Verifying Randomized Consensus Protocols with Common Coins

Song Gao[§†], Bohua Zhan[§†], Zhilin Wu[§†], Lijun Zhang[§†‡]
[§]*University of Chinese Academy of Sciences, China*
[†]*Institute of Software, Chinese Academy of Sciences, China*
[‡]*Institute of Intelligent Software, Guangzhou, China*
{gaos, bzhan, wuzl, zhanglj}@ios.ac.cn

***Abstract*—Randomized fault-tolerant consensus protocols with common coins are widely used in cloud computing and blockchain platforms. Due to their fundamental role, it is vital to guarantee their correctness. Threshold automata is a formal model designed for the verification of fault-tolerant consensus protocols. It has recently been extended to probabilistic threshold automata (PTAs) to verify randomized fault-tolerant consensus protocols. Nevertheless, PTA can only model randomized consensus protocols with local coins.**

In this work, we extend PTA to verify randomized fault-tolerant consensus protocols with common coins. Our main idea is to add a process to simulate the common coin (the so-called common-coin process). Although the addition of the common-coin process destroys the symmetry and poses technical challenges, we show how PTA can be adapted to overcome the challenges. We apply our approach to verify the agreement, validity and almost-sure termination properties of 8 randomized consensus protocols with common coins.

***Index Terms*—Randomized consensus, Threshold automata, Distributed protocols, Common coin**

I. Introduction

Consensus is a fundamental problem in distributed computing, where a number of processes need to agree on some data values during computation. Consensus protocols are generally designed to be fault-tolerant or resilient, which means that they can withstand the existence of Byzantine or unreliable processes. Consensus protocols have many important applications in various fields, such as cloud computing and blockchains. As a result, there is a large body of work on designing and verifying consensus protocols [1]–[7]. However, formal verification of consensus protocols remains a difficult challenge, especially when the protocols are probabilistic and/or make use of additional cryptographic primitives.

The correct consensus protocols must meet three conditions: agreement, validity and termination. The result of FLP impossibility [8] states that there is no deterministic protocol that satisfies the consensus of an asynchronous distributed system, where any process can fail arbitrarily. Randomness provides a solution to reach consensus when the termination requirement is weakened to require termination with probability 1. In this way, the FLP argument no longer prohibits consensus: non-terminating executions still exist, but collectively they can only occur with probability 0.

Ben-or [1] and Rabin [2] proposed the first randomized consensus protocols, which laid the foundation for subsequent designs and contributed significantly to the development of the field. Early versions of randomized consensus protocols make use of a *local coin* for randomization, where each process throws the coin independently. As a consequence, they have an exponential expected number of rounds, making them of theoretical interest but of limited practical use. Rabin [2] introduced an additional computational power called a *common coin*, which delivers the same sequence of random bits $b_0, b_1, \ldots, b_r$ to all processes (each bit b_i has the value 0 or 1 with probability $1/2$). The common coin is powerful as it can provide a constant expected number of rounds.

However, designing and proving the correctness of a randomized fault-tolerant distributed protocol is challenging. Additionally, there exist several attacks [9], [10] against peer-reviewed and even practically used protocols. The attacks motivate formal verification of randomized fault-tolerant protocols. Threshold automata (TA) [4], [11] have been used extensively for the verification of fault-tolerant distributed protocols. The work of Bertrand et al. [5] proposes an extension of threshold automata by adding probability, probabilistic threshold automata (PTA). The consensus properties of protocols are reduced to queries on a one-round threshold automaton, which can be automatically checked using Byzantine Model Checker (ByMC) [12]. However, this approach assumes local coins (the randomness in each process is independent) and cannot be used directly for protocols involving common coins.

Extending PTAs to deal with common coins brings additional challenges. As processes are modeled as identical automata and the symmetry is essential in the TA and PTA theories, we cannot simply let one distinguished process toss the common coin. Moreover, throwing the common coin is assumed not to be subjected to Byzantine faults. Hence, a nontrivial extension of PTA is necessary for modeling common coins as well as extending parameterized verification methods to an asymmetric system. We propose such an extension, by adding an additional automaton for modeling the common coin, as well as extra shared variables for communication between the common coin automaton and other processes. We fully revisit the theories of threshold automata and probabilistic counter systems as well as adapt the theorems and proofs. We then reduce correctness and termination checks of the extended model to single-round queries on non-probabilistic threshold automata, which can be checked using ByMC.

