Modeling Mobile Crash in Byzantine Consensus

Hans Schmiedel
Monash University
Australia
hans.schmiedel@monash.edu

Runchao Han
BabylonChain Inc.
Australia
me@runchao.rocks

Qiang Tang
The University of Sydney
Australia
qiang.tang@sydney.edu.au

Ron Steinfeld
Monash University
Australia
ron.steinfeld@monash.edu

Jiangshan Yu *
The University of Sydney
Australia
jiangshan.yu@sydney.edu.au

***Abstract*—Targeted Denial-of-Service (DoS) attacks have been a practical concern for permissionless blockchains. Potential solutions, such as random sampling, are adopted by blockchains. However, the associated security guarantees have only been informally discussed in prior work. This is due to the fact that existing adversary models are either not fully capturing this attack or giving up certain design choices (as in the sleepy model or asynchronous network model), or too strong to be practical (as in the mobile Byzantine adversary model).**

This paper provides theoretical foundations and desired properties for consensus protocols that resist against targeted DoS attacks. In particular, we define the *Mobile Crash Adaptive Byzantine (MCAB) model* to capture such an attack. In addition, we identify and formalize two properties for consensus protocols under the MCAB model, and analyze their trade-offs. As case studies, we prove that Ouroboros Praos and Algorand are secure in our MCAB model, giving the *first* formal proofs supporting their security guarantee against targeted DoS attacks, which were previously only informally discussed. We also illustrate an application of our properties to secure a streamlined BFT protocol, chained Hotstuff, against targeted DoS attacks.

I. Introduction

Byzantine Fault Tolerant (BFT) protocols provide a performant and energy-efficient alternative to Proof-of-Work (PoW) consensus for distributed ledgers. Many BFT protocols follow the leader-based design [1]–[4], which uses a leader to propose values, replaced in case of misbehavior or simply for fairness.

For leader-based protocols, and generally deterministic consensus, security is guaranteed against a static or adaptive Byzantine adversary, who can corrupt up to a threshold of f nodes, and under a synchronous or partially synchronous network to circumvent the Fischer, Lynch, and Paterson (FLP) impossibility result [5]. In these network models, an honest leader's message is guaranteed to be delivered within a known upper bound of network delay (after global stabilization time in a partially synchronous network). The bounded network delay enables leader rotations after a timeout.

Mobile DoS attacks. For internet implementations, an adversary may attempt Denial-of-Service (DoS) attacks on leaders to stall the protocol. If leaders are known a priori, then the adversary can target every future leader in each rotation. To highlight the capability of moving the target between leaders, we call such an attack a *mobile DoS* attack. To date, various DoS attacks on deployed blockchains have already been attempted [6], [7], illustrating the need to consider protocols secure in a model fully capturing DoS threats.

While such an attack is a common attack vector, classic security models do not cover it — this attack is orthogonal to the Byzantine threshold or network assumption. As a result, BFT protocols under these models either do not tackle such an attack, or only provide an informal discussion on it.

For example, two deployed Proof-of-Stake (PoS) blockchains using provably secure protocols, Cardano and Algorand, seem to consider handling DoS as a design goal: Algorand [8] states that *"Algorand is not susceptible to either targeted compromises or DoS attacks"* and Cardano documentation states that Ouroboros Praos prevents DoS attacks [9] without proofs. In fact, Ouroboros Praos and Algorand are shown to be secure under an adaptive adversary in a semi-synchronous or weak synchronous network respectively, either of which does not allow an attacker to perform mobile DoS attacks. This highlights the gap between the considered adversary in practice and in theoretical security analysis. A deep and systematic understanding is lacking, including the desired properties required to resist against such attacks.

Existing models and adversaries. Alternative system models (summarised in Table I and analyzed in §VII) have been proposed in the literature. However, they either only partially capture the mobile DoS attack, or make it impossible to design protocols with desired features. For example, it is impossible to design a system that is secure in the Sleepy or Mobile Byzantine model in a network that is not synchronous [31], [34]. For the asynchronous network model, deterministic consensus is impossible due to the FLP result [5]. Therefore, an adversary model is needed to capture mobile DoS attacks but still allow consensus under different network models, and allow both probabilistic or deterministic guarantees, without contradicting the FLP result [5].

Our contribution. This work addresses the research gap surrounding the mobile DoS attack with three major contribu-

* Corresponding author

TABLE I: Comparison of existing works and their system models. Further discussions for each column are in Section VII.

Existing works	Adversary model	Network	*Deterministic consensus*	*No identity re-establishment**	*Partition tolerance possible***	*Models mobile DoS*
[10]–[13]	Adaptive/static Byzantine	Asynchronous	✗	✓	✓	✓
This work	**MCAB**		✗	✓	✓	✓
[1], [2], [4]	Adaptive/static Byzantine	Partially synchronous	✓	✓	✓	✗
[14]–[16]	Mixed faults		✓	✓	✓	✗
[17]	Alive-but-corrupt faults		✓	✓	✓	✗
This work	**MCAB**		✓	✓	✓	✓
[18]–[20]	Adaptive Byzantine	Synchronous	✓	✓	✓	✗
[21]–[25]	Mobile Byzantine		✓	✗	✓	✓
[2], [26]–[28]	Slowly Mobile Byzantine		✓	✗	✓	✗
[29]–[31]	Adaptive Sleepy model		?†	✓	✗	✓
[18], [32], [33]	Mobile sluggish faults		✓	✓	✗	✓
This work	**MCAB**		✓	✓	✓	✓

* Identity re-establishment is a very strong requirement, often requires manual operations, making it impractical (More details in Section VII-C).
** This indicates the possibility of making protocols simultaneously tolerate partition and secure against a given adversary model. (More details in Section VII-D.)
† This is still an open question as per [31, Section 6].

tions. First, we define an adversary model, the *mobile crash adaptive Byzantine (MCAB)* adversary. While the mobile crash adversary solely models mobile DoS attacks, it can be combined with the widely accepted adaptive Byzantine adversary for the targeted applications, i.e., reaching consensus over the Internet. We adapt and extend the concept of roles [35] to model consensus, in order to formally analyze the security gap between the MCAB model and the adaptive Byzantine model. Second, we explore and identify the entire design space for maintaining liveness of consensus protocols under the MCAB model through two key properties that we call *abundance* and *concealment*, and analyze their trade-offs. Our general positive results show that either one of those properties are sufficient for liveness in the MCAB model, and can serve as a modular tool for simplifying the analysis of existing and future protocols in the MCAB model. Third, to illustrate the modular utility of our general results, we leverage our findings on case studies: we show that Ourorobors Praos and Algorand are indeed still secure against the MCAB adversary by showing that they satisfy our concealment key property above. We also show how to augment Hotstuff to remain secure against the MCAB adversary via showing its abundance property.

*Formalizing the model and roles.*We formally model consensus by adopting the concept of roles. In addition, to decouple the adversary's capacity from network assumption, we define the capacity of an adversary to corrupt nodes as oracles, allowing our model to be applied to different networks, not only covering different models in the literature but also providing new unexplored scenarios (as shown in Fig 1). The MCAB adversary has access to f adaptive Byzantine faults, and c mobile crash faults. Assumed bounds on f and c enable a fine-grained choice of adversarial power for different applications.

Properties for liveness in the MCAB adversary model. We define two properties and prove that either is necessary, and both are sufficient for liveness:

- *Concealment*, inspired by the random secret leaders in blockchains [8], [36] and BFT achieving constant expected rounds [37], covers roles performed by nodes whose identities are only revealed after they broadcast the role's messages.
- *Abundance*, inspired by paralleled BFT where every node participates in each step [11], [13], [38], captures roles performed by more nodes than the adversary's fault capacity f.

We also prove that the safety guarantee of consensus protocols in their original model can be transferred into the MCAB model (Theorem 1 and Theorem 2).

Understanding design trade-offs. To further our understanding on concealment and abundance, we derive lower bounds for the minimum number of communication rounds and message complexity introduced by concealment or abundance (Theorem 5-7). We are then able to present a full picture of the trade-offs between concealed and abundant protocol designs using our lower bounds and applicable existing work.

Case studies with known protocols. We show that Algorand is safe and live in the MCAB adversary model, confirming the informal discussion that Algorand solves consensus against a stronger adversary than the adaptive Byzantine adversary. Hotstuff, on the other hand, loses liveness as it was strictly designed for the adaptive Byzantine adversary.

The rest of this paper is as follows. Section II introduces the system model. Section III formalizes the MCAB adversary model and proves that protocol's persistence is maintained against the MCAB adversary. Section IV analyzes necessary and sufficient properties for liveness. Before concluding our work, Section VI leverages our findings on two case studies, Algorand and Hotstuff.

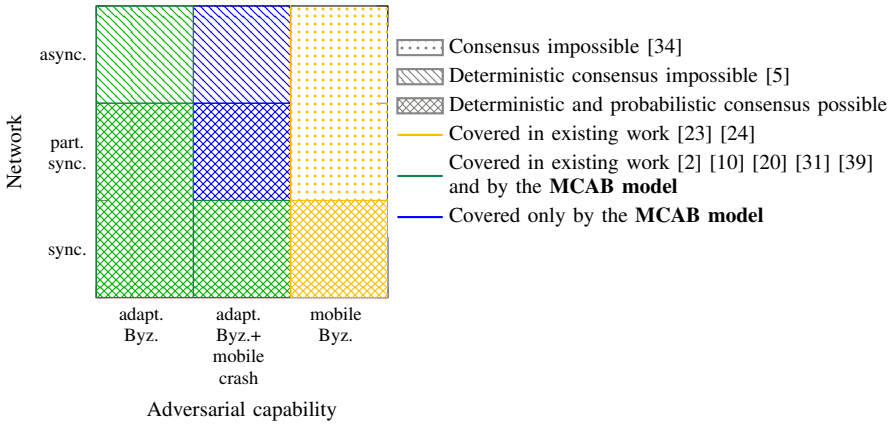

Fig. 1: Landscape of existing models (with more details in Section VII).

