version). Cryptology ePrint Archive (2022)

[14] Cohen, S., Gelashvili, R., Kogias, L.K., Li, Z., Malkhi, D., Sonnino, A., Spiegelman, A.: Be aware of your leaders. arXiv preprint arXiv:2110.00960 (2021)

[15] Gaži, P., Ren, L., Russell, A.: Practical settlement bounds for proof-of-work blockchains. Cryptology ePrint Archive (2021)

[16] Guo, D., Ren, L.: Bitcoin's latency–security analysis made simple. arXiv preprint arXiv:2203.06357 (2022)

[17] Hasan, O., Brunie, L., Bertino, E.: Privacy-preserving reputation systems based on blockchain and other cryptographic building blocks: A survey. ACM Computing Surveys (CSUR) **55**(2), 1–37 (2022)

[18] Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web. pp. 640–651 (2003)

[19] Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Annual international cryptology conference. pp. 357–388. Springer (2017)

[20] Kleinrock, L., Ostrovsky, R., Zikas, V.: Proof-of-reputation blockchain with nakamoto fallback. In: International Conference on Cryptology in India. pp. 16–38. Springer (2020)

[21] Kogias, E.K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing bitcoin security and performance with strong consistency via collective signing. In: 25th usenix security symposium (usenix security 16). pp. 279–296 (2016)

[22] Malkhi, D., Reiter, M.: Byzantine quorum systems. In: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. pp. 569–578 (1997)

[23] Marti, S., Garcia-Molina, H.: Taxonomy of trust: Categorizing p2p reputation systems. Computer Networks **50**(4), 472–484 (2006)

[24] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review p. 21260 (2008)

[25] Sheng, P., Wang, G., Nayak, K., Kannan, S., Viswanath, P.: Bft protocol forensics. In: Proceedings of the 2021 ACM SIGSAC conference on computer and communications security. pp. 1722–1743 (2021)

[26] Tas, E.N., Tse, D., Yu, F., Kannan, S.: Babylon: Reusing bitcoin mining to enhance proof-of-stake security. arXiv preprint arXiv:2201.07946 (2022)

[27] Wang, X.O., Cheng, W., Mohapatra, P., Abdelzaher, T.: Artsense: Anonymous reputation and trust in participatory sensing. In: 2013 Proceedings IEEE INFOCOM. pp. 2517–2525. IEEE (2013)

[28] Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)

[29] Yu, J., Kozhaya, D., Decouchant, J., Esteves-Verissimo, P.: Repucoin: Your reputation is your power. IEEE Transactions on Computers **68**(8), 1225–1237 (2019)