2158-3927/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN58291.2024.00047

a) Computation Model: In this work we consider asynchronous processes, which means that each process proceeds at its own pace, which may vary arbitrarily with time, and remains always unknown to the other processes. The system is made up of a finite set of n asynchronous processes.

Processes communicate by exchanging messages through an asynchronous reliable point-to-point network. This network ensures that a message that has been sent is eventually received by its destination process without any loss, duplication or modification. Although there is no bound on the delay for message transfer, the network guarantees that the messages will be delivered correctly. The term "point-to-point" indicates that there is a bi-directional communication channel between each pair of processes, allowing a receiving process to identify the sender of a message. We assume that there are up to t processes may exhibit Byzantine faults.

This computation model is denoted $\mathcal{BAMP}_{n,t}[\emptyset]$ (BAMP stands for Byzantine Asynchronous Message Passing). This model is both restricted with a resilience condition and enriched with additional computational power. More precisely, $\mathcal{BAMP}_{n,t}[n > 3t, \text{CC}]$ denotes that the computational model is enriched with a common coin with up to $t < n/3$ Byzantine processes in the system.

ϵ-Good is an important property of the common coin abstraction: for any value $v \in \{0, 1\}$, all correct parties output v with probability $\geq \epsilon$. If a coin is $\frac{1}{2}$-good, we call it a *strong coin*. In this paper, we consider only the protocols that employ strong coins.

b) Contributions: The contributions of our work are as follows:

- We propose an extension of probabilistic threshold automata to incorporate common coins, and an extension of probabilistic counter system which removes the restrictions in the PTA method.
- As the proposed extensions break the crucial symmetry in TA and PTA, we revisit several theorems for reducing the verification of multi-round, probabilistic specifications to the checks of single-round, non-probabilistic formulas, and re-prove their correctness in our extended model.
- We reduce the correctness and termination conditions for probabilistic threshold automata with a common coin to a set of queries for single-round threshold automata. The termination conditions include those for checking the *binding* hyperproperty [7].
- Using the above framework and the ByMC tool, we verify a benchmark of 8 randomized distributed protocols using common coins. The verification is capable of reproducing the adaptive attack proposed in [9] against [13], and verifying that the fixed versions are correct.

c) Outline of the paper: The remaining sections of the paper are as follows. Sect. II gives a motivating example of common-coin-based protocol and its attack under an adaptive adversary. Sect. III introduces the framework of (probabilistic) threshold automata and extended counter systems with common coins. Sect. IV introduces randomized distributed consensus protocols and its correctness properties. Sect. V is the main part of the paper, describing our verification approach, reducing correctness of the protocol to proof obligations that can be checked using ByMC. We describe the experiments in Sect. VI, review related work in Sect. VII and conclude in Sect. VIII with a discussion of future work.

II. Motivating Protocol and its Attack

Common coin is a powerful abstraction to achieve randomized consensus, but the use of common coin in the presence of an *adaptive* adversary makes it more difficult to reason about termination of the protocol. In particular, the work of Mostéfaoui et al. [13] proposed the first signature-free protocol, namely `MMR14`, to achieve asynchronous Byzantine consensus, with $O(n^2)$ messages and tolerance of $t < n/3$ Byzantine processes. However, an attack [9] was later found for the protocol. In order to demonstrate the abstraction of common coin and the structure of a multi-round BFT protocol, we describe the protocol and its attack in some detail.

The protocol `MMR14` makes use of another abstraction called BV-broadcast, where each process broadcasts a binary value and obtains binary values in return. First, the i'th process $\mathcal{P}_i$ broadcasts its chosen value v_i. Then some value v is received from $t + 1$ processes and if v is not broadcast, it again broadcasts v. Finally, when some value v is received from $2t + 1$ different processes, it adds the value v to the set bin_values of values it received.