II. System model and preliminaries

This section introduces the system model, network assumptions, and preliminaries on consensus.

A. System model

The system consists of a set of n distributed participants, which we call *nodes*, running a pre-defined consensus protocol. In applications over the internet, they are usually online servers. We include scenarios where nodes represent fractions of stake or total hash power by considering the smallest units of stake or hash power as the equivalent to single nodes. For example, a participant controlling $1/3$ of stake can be modeled as $1/3$ of all individual nodes in the system.

Honest nodes follow the specified protocol and send or receive messages according to the network model. However, some nodes may be faulty, which is modelled as an *adversary* with access to different corruption oracles.

Section III defines the full range of adversarial capabilities we consider.

B. Network assumptions

Nodes are connected to each other in a network of point-to-point communication links. The three types of assumptions on message delivery are as follows:

Synchrony In synchronous networks, all messages sent between honest nodes are delivered within a known time bound Δ. The order in which they arrive can be determined by the adversary.

Asynchrony In an asynchronous network, message delivery is determined by the adversary, as long as all messages sent between honest nodes are eventually delivered. Messages between honest nodes cannot be dropped or modified.

Partial synchrony Partial synchrony behaves asynchronously until after an unknown time, Global Stabilization Time (GST), after which the network behaves synchronously with a known time bound Δ between honest nodes.

C. Consensus protocols

We first define the concept of communication steps and negligible functions and present the properties of consensus.

Communication steps To track time in the system, we use a *communication step* counter denoted with $s \in \mathbb{N}_0$ and initialized at 1. As considered in asynchronous networks to count rounds [13], all messages are assigned a virtual round number by the system, with the condition that messages with virtual round $s+1$ can only be sent once all messages with virtual round $s-1$ between honest nodes have been delivered. Note that virtual rounds are only used for analysis and not contained in concrete messages. The current communication step s is the highest assigned virtual round. In (partially) synchronous networks, the communication step s is also incremented once the network's delay bound Δ is reached.

Negligible function A negligible function refers to a function $negl(x)$ where for every $y \in \mathbb{N}$, there exists an $z \in \mathbb{N}$ such that $negl(x) < 1/x^y$ for all $x \geq z$.

Message complexity Message complexity is the expected total number of messages generated by honest nodes.

Consensus Blockchain consensus mainly considers two properties, namely *persistence* and *liveness* [20]. Intuitively, persistence guarantees that a valid entry in the ledger is consistent across nodes and time, while liveness ensures that new inputs can be added to the ledger.

Definition 1 (Persistence [20])**.** *A consensus protocol satisfies persistence if the following holds. At each communication step s, if an honest node reports a ledger that contains an input v in a block more than $k \geq 0$ blocks away from the end of the ledger, then v will be eventually always be reported in the same position in their ledger by all honest nodes for all communication steps $s' > s$ except with probability $negl(k)$.*

The block depth k starts at $k = 0$ for the most recent one. Persistence is achieved with probability 1 once a block reaches a pre-defined k. In contrast, the probability of persistence in Nakamoto consensus grows asymptotically close to 1 with k.

Definition 2 (t-Liveness [20])**.** *A protocol $\mathcal{P}$ achieves t-liveness if the following holds. At each communication step s, if a valid input v, given to all honest nodes during each communication steps s to $s+t(k)$, then v is reported as k deep by all honest nodes after communication step $s' > s+t(k)$, except with probability $negl(k)$. The number of communication steps t is a known polynomial function of k.*

III. Mobile crash adaptive Byzantine adversary model

This section defines adversarial capabilities, presents a framework for analyzing protocols with roles, and shows how the persistence of protocols against the adaptive Byzantine adversary can be transferred to the MCAB model.

A. Corruption oracles and fault mobility

A *corruption oracle* is an abstract black box representing the adversary's capabilities. It refers to either the Byzantine oracle in Definition 3 or the crash oracle in Definition 4.

Definition 3 (Byzantine oracle $\mathcal{O}_B$)**.** *The Byzantine oracle $\mathcal{O}_{B,f}$ takes queries with an input id and is parameterized by the corruption budget $f \in \{0, ..., n\}$. For simplicity, we denote $\mathcal{O}_{B,f}$ with $\mathcal{O}_B$. For a query at communication step s, if $|\mathcal{B}_s| < f$ for the set $\mathcal{B}_s$ of Byzantine nodes at step s, $\mathcal{O}_B$ adds id to $\mathcal{B}_s$, and outputs the internal state of node id to the adversary. Nodes in $\mathcal{B}_s$ can deviate arbitrarily from the protocol during s. If $|\mathcal{B}_s| \geq f$, $\mathcal{O}_B$ outputs $\perp$.*

Definition 4 (Crash oracle $\mathcal{O}_C$)**.** *The crash oracle $\mathcal{O}_{C,c}$ takes queries with an inputs id and is parameterized by by the corruption budget $c \in \{0, ..., n\}$. For simplicity, we denote $\mathcal{O}_{C,c}$ with $\mathcal{O}_C$. For a query at communication step s, if $|\mathcal{C}_s| < c$ for the set $\mathcal{C}_s$ of crashed nodes, $\mathcal{O}_C$ adds id to $\mathcal{C}_s$. Nodes in $\mathcal{C}_s$ keep their memory and follow the protocol, but outgoing and incoming messages may not deliver during communication step s. If $|\mathcal{C}_s| \geq c$, $\mathcal{O}_C$ outputs $\perp$.*

Blockchains traditionally use an adaptive adversary [8], [19], [20] that gradually corrupts nodes over time, reflected in our definition by allowing corruption queries at any communication step. Once corrupted, nodes remain under adversarial control. We define adaptive adversaries and obtain the adaptive Byzantine adversary $\mathcal{A}_{AB}$.

Definition 5 (Adaptive adversary)**.** *An adaptive adversary with a corruption oracle $\mathcal{O} \in \{\mathcal{O}_C, \mathcal{O}_B\}$ can query $\mathcal{O}$ any number of times at the start of any communication step s, before nodes send or receive messages during s. A node corrupted by an adaptive adversary at s is corrupted for all $s' > s$.*

Introduced in the context of traditional BFT [23], [24] and proactive secret sharing [26], [27], mobile adversaries also gradually corrupt nodes but can additionally make corrupted nodes honest again. As the protocol progresses, this lets the adversary corrupt more nodes in total while the number of faults at any time remains bounded. We formally define a mobile adversary in Definition 6.

Definition 6 (Mobile adversary)**.** *A mobile adversary with a corruption oracle $\mathcal{O} \in \{\mathcal{O}_C, \mathcal{O}_B\}$ can query $\mathcal{O}$ any number of times at the start of any communication step s, before any nodes send or receive messages. At the start of s, before querying $\mathcal{O}$, a mobile adversary is also able to* cure *a node it has corrupted so that it becomes honest again, and receives all messages previously sent to it.*

By limiting the oracle queries and curing of nodes at a communication step s before any nodes send or receive messages, the possible number of faulty nodes at any communication step s is not higher than the corruption budget of the adversary, as in Buhrman et al. [23]. For example, by replacing a faulty node during s with another one, both count as a fault during s.

Using the previous definitions, we can define the mobile crash fault adaptive Byzantine adversary $\mathcal{A}_{MCAB}$.

Definition 7 (Mobile crash adaptive Byzantine adversary)**.** *The mobile crash adaptive Byzantine adversary $\mathcal{A}_{MCAB}$ is mobile with the crash oracle $\mathcal{O}_c$ and corruption budget c, and is adaptive with the Byzantine corruption oracle $\mathcal{O}_B$ and corruption budget f.*

Nodes in the Byzantine set cannot be replaced or removed without some recovery mechanism for secret keys and authenticated communication channels. In contrast, crashed nodes go back online if they are not targeted again in communication step $s+1$ and receive all previous pending messages as in the sleepy model [31]. As a result $\mathcal{A}_{MCAB}$ has adaptive Byzantine and mobile crash faults.

B. Modelling roles in consensus

To analyze protocols in our proposed model, we follow and extend the idea of a role/node framework for general multi-party computation introduced in YOSO [35] for consensus. A protocol consists of atomic tasks, performed by different roles at every communication step. Roles, formalized in Definition 8, are then assigned to nodes with a role assignment mechanism as the protocol runs. Known protocols described in this framework are provided in Section VI.

Definition 8 (Role)**.** *Roles are a subset of the protocol's sequence of tasks, whose input values come from received messages and local memory and whose outputs are one or more messages communicated to one or more other participants and updating local memory.*

- *Role assignment: A role $\mathcal{R}$ is assigned at communication step s to a set of nodes, the committee $\mathcal{M}_s$, by a role assignment oracle $assignRoles_{\mathcal{R}}(s, \lambda)$ with security parameter λ.*
- *Role existence: A role $\mathcal{R}$ exists at a communication step s if the protocol requires at least one node to perform the role's tasks at s. If $\mathcal{R}$ does not exist at step s, then $\mathcal{M}_s = \emptyset$.*

C. Persistence in the MCAB model

We prove that protocols with persistence against $\mathcal{A}_{AB}$ can also achieve persistence against $\mathcal{A}_{MCAB}$.

Theorem 1 (Persistence against MCAB)**.** *If a protocol P achieves persistence against $\mathcal{A}_{AB}$ with f Byzantine corruptions, then P achieves persistence against $\mathcal{A}_{MCAB}$ with c mobile crash and $f-c$ Byzantine corruptions.*

Proof. We prove by contradiction. Consider a protocol P that achieves persistence against $\mathcal{A}_{AB}$, but not against $\mathcal{A}_{MCAB}$. This means that, with probability larger than $negl(k)$, a stable input $v' \neq v$ is reported at the same position as v in the ledger by an honest node at communication step s', where $s' > s$. The adversary may cause crashed nodes to send messages supporting this persistence violation, such that in the worst case, $\mathcal{A}_{MCAB}$ has $f-c+c$ nodes with Byzantine behaviour. $\mathcal{A}_{MCAB}$ must be able to create this persistence violation with probability greater than $negl(k)$ using only these $f-c+c$ nodes. This forms a contradiction as P achieves persistence with up to f Byzantine nodes, and $\mathcal{A}_{MCAB}$ must be able to create a persistence violation controlling only up to f nodes with Byzantine behaviour. □

IV. Liveness in the MCAB model

In this section, we explore security requirements for a protocol to maintain liveness against the MCAB adversary. We first show that protocols secure in an asynchronous network are always secure in the MCAB model. We then introduce and prove the necessary and sufficient conditions for liveness against the MCAB adversary.

Definitions in this section consider the following notations: A consensus protocol P with persistence and t-liveness against an adversary $\mathcal{A}_{AB}$ with access to the Byzantine oracle $\mathcal{O}_B$ and corruption budget f; An adversary $\mathcal{A}_{MCAB}$ with access to the crash oracle $\mathcal{O}_C$ with corruption budget c and to the Byzantine oracle $\mathcal{O}_B$ with corruption budget $f-c$; The set $\mathcal{N}_s$ of n_s nodes running P at a communication step s, identified by id where $id \in [1,...,n_s]$; The Byzantine set $\mathcal{B}_s$, the crashed set $\mathcal{C}_s$, and the honest set $\mathcal{H}_s$ where $\mathcal{H}_s = \mathcal{N}_s \setminus (\mathcal{B}_s \cup \mathcal{C}_s)$ at a communication step s; A role $\mathcal{R}$ in P, assigned to nodes in the set $\mathcal{M}_s$ at a communication step s; The set $\mathcal{S}_{\mathcal{R}} = \{\mathcal{R}_j | j \in [1,p]\}$ of all roles in P. Each $\mathcal{R}_j$ where $j \in [1,p]$ is assigned to nodes in the set $\mathcal{M}_{j,s}$ at a communication step s; The role assignment oracle for $\mathcal{R}$ denoted as $assignRoles_{\mathcal{R}}$.

A. Asynchronous networks

We prove that asynchronous BFT protocols are also secure in our model. In partial synchrony, $\mathcal{A}_{MCAB}$ is hence only different from $\mathcal{A}_{AB}$ when considering liveness in the synchronous period after GST.

Theorem 2 (Asynchronous BFT is secure against the MCAB adversary)**.** *If a protocol P achieves persistence and t-liveness (after t asynchronous rounds) in an asynchronous network against $\mathcal{A}_{AB}$ with f adaptive Byzantine corruptions, then it achieves persistence and t-liveness against $\mathcal{A}_{MCAB}$ in a synchronous network with c mobile crash corruptions and $f-c$ adaptive Byzantine corruptions.*

Proof. To show that persistence is maintained against $\mathcal{A}_{MCAB}$ by any protocol with persistence against $\mathcal{A}_{AB}$, see Theorem 1. We prove t-liveness is maintained by contradiction. Consider a protocol P with t-liveness in an asynchronous network against $\mathcal{A}_{AB}$, but not against $\mathcal{A}_{MCAB}$ in a synchronous network. There therefore exists a valid input v given to all honest nodes in $\mathcal{H}_s$ at communication step s, not considered stable for any nodes in $\mathcal{H}_{s'}$ for all communication steps $s' > s+t$. This means that the $n-f$ nodes in each set $\mathcal{H}_{s+i}$ where $i \in [0,t)$, who receive all previous honest messages and deliver new messages within a known delay Δ, don't achieve stability for v for at communication step $s+t$. However, P has t-liveness in an asynchronous network against $\mathcal{A}_{AB}$, and any input becomes stable after t asynchronous rounds between $n-f$ honest nodes. As t asynchronous rounds can always model t synchronous communication steps by definition, this is a contradiction. □

B. Properties for liveness

The rest of our analysis focuses on liveness in a synchronous network or the synchronous period of a partially synchronous setting when transferring protocols live against $\mathcal{A}_{AB}$ to $\mathcal{A}_{MCAB}$.

Inspired by protocols with secret block proposers [8], [19], [20] and protocols where every node participates at every step [11], [13], [38], we derive two properties using the role framework of Section III-B, abundance and concealment.

A (r,q,t)-essential role is first defined to exclude superfluous roles for the sake of counter-examples It is parameterized by the minimum number r of assignments to the role that need to be honestly executed at least q times out of t communication steps for the t-liveness of the protocol.

Informally, essential roles represent the steps in a protocol crucial to its proper execution, while non-essential roles could be skipped without consequence.

Definition 9 ((r,q,t)-essential)**.** *Every role $\mathcal{R}_j$ with $j \in [1,p]$ in the set $\mathcal{S}_{\mathcal{R}}$ is (r_j,q_j,t)-*essential *where $r_j, q_j \in \mathbb{N}$ if the following holds. At communication step s, for a valid input w given to all nodes in every $\mathcal{H}_{s+i}$ where $i \in [0,t)$ in P, if there exists $l \in [1,p]$ such that $|\mathcal{H}_{s+i} \cap \mathcal{M}_{l,s+i}| < r_l$ more than $t-q_l$ times in t communication steps $s+i$ where $i \in [0,t)$, then w is stable in the ledger after t communication steps with probability lower than $1-negl(t)$.*

Conversely, if for every $j \in [1,p]$, $|\mathcal{H}_{s+i} \cap \mathcal{M}_{j,s+i}| \geq r_j$ more than q_j times in t communication steps $s+i$ where $i \in [0,t)$, then w is stable in the ledger after t communication steps except with probability $negl(t)$.

We first define abundance, where the role is assigned to a sufficient number of nodes to tolerate all corruptions.

Definition 10 (Abundance)**.** *The (r,q,t)-essential role $\mathcal{R}$ is* abundant *if, for at least q out of any t communication steps $s+i$ where $i \in [0,t)$}, we have $|\mathcal{M}_{s+i}| \geq f+r$.*

The second property is concealment, where the identity of the assigned nodes is secret and $\mathcal{A}_{MCAB}$ can only randomly guess which nodes to target. This ensures that the adversary can't reliably stall the protocol.

Concealment is defined by considering the adversary's win probability in a game where the adversary tries to guess the nodes assigned to a role.

Definition 11 (Concealment)**.** *The non-abundant (r, q, t)-essential role $\mathcal{R}$ is concealed if it always wins the following experiment except with probability negligible in t. The* concealment *experiment between the (r, q, t)-essential role $\mathcal{R}$ and the adversary $\mathcal{A}_{MCAB}$ is defined as t consecutive runs of the following game in all communication steps $s + i$ for some s with $i \in [0, t)$. At the start of the game, up to $f - c$ nodes are in the Byzantine set $\mathcal{B}_{s+i}$ for $i \in [0, t)$.*

- *Learning phase: $\mathcal{A}_{MCAB}$ chooses any number of learning rounds. For each learning round, $\mathcal{A}_{MCAB}$ picks $l \notin [s, s+t)$, sends it to $\mathcal{R}$ who calls $assignRoles_{\mathcal{R}}(l, \lambda)$ to obtain $\mathcal{M}_l$, and sends $\mathcal{M}_l$ to $\mathcal{A}_{MCAB}$.*
- *Challenge phase: $\mathcal{A}_{MCAB}$ initiates the challenge phase, calling $\mathcal{O}_C$ any number of times and obtaining the set of crashed nodes $\mathcal{C}_{s+i}$. $\mathcal{R}$ obtains $\mathcal{M}_{s+i}$ with $assignRoles_{\mathcal{R}}(s + i, \lambda)$, and $\mathcal{A}_{MCAB}$ wins the game if $|\mathcal{M}_{s+i} \setminus (\mathcal{B}_{s+i} \cup \mathcal{C}_{s+i})| < r$, which we denote by $\mathbf{d}_i$. We also denote by $\mathbf{d}_i$ the indicator random variable for the event $\mathbf{d}_i$, which equals 1 if $\mathbf{d}_i$ occurs and 0 otherwise.*

If event $\mathbf{d}_i$ occurs for more than $t - q$ values of $i \in [0, t)$, $\mathcal{A}_{MCAB}$ wins the experiment, denoted by the event $\mathbf{D}$. Note that $\mathcal{A}_{MCAB}$ can add Byzantine nodes with $\mathcal{O}_B$ each communication step $s + i$ if $|\mathcal{B}_{s+i}| < f - c$.