The complete consensus protocol proceeds in a number of rounds. Each process begins with a proposed value v_i, and let est_i be the current estimate of the value to be decided upon, which is initialized to be v_i. In each round, each process performs BV-broadcast of est_i using message of type EST, and waits until the set bin_values becomes nonempty. Then it BV-broadcasts each value in bin_values using message of type AUX. Next, it waits until receiving $n-t$ messages of type AUX, carrying values in the set bin_values, and then let s be the value of the next throw of the common coin. Let $values$ be the set of values in these AUX messages. If $values$ is a singleton $\{v\}$, then v becomes the new est_i. Further, if $v = s$, then v is decided upon. If $values$ contains both 0 and 1, then the new est_i is the coin value s.

The paper [13] gives a proof of termination of the protocol in expected finite number of rounds. The basic idea is by dividing the protocol into two distinct phases. In the first phase, it is guaranteed that all correct processes will eventually have the same value. It follows from the observation that at the end of each round, a correct process updates its value to either the only majority v (if it exists) or the common coin result. Then with probability $1/2$ the common coin result is equal to v and all correct processes get the same value v at the end of this round. Thus, the expected number of rounds for this to occur is bounded by 2. Moving on to the second phase, all correct processes broadcast the agreed-upon value v and we can easily conclude that it happens in every later round. Then with probability $1/2$ the common coin result aligns with v and consequently all correct processes decide v in this round. Similarly, the expected number of rounds for the second phase

1: $est_i \leftarrow init$; $r_i \leftarrow 0$;
2: **while** true **do**
3: $r_i \leftarrow r_i + 1$; BV_broadcast(EST,r_i,est_i);
4: **wait until** ($bin\_values_i[r_i] \neq \emptyset$);
5: broadcast(AUX,r_i,w), where $w \in bin\_values_i[r_i]$;
6: **wait until** ($\exists$ a set of $(n-t)$ (AUX,r_i,x) messages from distinct processes such that $values_i \subseteq bin\_values_i[r_i]$, where $values_i$ is the set of values in the messages);
7: $s \leftarrow$ random(); % common coin %
8: **if** ($values_i = \{v\}$) % i.e., $|values_i| = 1$ %
9: **then** $est_i \leftarrow v$;
10: **if** $v = s$ **then** decide(v) if not yet done;
11: **end if**
12: **else** $est_i \leftarrow s$;
13: **end if**
14: **end while**

Fig. 1: Randomized Consensus Protocol MMR14 for Correct Process $\mathcal{P}_i$

is bounded by 2. Combining the two phases, the expected termination time is four rounds.

However, this proof neglects the ability of an adaptive adversary to obtain the value of the common coin and then manipulates the behavior of the Byzantine processes as well as the schedule for sending messages to make the protocol never terminate. In more detail, consider a smallest system consisting of 3 correct processes A_1, A_2, B_1 and a Byzantine process P_{byz}, and at a round k, A_1, A_2 propose estimate value 0 while B_3 proposes 1. The adversary can delay A_2's reception of all messages unboundedly, while A_1 and B_1 proceed until they have the same set $values = bin\_values = \{0, 1\}$. Therefore they both enter Line 12 and can only set their est value to be the coin result s. Later P_{byz} can manipulate the order of A_2 receiving messages, make its $values$ set equal to $\{1 - s\}$ and finally set its est value as $1 - s$. At the end of this round, two correct processes have their new est value s and one has $1 - s$, which is either the same or dual to the initial state. It indicates that no correct process can ever terminate.

This attack was later fixed by the authors in the journal version of the paper [14]. Moreover, the work by Abraham et al. [7] proposes a general framework for building protocols tolerating crash or Byzantine failures, and the implementation in the paper can also be viewed as a fixed version of MMR14. A key contribution of [7] is proposing the *binding* condition, which summarizes the property the protocol must satisfy in order to prevent the above attack. Intuitively, the binding condition states that in any round of an execution, by the time the first correct process accesses the common coin, there is already a value $b \in \{0, 1\}$ such that in any extension of the execution, no process may output b in the same round. Since the adversary gains knowledge of the common coin only when the first correct process accesses it, the adversary can no longer always manipulate the Byzantine processes to output the opposite value. A formal definition of the binding condition, in the language of threshold automata defined in our work, will be given in Sect. V-B.