The (r, q, t)-essential concealed role $\mathcal{R}$ is ideally-concealed *if, for all i where $\mathcal{R}$ exists, the indicator random variables $\mathbf{d}_i$ are i.i.d and $\Pr[\mathbf{d}_i] - \sum_{j=0}^{r} \binom{\mathcal{M}}{j} (\frac{f}{n})^j (1 - \frac{f}{n})^{r-j} < negl(\lambda)$.*

We define protected roles as roles that can't be reliably targeted by the additional capabilities of $\mathcal{A}_{MCAB}$ by being abundant or concealed.

Definition 12 (Protected)**.** *The (r, q, t)-essential role $\mathcal{R}$ is protected if it is abundant or concealed.*

C. Necessity and sufficiency for liveness

We show that each role of a protocol being protected is sufficient and necessary to remain secure against $\mathcal{A}_{MCAB}$.

With Lemma 1 we show the necessity. Intuitively this is due to the adversary being able to target all roles that are not protected with mobile crashes.

Lemma 1 (Necessity for liveness)**.** *For any consensus protocol P, t-liveness against $\mathcal{A}_{MCAB}$ is impossible if there exists one (r, q, t)-essential role that is not protected.*

Proof. We prove by contradiction. Say an (r_j, q_j, t)-essential role $\mathcal{R}_j$ for P is not protected but P achieves t-liveness against $\mathcal{A}_{MCAB}$.

As $\mathcal{R}$ is not protected, for t communication steps $s+i$ where $i \in [0, t)$, the event $(|\mathcal{H}_{s+i} \cap \mathcal{M}_{j,s+i}| < r_j)$ can happen more than $t - q_j$ times with probability larger than $negl(t)$. Due to the (r_j, q_j, t)-essential property of $\mathcal{R}_j$, this means that a valid input w given to every $\mathcal{H}_{s+i}$ where $i \in [0, t)$ is not stable in the ledger after t communication steps with probability larger than $negl(t)$. This contradicts the assumed t-liveness. □

Lemma 2 shows the sufficiency of protected roles for liveness.

Lemma 2 (Sufficiency for liveness)**.** *A consensus protocol P with persistence and t-liveness against $\mathcal{A}_{AB}$ always achieves t-liveness in $\mathcal{A}_{MCAB}$ if all node actions are implemented through protected (r, q, t)-essential roles and their respective role assignment mechanism.*

Proof. We prove by contradiction. Let P be a consensus protocol with persistence and t-liveness for $\mathcal{A}_{AB}$ but not for $\mathcal{A}_{MCAB}$. Let $\mathcal{S}_{\mathcal{R}} = \{\mathcal{R}_j | j \in [1, p]\}$ be the set of all p (r_j, q_j, t)-essential roles in P, which are protected, and let $\mathcal{H}_s$ be the set of honest nodes at communication step s. For every role $\{\mathcal{R}_j \in \mathcal{S}_{\mathcal{R}} | j \in [1, p]\}$, consider the following.

Since $\mathcal{R}_j$ is protected, in t communication steps $s+i$ where $i \in [0, t)$, $|\mathcal{H}_{s+i} \cap \mathcal{M}_{j,s+i}| \geq r_j$ occurs at least q_j times with probability $1 - negl(t)$. By the r_j, q_j, t-essential property of every $\mathcal{R}_j, j \in [1, p]$, any valid input v given to every $\mathcal{H}_{s+i}$ where $i \in [0, t)$ is stable in the ledger after t communication steps except with probability negligible in t. By definition of t-liveness, P also has t-liveness in $\mathcal{A}_{MCAB}$, a contradiction. □

Finally we show that, for protocols to maintain their security properties against $\mathcal{A}_{MCAB}$, their roles must be protected.

Theorem 3 (Necessary and sufficient for liveness)**.** *Consider a consensus protocol P that achieves t-liveness against $\mathcal{A}_{AB}$. For P to have t-liveness against $\mathcal{A}_{MCAB}$, it is necessary and sufficient that all node actions can be defined through protected roles and their respective role assignment mechanism.*

Proof. This statement follows from Lemma 2 and Lemma 1. □

V. Performance trade-offs in the MCAB model

In this section we derive bounds and leverage existing works to analyze the trade-offs between abundance and concealment, summarized in Table I.

A. Lower bounds

The lower bound on t-liveness due to an abundant role follows from Definition 9.

Theorem 4 (Abundance lower bound on t-liveness)**.** *For a consensus protocol P_a with t_a-liveness against $\mathcal{A}_{MCAB}$ containing an abundant (r_a, q, t_a)-essential role $\mathcal{R}_a$, we always have $t_a \geq q$.*

Proof. Due to the (r_a, q, t_a)-essential property as defined in Definition 9, t_a-liveness requires at least $t_a \geq q$. □

We show that using an abundant essential role requires at least $\Omega(n)$ message complexity, due to Definition 10. When every role is abundant, each node performing a role sends its message to the nodes assigned to the next abundant role, obtaining $\Omega(n^2)$ message complexity.

Theorem 5 (Abundance lower bound on message complexity)**.** *The minimum message complexity of a consensus protocol P_a achieving t_a-liveness against $\mathcal{A}_{MCAB}$ containing an abundant (r_a, q, t_a)-essential role $\mathcal{R}_a$, which sends at least one message, is $\Omega(n)$. If every role in P_a is abundant, the message complexity of P_a is $\Omega(n^2)$.*

Proof. We prove by contradiction. Say an (r_a, q, t_a)-essential role $\mathcal{R}_a$, which sends one message in a consensus protocol P_a, is abundant but the message complexity of P_a is less than $\Omega(n)$. For $\mathcal{R}_a$, we have $|\mathcal{M}| \geq \ f + r_a$ at least q times in t_a communication steps since it is abundant. Each node in $\mathcal{M}$ sends at least one message by definition. At least $f + r_a$ messages are therefore created in communication steps where $\mathcal{R}_a$ exists. As f scales with n, this implies $\Omega(n)$ message complexity, a contradiction.

We prove by contradiction if every role in P_a is abundant. Say an (r_a, q, t_a)-essential role $\mathcal{R}_a$, which sends messages to the next abundant $(r_{a'}, q', t_{a'})$-essential role $\mathcal{R}_{a'}$ in a consensus protocol P_a, is abundant but the message complexity of P_a is less than $\Omega(n^2)$. For $\mathcal{R}_a$, we have $|\mathcal{M}| \geq \ f + r_a|$ at least q times in t_a communication steps since it is abundant. Each node in $\mathcal{M}$ sends at least $\Omega(n)$ messages to the $|\mathcal{M}'| \geq \ f + r_{a'}|$ nodes assigned to $\mathcal{R}_{a'}$. At least $(f + r_a)(f + r_{a'})$ messages are sent in communication steps where $\mathcal{R}_a$ exists and is followed by $\mathcal{R}_{a'}$. As f scales with n, this implies $\Omega(n^2)$ message complexity, a contradiction. □

To lower bound t-liveness when introducing a concealed role, we use the Chernoff bound to obtain t given its parameters and the failure probability. Note that the bound on t is optimistic due to our looser approximations of the Binomial distribution tail bound, and a tighter bound could give a slightly higher t. If $\epsilon \to 0$, $t \to \infty$, and if $p = 0$, $t \geq q$ matching the abundant case.

Theorem 6 (Concealment lower bound on t-liveness)**.** *For a consensus protocol P_c achieving t-liveness with failure probability ϵ containing a concealed (r, q, t)-essential role $\mathcal{R}_c$, we always have $t \geq q + \nu(q, \epsilon)$ where $\nu(q, \epsilon) > 0$ and ϵ is the liveness error probability.*

Proof. For a protocol using $\mathcal{R}_c$, t-liveness has a failure probability ϵ. To get the lowest t, $\mathcal{R}_c$ is ideally-concealed and for all $i \in [0, t)$, we denote the i.i.d. $\Pr[\mathbf{d}_i]$ as p from the game in Definition 12. We can therefore consider a Binomial distribution with t trials, p as the success probability, and X as the number of successes. We use the tail bound on Binomial distributions [40, Lemma 4.7.2] to obtain a lower bound on t, parameterized by ϵ, p, and q. Using the tail bound to bound the maximal number of successes $t - q$, we get

$$\Pr[X > t - q] = \epsilon \geq \frac{1}{\sqrt{8t\lambda(1-\lambda)}} 2^{-tB(\lambda,p)} \geq \frac{1}{\sqrt{2t}} 2^{-tB(\lambda,p)}.$$

where $\lambda = \frac{t-q}{t}$ and $B(\lambda, p) = \lambda \log \frac{\lambda}{p} + (1-\lambda) \log \frac{1-\lambda}{1-p}$. Using the inequality $\log x \leq x - 1$ where $x \geq 0$, we loosen the bound and derive the following inequality :

$$\log \frac{1}{\epsilon} \leq \frac{1}{2} \log(2t) + tB(\lambda, p)$$
$$0 \leq \frac{t^2}{p} + t(\log \epsilon - \frac{2q}{p} - 1) + \frac{q^2}{1-p} + \frac{q^2}{p}$$

Solving for t assuming $t > 0$ we obtain the lower bound:

$$t \geq p \log \tfrac{1}{\epsilon} + q + \tfrac{p}{2}(1 + \sqrt{(\log \epsilon - \tfrac{2}{p} - 1)^2 - \tfrac{4q^2}{p}(\tfrac{1}{1-p} + \tfrac{1}{p})})$$

Since $q > 0$, and $p, \epsilon \in [0, 1]$, the terms are all positive.

We therefore have $t \geq q + \nu(q, \epsilon)$ where $\nu(q, \epsilon) > 0$ and ϵ is the liveness error probability. □

As a concealed role could be assigned to only one node, the lower bound on message complexity comes from the requirement that all $n - f$ honest nodes receive the value to decide upon, implying at least $\Omega(n)$ message complexity.

Theorem 7 (Concealment lower bound: message complexity)**.** *The minimum message complexity of a consensus protocol P_c amongst n nodes and f faults achieving t-liveness containing a concealed (r, q, t)-essential role $\mathcal{R}_c$ is $\Omega(n)$.*

Proof. We have $|\mathcal{M}| < f + r$ all communication steps and $|\mathcal{M} \setminus (\mathcal{B} \cup \mathcal{C})| \geq r$ at least q times in t communication steps. As a result, there are at least r messages sent in the communication steps where $\mathcal{R}_c$ exists. However by definition, all $n - f$ honest nodes must receive the input reported in the same position in the ledger, requiring at least $n - f$ messages for P_c to achieve persistence. These $n - f$ messages must be sent by a role in P_c and its message complexity is $\Omega(n)$. □

B. Discussion on trade-offs

Communication complexity and rounds It is proven that deterministic consensus requires at least $f + 1$ rounds in the worst case with f adaptive crash faults [41], and $O(n^2)$ communication complexity [42]. These results apply to deterministic fully abundant protocols. In contrast, with randomised consensus, the number of rounds to reach an agreement is probabilistic and is dependent on the error probability as a system parameter (Theorem 6).

A protocol using concealment only requires $O(n)$ message complexity (Theorem 7), enabling subquadratic communication complexity similarly to Algorand [43] and Abraham et al. [44].

After-the-fact-removal However, as shown by Abraham et al. [44], subquadratic complexity is only possible without after-the-fact-removal, which means that the adversary is not able to stop messages already sent by a node it corrupts. With after-the-fact-removal, the adversary is able to "cheat" the game defined in Definition 11 by modifying it such that in the learning phase, it can pick the challenge phase's communication steps. This implies that no role can be concealed, allowing only abundance. The communication complexity lower bound $O(n^2)$ with after-the-fact-removal [44] then matches our derived bound in Theorem 5.

Consider the additional cases where after-the-fact removal is allowed for one type of fault only:

- After-the-fact removal adaptive Byzantine faults only, not mobile crashes: while the adversary hasn't selected all adaptive Byzantine faults, a protocol with subquadratic communication complexity cannot achieve consensus [44]. However,if the adversary has selected its maximum allowed number of adaptive faults, the adversary has no after-the-fact removal ability anymore, and concealment is possible to maintain liveness.
- After-the-fact removal mobile crashes only, not adaptive Byzantine: After-the-fact-removal mobile crashes can perpetually target concealed committees, and only abundance is possible.

Sources of randomness While the implementation of abundant roles has no additional overhead, concealed roles require a mechanism to secretly assign roles to nodes. This line of research has notable examples such as PoW mining [20], Verifiable Random Function (VRF) selection [45], and Single Secret Leader Election (SSLE) [46]. When the the size $|\mathcal{M}|$ of the role's committee is random, rounds where not enough nodes are assigned may be wasted, but achieving deterministic $|\mathcal{M}|$ is expensive.

Probabilistic $|\mathcal{M}|$. A role assignment with probabilistic $|\mathcal{M}|$ assigns a number of nodes sampled from a certain probability distribution, usually such that $\mathbb{E}[|\mathcal{M}|] > r$. Since $|\mathcal{M}|$ is probabilistic, it is still possible to occasionally have $|\mathcal{M}| < r$ without interference from the adversary. This can add additional communication rounds to terminate the protocol, leading to higher values of t for the t-liveness guarantee. For example, VRF random assignments such as in Algorand [8] secretly select nodes, satisfying concealment, with a probabilistic expected number of nodes in one committee. It can be the case however, that a committee in Algorand does not reach quorum due to not enough nodes being selected.

Deterministic $|\mathcal{M}|$. A role assignment with deterministic $|\mathcal{M}|$ assigns a fixed number of randomly chosen nodes. Since $|\mathcal{M}| \geq r$ is guaranteed, only by winning the game defined in Definition 12 can $|\mathcal{H} \cap \mathcal{M}| < r$ still happen. SSLE introduces a primitive to achieve such a design.

However, in the only existing protocol with adaptive security to our knowledge [47], SSLE runs with additional communication between nodes which has at least $O(n^2)$ message complexity for n synchronous broadcasts. It should also maintain its security properties in the MCAB model, which seems possible at first glance with similar arguments as in Theorem 1 and due to the abundant all-to-all communications.

Random beacon. Concealed roles inherently require a trusted source of randomness for their role assignment. This has been achieved in various ways like an initial trusted nonce for Bitcoin [20] or Algorand [8] that is updated as the protocol runs. Distributed random beacons that uses either consensus [48] or trusted setups [49] are also a way to agree on a common source of randomness. However, they may leak information, increasing $\Pr[\mathbf{d}]$ in Definition 12 and t for t-liveness which must be analyzed properly.

Previous work has also shown the benefit of load balancing [38], [50], [51], which distributes computation and communication workload to nodes, avoiding performance bottlenecks and obtain higher throughput akin to abundant roles.

TABLE II: Comparison of abundant and concealed roles. We consider protected (q, r, t)-essential roles with the positive factors $\mu(\epsilon), \nu(q, \epsilon) > 0$ depending on q and the acceptable error probability for liveness ϵ.

	t-liveness bound	Message complexity	No random beacon	Allow after--the-fact--removal	Load balancing
Abundant and deterministic	$t \geq f + 1$	$\Theta(n^2)$	✓	✓	✓
Abundant and randomized	$t \geq q + \mu(\epsilon)$	$\Theta(n^2)$	✗	✓	✓
Concealed with determin. $\lvert\mathcal{M}\rvert$	$t \geq q + \nu(q, \epsilon)$	$O(n^2)$	✗	✗	✗
Concealed with random $\lvert\mathcal{M}\rvert$	$t \geq q + \nu(q, \epsilon)$	$\Theta(n)$	✗	✗	✗

Table V-B summarizes the performance trade-offs between the variants of abundance and concealment.

VI. Case studies: existing protocols in the MCAB model

This section leverages our theorems to analyze the security of Ouroboros [36], [52], Algorand [8], [43] and Chained Hotstuff [4] against $\mathcal{A}_{MCAB}$. We briefly summarize their protocol flow and show Chained Hotsuff's lack of liveness and Algorand's security against $\mathcal{A}_{MCAB}$. Their respective full papers provide formal descriptions and proofs of security against $\mathcal{A}_{AB}$. We additionally present a modified illustrative version of Chained Hotstuff maintaining its liveness against $\mathcal{A}_{MCAB}$ with abundance, and discuss how to leverage existing work on fallback protocols for $\mathcal{A}_{MCAB}$.

A. Ouroboros Praos

Ouroboros Praos is an adaptively secure PoS protocol where nodes regularly compute a VRF output to check eligibility to be a block proposer. The chain selection rule followed by honest nodes allows them to converge to a consistent view of the blockchain.

Only one role is needed to describe Ouroboros Praos, the slot leader role where a node extends its view of the canonical chain with a new block and broadcasts it to the network. Arguments for Ouroboros' security in the MCAB model are analogous for the adaptively secure Snow White and other similar protocols with a single leader role selected randomly and secretly, for example using a VRF or PoW lottery.

Roles in Ouroboros Praos

- Slot leader role $\mathcal{R}_{sl}$: Create a block of transactions and gossip it to the network. The r value is 1.

Committees of roles

- Committee of role $\mathcal{R}_{sl}$: contains a random number of slot leaders, in the deployed protocol [9], we have expected value $\mathbb{E}[\tau_{sl}] = 0.05$.

Role assignment

The role assignment oracle $assignRoles$ in Ouroboros Praos is implemented via the VRF selection mechanism. Nodes are selected to be slot leaders with a parameterized probability weighted by stake. The inputs to the VRF check are their secret key, their number of coins, the output of the on-chain random beacon, and the τ parameter determining the expected number of selected nodes.

To prove the security of Ouroboros Praos against $\mathcal{A}_{MCAB}$, we simply show that the role in Ouroboros Praos satisfies the concealment requirements of Definition 11, implying security against $\mathcal{A}_{MCAB}$ by Theorem 3 and Theorem 1.

Corollary 1 (Security of Ouroboros Praos)**.** *Ouroboros Praos achieves t-liveness and persistence against $\mathcal{A}_{MCAB}$ with $f + c \leq n/2$ where f is the number of adaptive Byzantine faults, c is the number of mobile crash faults, and n is the number of total nodes.*

Proof. Role $\mathcal{R}_{sl}$ is concealed since the assigned nodes are not public until the blocks are broadcast, and the probability of $\mathcal{A}_{MCAB}$ guessing all proposers is smaller than 1. As per the full analysis of Ouroboros Praos [52], the probability that the adversary influences the source of randomness drop exponentially with the number of honest blocks in an epoch, and therefore adds a probability negligible in the security parameter for the adversary to guess assignments. This means $\mathcal{R}_{sl}$ is even ideally-concealed. □

B. Algorand

Nodes in Algorand [8] regularly obtain a pseudorandom number with a Verifiable Random Functions (VRF) whose randomness is verifiable with the node's public key. Computing the VRF output is done locally, allowing nodes to check eligibility to propose blocks and participate in BA⋆ [8], a Byzantine Agreement algorithm, without revealing their identity. Once block proposers broadcast a block, nodes are selected for each step of BA⋆ to eventually finalize the block with the lowest VRF output, or an empty block in bad network conditions.