1: **input** $b_i \in \{0, 1\}$;
2: broadcast message b_i;
3: **wait until** ($\exists$ $d_i \in \{0, 1\}$ is received $\lceil \frac{n+1}{2} \rceil$ times);
4: decide(d_i)

Fig. 2: Naive Voting Protocol for Correct Process $\mathcal{P}_i$

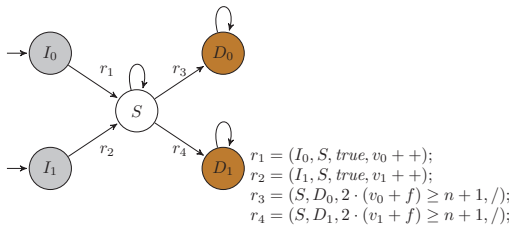


Fig. 3: Threshold Automaton for Naive Voting

III. The Framework of Probabilistic Threshold Automata Extended with Common Coins

A. Threshold Automata

Let us start with threshold automata [4], [11], [12]. A threshold automaton describes how a correct process runs in a concurrent system, and it usually contains local locations, variables and transition rules. Then the system can be abstracted as a counter system of multiple copies of such threshold automata, and its running states can be captured by the counters of locations.

Example 1 *Fig. 2 shows a simple protocol of majority voting, and we model it with the threshold automaton shown in Fig. 3. We assume that there are n processes in total, and f is the number of Byzantine processes. We run $n - f$ instances of the threshold automaton; each instance is modeling a correct process.*

There are two initial locations $\{I_0, I_1\}$, which indicate the input of the process, and two final locations $\{D_0, D_1\}$ for its decision value. The shared variables v_0, v_1 represent the number of messages sent by the correct processes. The transition rules show steps of the protocol: r_1, r_2 for Line 2 and r_3, r_4 for Line 3. Let c_0 be the configuration where $v_0 = v_1 = 0$, and all counters are 0 expect the counter of I_0 equals $n - f$. This configuration corresponds to a concurrent system where all correct processes have the same input 0 and have not yet broadcast any message.

Byzantine processes are not directly modeled in the automaton. However, Byzantine behaviors can be captured by their impact on the transitions of correct processes and on the threshold guards. For instance, assume that $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ involve in the naive voting protocol and $\mathcal{P}_3$ is Byzantine. $\mathcal{P}_1$ and $\mathcal{P}_2$ propose different binary values, therefore their threshold automata start with different initial states and reach location S. $\mathcal{P}_1$ and $\mathcal{P}_2$ cannot proceed if they receive no message from $\mathcal{P}_3$, and if any of them decides value 0, i.e. its automaton

reaches location D_0, we can infer that Byzantine process $\mathcal{P}_3$ sends a message with value 0 to it. In the threshold automaton, it is represented as a non-deterministic choice of r_3 and r_4.

B. Probabilistic Threshold Automata Extended with Common Coins

We present the framework of probabilistic threshold automata and counter systems extended with common coins, illustrating the definitions in the example of Fig. 4, a model of `MMR14`. The automata rules are given in Table I.

a) Environments: (Probabilistic) threshold automata are defined relative to an environment $Env = (\Pi, RC, N)$, where Π is a set of parameters that range over $\mathbb{N}_0$, RC is the resilience condition, a formula in linear integer arithmetic over parameters. Intuitively, a valuation of Π determines the number of different types of process in the system, RC defines the set of admissible parameters $\mathbf{P}_{RC} = \{\mathbf{p} \in \mathbb{N}_0^{|\Pi|} : \mathbf{p} \models RC\}$. $N : \mathbf{P}_{RC} \to \mathbb{N}_0^2$ is a function that maps a vector of admissible parameters to the number of modeled processes and common coins in the system.