We obtain three roles comprising Algorand: proposing a block, performing a step in its own Byzantine Agreement algorithm BA⋆, and finalizing a block in the last step of BA⋆.

Roles in Algorand

- Block proposer role $\mathcal{R}_{\texttt{PROPOSER}}$: Create a block of transactions and gossip it to the network. The r value is 1.
- Voter in the binary BA⋆ algorithm, step $i \in [1, \texttt{MaxSteps}]$ $\mathcal{R}^i_{\texttt{STEP}}$: The CommitteeVote function checks if one is selected for the next step's role, and is part of the role assignment mechanism of Algorand. The r for $\mathcal{R}_{\texttt{PROPOSER}}$ is $T_{\texttt{STEP}} \cdot \tau_{\texttt{STEP}} = 0.685 \cdot 2000 = 1370$. We refer to Algorithm 8 in [8] for the specific operations performed in each step i of BA⋆.
- Voter in the binary BA⋆ algorithm, final step $\mathcal{R}_{\texttt{FINAL}}$: Perform final step of BA⋆. The r for $\mathcal{R}_{\texttt{FINAL}}$ is $T_{\texttt{FINAL}} \cdot \tau_{\texttt{FINAL}} = 0.74 \cdot 10000 = 7400$.

Since the parameters for the role assignment mechanism and the voting threshold per BA⋆ step are the same, we analyze them together indexed with i. The last step denoted FINAL has slightly different parameters and is therefore considered separately.

Committees of roles

- Committee of role $\mathcal{R}_{\texttt{PROPOSER}}$: contains a random number of block proposers with expected value $\mathbb{E}[\tau_{\texttt{PROPOSER}}] = 26$.
- Committee of role $\mathcal{R}^j_{\texttt{STEP}}$: contains a random number of BA⋆ members for BA⋆'s step j with expected value $\mathbb{E}[\tau_{\texttt{STEP}}] = 2000$.
- Committee of role $\mathcal{R}_{\texttt{FINAL}}$: contains a random number of final BA members with expected value $\mathbb{E}[\tau_{\texttt{FINAL}}] = 10000$.

Role assignment

The role assignment oracle $assignRoles$ in Algorand is implemented via the VRF selection mechanism. Nodes are selected for different roles with a parameterized probability weighted by stake. The inputs to the VRF check are their secret key, their number of coins, the current on-chain seed, and the τ parameter determining the expected number of selected nodes.

To prove the security of Algorand against $\mathcal{A}_{MCAB}$, we simply show that every role in Algorand satisfies the concealment requirements of Definition 11.

Corollary 2 (Security of Algorand)**.** *Algorand achieves t-liveness and persistence against $\mathcal{A}_{MCAB}$ with $f + c \leq n/3$ for f adaptive Byzantine faults, c mobile crash faults, and n total nodes at any time.*

Proof. $\mathcal{R}_{\texttt{PROPOSER}}$ is concealed since the assigned nodes are not public until the blocks are broadcast, and the probability of $\mathcal{A}_{MCAB}$ randomly guessing all proposers is smaller than 1. As per the full version of Algorand [8], probability of guessing the next selection seeds drop exponentially with the number of blocks committed in a synchronous period, and therefore adds a probability negligible in the security parameter for the adversary to guess assignments. This means $\mathcal{R}_{\texttt{PROPOSER}}$ is even ideally-concealed.

$\mathcal{R}^j_{\texttt{STEP}}$ is concealed for all j since the assigned nodes aren't public until the votes are broadcast, and the probability of $\mathcal{A}_{MCAB}$ guessing committee members for any s such that $\mathcal{R}_{\texttt{STEP}} \cap \mathcal{H}_s < T_{\texttt{STEP}} \cdot \tau_{\texttt{STEP}} = 1370$ is smaller than 1. As for $\mathcal{R}_{\texttt{PROPOSER}}$, $\forall j, \mathcal{R}^j_{\texttt{STEP}}$ is ideally concealed.

$\mathcal{R}_{\texttt{FINAL}}$ is concealed since the assigned nodes aren't public until the votes are broadcast, and the probability of $\mathcal{A}_{MCAB}$ guessing committee members for any s such that $\mathcal{R}_{\texttt{FINAL}} \cap \mathcal{H}_s < T_{\texttt{FINAL}} \cdot \tau_{\texttt{FINAL}} = 7400$ is smaller than 1. As for $\mathcal{R}_{\texttt{PROPOSER}}$, $\mathcal{R}_{\texttt{FINAL}}$ is ideally concealed.

All roles in Algorand are concealed, and by Lemma 3, Algorand achieves t-liveness and persistence in $\mathcal{A}_{MCAB}$. □

C. Chained Hotstuff

Chained Hotstuff's [4] protocol flow is as follows: a leader collects signatures on the last proposed block and forms a Quorum Certificate (QC) if the threshold is met. It broadcasts the QC along with its own proposed block, which nodes verify before sharing their signature with the next leader. This process

repeats and, to guarantee safety and liveness, blocks need three sequential QCs to be finalized. If a threshold of nodes don't receive a leader's message before a timeout, the next leader is called through a view-change.

Chained Hotstuff [4] can be described in two roles, the leader and voter roles. The leader role is first assigned to one node selected in a round-robin fashion, followed by the voter role assigned to every node.

Roles in Chained Hotstuff

- Voter role $\mathcal{R}_v$: Wait for message from current communication step's leader. If the SAFENODE($b^*, b^*.justify$) predicate passes, send a vote to the next leader. Decide for blocks with a three-chain, commit on blocks with a two-chain, and pre-commit blocks with a one-chain. If no message is received from the current communication step leader, send a NEXTVIEW(m)essage to the next leader. Its r value is $2f + 1$.
- Leader role $\mathcal{R}_l$: Wait for all messages until there are $n-f$ votes and form QC with the partial signatures. Extend the highest received QC with a new leaf, containing the new input, and broadcast it. Its r value is 1.

Commitees of roles

- Committee of role $\mathcal{R}_l$: contains only one node, that communication step's leader.
- Committee of role $\mathcal{R}_v$: contains n nodes, with the same voting rules.

Role assignment

For $\mathcal{R}_l$, $assignRoles_{\mathcal{R}_l}(s)$ outputs a single node in a round robin fashion. For $\mathcal{R}_v$, $assignRoles_{\mathcal{R}_v}(s)$ outputs every node that is assigned one vote each.

The leader role is neither abundant nor concealed, preventing Chained Hotstuff's liveness against $\mathcal{A}_{MCAB}$ by Theorem 3. It is easy to see that liveness is not guaranteed against $\mathcal{A}_{MCAB}$ in streamlined BFT protocols with single leaders, for example Tendermint [1], Damysus [3], and Jolteon [53].

Corollary 3 (Security of Chained Hotstuff)**.** *Chained Hotstuff does not achieve t-liveness against $\mathcal{A}_{MCAB}$ with $f + c \leq n/3$ for f adaptive Byzantine faults, c mobile crash faults, and n total nodes at any time.*

Proof. The role $\mathcal{R}_l$ is not abundant since $|\mathcal{R}_l| = 1 \leq c + 1$. In addition, $\mathcal{R}_l$ is not concealed since, with the public round-robin schedule, $\mathcal{A}_{MCAB}$ can win the protected game with probability 1 by choosing to crash the next publicly known leader. By Lemma 1, Chained Hotstuff does not achieve t-liveness against $\mathcal{A}_{MCAB}$. □

D. Adding abundance: Abundant Chained Hotstuff

By running n Chained Hotstuff instances in parallel whose round-robin leader schedules don't overlap, we obtain the *Abundant Chained Hotstuff* protocol where every role is abundant, live against $\mathcal{A}_{MCAB}$. This modified protocol does not necessarily perform better practically and simply illustrates the use of abundance to maintain security against $\mathcal{A}_{MCAB}$.

We use Hotstuff-2's [54] improvements where only three consecutive honest leaders are required to finalize blocks, while maintaining optimistic responsiveness. To handle conflicts between instances, each instance's blocks of the same height are combined according to the instance number. This necessitates a finalized block in an instance to wait for every other block of the same height from other instances. To maintain progress at each instance in spite of faulty leaders, honest leaders extend every instance's highest QC, instead of only its own QC.

Roles in Abundant Chained Hotstuff:

- Voter role $\mathcal{R}_v$: Wait for proposals from the n instances at communication step s or after waiting for Δ. For each i, if the SAFENODE($(b_i^*, b_i^*.justify)$) predicate passes, send a vote to instance i's next leader. If no message is received or the SAFENODE($(b_i^*, b_i^*.justify$) predicate fails, send a nextView message to instance i's next leader. Once every instance's block for the same height have a two-chain, decide for these blocks, and commit on blocks with a one-chain. There are n messages in total, either a vote or a nextView, which are sent once n messages are received, or after timeout. The r value of $\mathcal{R}_v$ is $2(f + c) + 1$.
- Leader role $\mathcal{R}_l$: Wait for all messages until there are $2(f + c)+1$ votes and form QC with the partial signatures. Wait for up to a timeout Δ to receive the highest QC for every instance, extend them with a new leaf, containing the new input, and broadcast it. The r value of $\mathcal{R}_l$ is $f + 2c + 1$ to obtain 2 consecutive honest leaders on one instance.

Committees of roles

- Committee of role $\mathcal{R}_l$: contains n nodes, that communication step's leaders.
- Committee of role $\mathcal{R}_v$: contains n nodes.

Role assignment

For $\mathcal{R}_l$, $assignRoles_{\mathcal{R}_l}(s)$ assigns an instance to each node, rotating in round robin. For $\mathcal{R}_v$, $assignRoles_{\mathcal{R}_v}(s)$ assigns every node to vote for every proposals.

To show that Abundant Chained Hotstuff achieves persistence, we first show that Abundant Chained Hotstuff has persistence against $\mathcal{A}_{AB}$. We analyze all cases for two conflicting decisions v and w and show that each case leads to a contradiction. We then use Theorem 1 to show persistence against $\mathcal{A}_{MCAB}$.

Corollary 4 (Persistence of Abundant Chained Hotstuff)**.** *Abundant Chained Hotstuff achieves persistence against $\mathcal{A}_{MCAB}$ with $n > 3(f + c)$ nodes for f adaptive Byzantine faults and c mobile crash faults at any time.*

Proof. We first show that Abundant Chained Hotstuff has persistence against $\mathcal{A}_{AB}$.

Proof by contradiction: Assume two conflicting proposals v and w are delivered at communication steps s_v and s_w respectively. We analyze the three cases for this assumption, each leading to a contradiction.

- Case 1: an individual Chained Hotstuff instance in the protocol delivered v and w. This is a contradiction as two-chain Chained Hotstuff is safe against $\mathcal{A}_{AB}$ as shown in the original work [4], and can't deliver v and w.

- Case 2: v and w are delivered by separate Chained Hotstuff instances, and $s_v = s_w$. Each instance's proposal has received a two-chain, at which point every honest node will discard conflicting proposals. Only v or w is therefore delivered, a contradiction.
- Case 3: v and w are delivered by separate Chained Hotstuff instances and, w.l.o.g, $s_v < s_w$. To deliver w, w requires at least $2f + 1$ votes in one step. However, to reach $2f + 1$ votes, at least one honest node has voted for both v and w, a contradiction.

By Theorem 1, persistence is maintained against $\mathcal{A}_{MCAB}$. □

For liveness, we use the liveness against of $\mathcal{A}_{AB}$ of Hotstuff-2 [54] to show that individual Abundant Chained Hotstuff instances have liveness. We then show that every role is abundant, implying liveness against $\mathcal{A}_{MCAB}$ by Theorem 3.

Corollary 5 (Liveness of Abundant Chained Hotstuff)**.** *Abundant Chained Hotstuff achieves liveness against $\mathcal{A}_{MCAB}$ with $n > 3(f + c)$ nodes for f adaptive Byzantine faults and c mobile crash faults in a partially synchronous network.*

Proof. Hotstuff-2 has liveness after GST against $\mathcal{A}_{AB}$ as shown in the original paper [54]. For its chained version, it is known that liveness additionally requires three consecutive honest leaders to commit a proposal thanks to previous work on Chained Hotstuff [4], [55]. After GST when $n > 3f$ with round-robin leaders, three consecutive honest leaders are eventually obtained and Chained Hotstuff-2 also has liveness after GST against $\mathcal{A}_{AB}$. In Abundant Chained Hotstuff, each instance has round-robin leaders and obtains three consecutive honest leaders after GST against $\mathcal{A}_{AB}$.

To show that Abundant Chained Hotstuff maintains liveness against $\mathcal{A}_{MCAB}$, we show that both roles are abundant. $\mathcal{R}_v$ is abundant since $|\mathcal{R}_v| = n \geq\ c + 2f + 1$ every communication step where it is assigned. For $\mathcal{R}_l$ to be abundant, $|\mathcal{R}_l|$ must be large enough so that at least one instance is able to have three consecutive honest leaders. In three leader phases, $\mathcal{A}_{MCAB}$ is able to mobile crash the leader of $3c$ instances, while $3f$ instances may have a Byzantine leader. We therefore require $3f + 3c + 1$ leader roles assigned to distinct nodes. $\mathcal{R}_l$ is abundant since $|\mathcal{R}_l| = 3f + 3c + 1 \geq\ 3f + 3c + 1$ every communication step where it is assigned. By Theorem 3, Abundant Chained Hotstuff has persistence in $\mathcal{A}_{MCAB}$. □

E. Extensions and future directions

Generic frameworks for asynchronous fallback protocols [53], [56] may be modified for the MCAB setting, such that a protocol can switch between a protocol without concealment or abundance, vulnerable to DoS attacks, and a protocol secure against $\mathcal{A}_{MCAB}$. Knowing how to efficiently switch modes involves subtleties explored for the asynchronous case, for MCAB fallback we may follow the general framework [56], but dedicated redesign might be needed. For example, differentiating between simple leader failures, mobile DoS attacks, and network failures requires care.

The ACE framework [57] has a similar design to Abundant Hotstuff as it runs multiple instances of leader-based protocols in parallel, however it is designed to add asynchronous liveness while Abundant Hotstuff is secure in a partially synchronous network against $\mathcal{A}_{MCAB}$.

The condition of abundance or concealment may be a starting point to generalize Byzantine quorum systems' [58] availability property with parameters r and q. Instead of requiring the number of honest roles in a committee to be a full quorum, only r are required, and it must hold only for at least q communication steps in t, as opposed to at all steps. Links between Byzantine quorum systems [58] and conditions in the MCAB model could be further explored.

Optimal resilience bounds in the MCAB are still open. In a synchronous network where $n > 2f + c$ with MCAB, or the sleepy model, no known deterministic protocol exists to our knowledge. In partial synchrony, Abundant Chained Hotstuff is an example of a deterministic protocol with $n > 3(f + c)$. When the crashes are static and not mobile, $n > 3f + 2c$ has been achieved [15].

VII. Existing adversary models

Compared to existing models, only our model allows consensus protocols with the following properties, which we explain in the following paragraphs and summarize in Table I: capturing mobile DoS attacks, allowing deterministic consensus, allowing partition tolerant protocols, and requiring no identity re-establishment.

A. Capturing mobile DoS

To model the mobile DoS attack, the adversary always needs the ability to target a new node. This is not the case when the adversary eventually can't target certain honest nodes. For example, the classical adaptive adversary can't corrupt new nodes after f corruptions. Network models that eventually deliver all messages similarly can't model mobile DoS.

The mobile Byzantine adversary in some works [2], [26], [27], [59], which we call slow, compromises new sets of nodes periodically, such that the adversary can't corrupt new nodes after f corruptions within one period.

B. Deterministic consensus

As shown by the FLP impossibility result [5], deterministic consensus is impossible in a fully asynchronous network. In the sleepy model [31], deterministic protocols remain an open question. As a result, using asynchronous BFT's or the sleepy model's line of work may be unsuitable for applications requiring deterministic guarantees.

C. Requiring no identity re-establishment

Protocols requiring identity re-establishment need to re-gain control of the identity after the mobile attacker leaves [23], [24], [26], [27], [60]. This includes two common implementations, including physical links and manual operations. For the former, it assumes that the servers are authenticated through physical channels, such that when the attacker leaves at time

t', it cannot impersonate as the victim at time $t > t'$. For the latter, it assumes that the victim has the capacity to re-establish its identity at the speed of the adversary in corrupting a node. This typically requires manual process such as re-registration over the PKI or manually accessing a secure off-line storage.

D. Partition tolerance

Typically protocols under synchronous networks do not consider partition tolerance; however, partition tolerant protocols can still be secure against some adversaries in synchronous networks. Nonetheless, it is impossible for protocols to be secure under the sleepy model [31] and the mobile sluggish fault model [32], while tolerating partition.

VIII. Conclusion

In this work, we identified and filled a gap for generic adversary models, between traditional blockchain adversaries unable to model mobile DoS, and existing models that do model mobile DoS but only under restrictive conditions. To understand the security requirements of our model, we proved that *abundance* and *concealment* are sufficient, and that at least one of them is necessary for security. Additionally, we analyzed the trade-offs between the two properties. We applied our findings to evaluate the security of Ouroboros Praos, Algorand, and Hotstuff as case studies, confirming the intuition that Ouroboros Praos and Algorand tolerate a stronger adversary.

IX. Acknowledgement

This work was partially supported by the Australian Research Council (ARC) under project DE210100019.

References

[1] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, University of Guelph, 2016.

[2] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[3] J. Decouchant, D. Kozhaya, V. Rahli, and J. Yu, "DAMYSUS: streamlined BFT consensus leveraging trusted components," in *EuroSys*, Y. Bromberg, A. Kermarrec, and C. Kozyrakis, Eds. ACM, 2022, pp. 1–16.

[4] M. Yin, D. Malkhi, M. K. Reiter, G. Golan-Gueta, and I. Abraham, "Hotstuff: BFT consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. ACM, 2019, pp. 347–356.

[5] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[6] A. Hertig, "So, ethereum's blockchain is still under attack…," https://www.coindesk.com/markets/2016/10/06/so-ethereums-blockchain-is-still-under-attack/, 2016, accessed: 2023-04-06.

[7] H. Maishera, "Solana's latest ddos attack leads to poor network performance," https://finance.yahoo.com/news/solana-latest-ddos-attack-leads-120022342.html, 2022, accessed: 2023-04-06.

[8] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[9] IOHK, "Cardano docs: Ouroboros overview," https://docs.cardano.org/learn/ouroboros-overview/, 2023, accessed: 2023-12-28.

[10] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous verifiable secret sharing and proactive cryptosystems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 88–97.

[11] Y. Lu, Z. Lu, Q. Tang, and G. Wang, "Dumbo-mvba: Optimal multi-valued validated asynchronous Byzantine agreement, revisited," in *Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC)*, 2020, pp. 129–138.

[12] B. Guo, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo: Faster asynchronous BFT protocols," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM, 2020, pp. 803–818.

[13] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 31–42.

[14] I. Abraham, D. Dolev, A. Kagan, and G. Stern, "Brief announcement: Authenticated consensus in synchronous systems with mixed faults," in *36th International Symposium on Distributed Computing (DISC)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[15] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu, "Sbft: a scalable and decentralized trust infrastructure," in *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2019, pp. 568–580.

[16] M. Serafini, P. Bokor, D. Dobre, M. Majuntke, and N. Suri, "Scrooge: Reducing the costs of fast Byzantine replication in presence of unresponsive replicas," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2010, pp. 353–362.

[17] D. Malkhi, K. Nayak, and L. Ren, "Flexible byzantine fault tolerance," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1041–1053. [Online]. Available: https://doi.org/10.1145/3319535.3354225

[18] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin, "Sync hotstuff: Simple and practical synchronous state machine replication," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 106–118.

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

[20] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 281–310.

[21] N. Banu, S. Souissi, T. Izumi, and K. Wada, "An improved Byzantine agreement algorithm for synchronous systems with mobile faults," *International Journal of Computer Applications*, vol. 43, no. 22, pp. 1–7, 2012.

[22] F. Bonnet, X. Défago, T. D. Nguyen, and M. Potop-Butucaru, "Tight bound on mobile Byzantine agreement," in *International Symposium on Distributed Computing*. Springer, 2014, pp. 76–90.

[23] H. Buhrman, J. A. Garay, and J.-H. Hoepman, "Optimal resiliency against mobile faults," in *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*. IEEE, 1995, pp. 83–88.

[24] R. Ostrovsky and M. Yung, "How to withstand mobile virus attacks," in *Proceedings of the tenth annual ACM symposium on Principles of distributed computing*, 1991, pp. 51–59.

[25] T. Sasaki, Y. Yamauchi, S. Kijima, and M. Yamashita, "Mobile Byzantine agreement on arbitrary network," in *International Conference On Principles Of Distributed Systems*. Springer, 2013, pp. 236–250.

[26] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *annual international cryptology conference*. Springer, 1995, pp. 339–352.

[27] D. Schultz, B. Liskov, and M. Liskov, "Mpss: mobile proactive secret sharing," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 4, pp. 1–32, 2010.

[28] L. Zhou, F. B. Schneider, and R. Van Renesse, "Coca: A secure distributed online certification authority," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 329–368, 2002.

[29] V. Goyal, H. Li, and J. Raizes, "Instant block confirmation in the sleepy model," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2021, pp. 65–83.

[30] A. Momose and L. Ren, "Constant latency in sleepy consensus," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. ACM, 2022, pp. 2295–2308.

[31] R. Pass and E. Shi, "The sleepy model of consensus," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 380–409.

[32] Y. Guo, R. Pass, and E. Shi, "Synchronous, with a chance of partition tolerance," in *Annual International Cryptology Conference*. Springer, 2019, pp. 499–529.

[33] J. Kim, V. Mehta, K. Nayak, and N. Shrestha, "Brief announcement: Making synchronous BFT protocols secure in the presence of mobile sluggish faults," in *ACM Symposium on Principles of Distributed Computing (PODC), Virtual Event, Italy, July 26-30, 2021*. ACM, 2021, pp. 375–377.

[34] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Resilient intrusion tolerance through proactive and reactive recovery," in *13th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2007, pp. 373–380.

[35] C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov, "Yoso: you only speak once," in *Annual International Cryptology Conference*. Springer, 2021, pp. 64–93.

[36] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros chronos: Permissionless clock synchronization via proof-of-stake," *Cryptology ePrint Archive*, 2019.

[37] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren, "Synchronous byzantine agreement with expected O(1) rounds, expected o(n^2) communication, and optimal resilience," in *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, ser. Lecture Notes in Computer Science, I. Goldberg and T. Moore, Eds., vol. 11598. Springer, 2019, pp. 320–334. [Online]. Available: https://doi.org/10.1007/978-3-030-32101-7_20

[38] C. Stathakopoulou, T. David, M. Pavlovic, and M. Vukolic, "[solution] mir-bft: Scalable and robust BFT for decentralized networks," *J. Syst. Res.*, vol. 2, no. 1, 2022. [Online]. Available: https://doi.org/10.5070/sr32159278

[39] C. Liu, S. Duan, and H. Zhang, "Epic: efficient asynchronous bft with adaptive security," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 437–451.

[40] R. B. Ash, *Information theory*. Courier Corporation, 2012.

[41] M. K. Aguilera and S. Toueg, "A simple bivalency proof that *t*-resilient consensus requires *t* + 1 rounds," *Inf. Process. Lett.*, vol. 71, no. 3-4, pp. 155–158, 1999. [Online]. Available: https://doi.org/10.1016/S0020-0190(99)00100-3

[42] D. Dolev and R. Reischuk, "Bounds on information exchange for Byzantine agreement," *J. ACM*, vol. 32, no. 1, pp. 191–204, 1985. [Online]. Available: https://doi.org/10.1145/2455.214112

[43] J. Chen and S. Micali, "Algorand: A secure and efficient distributed ledger," *Theor. Comput. Sci.*, vol. 777, pp. 155–183, 2019.

[44] I. Abraham, T.-H. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi, "Communication complexity of byzantine agreement, revisited," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, ser. PODC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 317–326. [Online]. Available: https://doi.org/10.1145/3293611.3331629

[45] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[46] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco, "Single secret leader election," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 12–24.

[47] D. Catalano, D. Fiore, and E. Giunta, "Adaptively secure single secret leader election from ddh," in *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing (PODC)*, 2022, pp. 430–439.

[48] S. Das, V. Krishnan, I. M. Isaac, and L. Ren, "Spurt: Scalable distributed randomness beacon with transparent setup," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2502–2517.

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

[50] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, "Narwhal and tusk: a dag-based mempool and efficient BFT consensus," in *EuroSys '22: Seventeenth European Conference on Computer Systems, Rennes, France, April 5 - 8, 2022*, Y. Bromberg, A. Kermarrec, and C. Kozyrakis, Eds. ACM, 2022, pp. 34–50. [Online]. Available: https://doi.org/10.1145/3492321.3519594

[51] Y. Gao, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo-ng: Fast asynchronous BFT consensus with throughput-oblivious latency," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 1187–1201. [Online]. Available: https://doi.org/10.1145/3548606.3559379

[52] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 913–930.

[53] R. Gelashvili, L. Kokoris-Kogias, A. Sonnino, A. Spiegelman, and Z. Xiang, "Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback," in *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, ser. Lecture Notes in Computer Science, I. Eyal and J. A. Garay, Eds., vol. 13411. Springer, 2022, pp. 296–315. [Online]. Available: https://doi.org/10.1007/978-3-031-18283-9_14

[54] D. Malkhi and K. Nayak, "Extended abstract: Hotstuff-2: Optimal two-phase responsive BFT," *IACR Cryptol. ePrint Arch.*, p. 397, 2023. [Online]. Available: https://eprint.iacr.org/2023/397

[55] N. Giridharan, F. Suri-Payer, M. Ding, H. Howard, I. Abraham, and N. Crooks, "Beegees: Stayin' alive in chained bft," in *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, ser. PODC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 233–243. [Online]. Available: https://doi.org/10.1145/3583668.3594572

[56] Y. Lu, Z. Lu, and Q. Tang, "Bolt-dumbo transformer: Asynchronous consensus as fast as the pipelined BFT," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 2159–2173. [Online]. Available: https://doi.org/10.1145/3548606.3559346

[57] A. Spiegelman, A. Rinberg, and D. Malkhi, "ACE: abstract consensus encapsulation for liveness boosting of state machine replication," in *24th International Conference on Principles of Distributed Systems, OPODIS 2020, December 14-16, 2020, Strasbourg, France (Virtual Conference)*, ser. LIPIcs, Q. Bramas, R. Oshman, and P. Romano, Eds., vol. 184. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 9:1–9:18. [Online]. Available: https://doi.org/10.4230/LIPIcs.OPODIS.2020.9

[58] O. Alpos, C. Cachin, and L. Zanolini, "How to trust strangers: Composition of Byzantine quorum systems," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2021, pp. 120–131.

[59] L. Zhou, F. B. Schneider, and R. Van Renesse, "Apss: Proactive secret sharing in asynchronous systems," *ACM transactions on information and system security (TISSEC)*, vol. 8, no. 3, pp. 259–286, 2005.

[60] J. A. Garay, "Reaching (and maintaining) agreement in the presence of mobile faults," in *International Workshop on Distributed Algorithms*. Springer, 1994, pp. 253–264.