Example 2 *In the threshold automata of Fig. 4, the parameters are n, f, t and cc, denoting the total number of processes, the actual number of Byzantine processes, the maximum number of Byzantine processes while ensuring the correctness, and the number of common coins, respectively. The resilience condition is $n > 5t \ \wedge \ t \geq f \ \wedge \ f \geq 0 \ \wedge \ cc \geq 1$, while the function N is given by $(n, f, t, cc) \mapsto (n - f, 1)$, as we model only $n - f$ correct processes and 1 common coin explicitly.*

Next, we can define the (non-probabilistic) threshold automata for correct processes and the probabilistic threshold automata for common coins in the multi-round setting.

b) Threshold Automata for Correct Processes: Formally, a (non-probabilistic) threshold automaton over an environment (Π, RC, N) is a tuple $\mathbf{TA}^n = (\mathcal{L}^n, \mathcal{V}^n, \mathcal{R}^n)$, where

- $\mathcal{L}^n$: a finite set of locations, which contains the following disjoint subsets: initial locations $\mathcal{I}^n$, final locations $\mathcal{F}^n$, and border locations $\mathcal{B}^n$, with $|\mathcal{B}^n| = |\mathcal{I}^n|$;
- $\mathcal{V}^n$: a finite set of variables, including shared variables Γ and coin variables Ω;
- $\mathcal{R}^n$: a finite set of rules;

A simple guard is an expression of the form

$$b \cdot x \geq \bar{a} \cdot \mathbf{p}^\top + a_0 \quad \text{or} \quad b \cdot x < \bar{a} \cdot \mathbf{p}^\top + a_0 \ ,$$

where $x \in \Gamma$ is a shared variable, $\bar{a} \in \mathbb{Z}^{|\Pi|}$ is a vector of integers, $a_0, b \in \mathbb{Z}$, and $\mathbf{p}$ is the vector of all parameters. Similarly, we can define a coin guard in the same form but on a coin variable.

A rule is a tuple $r = (from, to, \varphi, \mathbf{u})$, where $from, to \in \mathcal{L}^n$ are the source and destination locations, φ is a conjunction of guards, and $\mathbf{u} \in \mathbb{N}_0^{(|\Gamma|+|\Omega|)}$ is the update vector.

Note that for any rule $r = (from, to, \varphi, \mathbf{u})$, we have an additional restriction on φ that it should be either a conjunction of simple guards or a conjunction of coin guards. We call a rule r coin-based if $r.\varphi$ is a conjunction of coin guards; otherwise it is non-coin-based. Another restriction on $\mathbf{u}$ is necessary that the projection of $\mathbf{u}$ on the coin variables should be $\mathbf{0}$, that is, executing a rule in the threshold automata for correct processes should always keep the coin variables unchanged.

Threshold automata can model protocols with multiple rounds that follow the same code. Informally, a round starts from border locations and ends in final locations, and the code of a round is modeled by the transitions between initial locations and final locations. As $|\mathcal{B}^n| = |\mathcal{I}^n|$, we see that from each border location there is one rule towards an initial location, and it has the form $(\ell, \ell', true, \mathbf{0})$ where $\ell \in \mathcal{B}^n$ and $\ell' \in \mathcal{I}^n$. There are round-switch rules that let processes move from final locations of a certain round to border locations of the next round. They can be described as rules $(\ell, \ell', true, \mathbf{0})$ where $\ell \in \mathcal{F}^n$ and $\ell' \in \mathcal{B}^n$. The set of round-switch rules is denoted by $\mathcal{S}^n \subseteq \mathcal{R}^n$. A location belongs to $\mathcal{B}^n$ if and only if all incoming edges are in $\mathcal{S}^n$. Similarly, a location is in $\mathcal{F}^n$ if and only if there is only one outgoing edge and it is in $\mathcal{S}^n$.

A threshold automaton is called canonical if every rule r that lies on a cycle ensures that $r.\mathbf{u} = \mathbf{0}$, and we consider only canonical ones in this work.

For binary consensus, every correct process, say $\mathcal{P}_i$, has an initial value $init_i \in \{0, 1\}$, and its valid decision value (if any) should also be binary. Therefore, we can partition every set of locations $\mathcal{I}^n$, $\mathcal{F}^n$ and $\mathcal{B}^n$ into two subsets $\mathcal{I}_0^n \uplus \mathcal{I}_1^n$, $\mathcal{F}_0^n \uplus \mathcal{F}_1^n$ and $\mathcal{B}_0^n \uplus \mathcal{B}_1^n$, respectively. For every $v \in \{0, 1\}$, the partitions follow the 2 rules below:

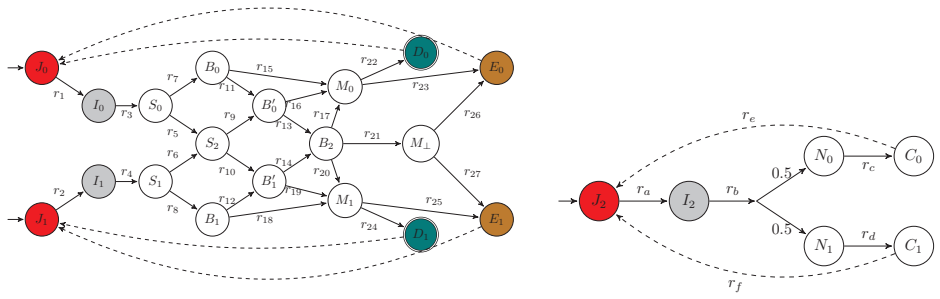
1) The processes that are initially in a location $\ell \in \mathcal{I}_v^n$ have the initial value v.
2) Rules connecting locations from $\mathcal{B}^n$ and $\mathcal{I}^n$ respect the partitioning, i.e., they connect $\mathcal{B}_v^n$ and $\mathcal{I}_v^n$. Similarly, rules that connect the locations of $\mathcal{F}^n$ and $\mathcal{B}^n$ respect the partitioning.

For common-coin-based protocols with a step for deciding a binary value, we can introduce two subsets, decision locations $\mathcal{D}_v^n \subseteq \mathcal{F}_v^n$, $v \in \{0, 1\}$. Intuitively, a process is in $\mathcal{D}_v^n$ locations in round k if and only if it decides v in that round. Decision locations are accepting locations in a threshold automaton.

Example 3 *Fig. 4(a) depicts a threshold automaton with border locations $\mathcal{B}^n = \{J_0, J_1\}$, initial locations $\mathcal{I}^n = \{I_0, I_1\}$, final locations $\mathcal{F}^n = \{E_0, E_1, D_0, D_1\}$, and decision locations $\mathcal{D}^n = \{D_0, D_1\}$. As a_0, a_1, b_0, b_1 are shared variables and cc_0, cc_1 are coin variables , there are 6 coin-based rules $r_{22}, r_{23}, r_{24}, r_{25}, r_{26}$ and r_{27}. The round-switch rules are represented by dashed arrows, and the self loops are omitted.*

c) Probabilistic Threshold Automata for Common Coins: A probabilistic threshold automaton for a common coin over an environment (Π, RC, N) is $\mathbf{PTA}^c = (\mathcal{L}^c, \mathcal{V}^c, \mathcal{R}^c)$, which extends the definition of threshold automata in the part of rules.

Here a rule is a tuple $r = (from, \delta_{to}, \varphi, \mathbf{u})$, where $from \in \mathcal{L}^c$ is the source location, $\delta_{to} \in \mathtt{Dist}(\mathcal{L}^c)$ is a probabilistic distribution over the destination locations, φ is a conjunction of simple guards, and $\mathbf{u} \in \mathbb{N}_0^{(|\Gamma|+|\Omega|)}$ is the update vector.



(a) (b)

Fig. 4: Multi-round Threshold Automata for MMR14, with Self Loops Omitted

If there exists $\ell \in \mathcal{L}^c$ such that $r.\delta_{to}(\ell) = 1$, we call the distribution a *Dirac* distribution and the rule a *Dirac* rule $r = (\mathit{from}, \ell, \varphi, \mathbf{u})$.

Note that we remove the restriction on the occurrence of non-*Dirac* rules, while [5] requires that the destination locations of all non-*Dirac* rules should be in $\mathcal{F}^c$. The restrictions on rules are different in probabilistic threshold automata for common coins: $r.\varphi$ should only involve simple guards, and $r.\mathbf{u}$ cannot modify the values of shared variables.

Example 4 *Fig. 4(b) shows a probabilistic threshold automaton for the common coin in* MMR14*. We have border locations* $\mathcal{B}^c = \{J_2\}$*, initial locations* $\mathcal{I}^c = \{I_2\}$*, and final locations* $\mathcal{F}^c = \{C_0, C_1\}$*. The only non-Dirac rule is* r_b*.*

Finally, given $\mathit{Env} = (\Pi, \mathit{RC}, N)$, the non-probabilistic threshold automaton $\mathbf{TA}^n = (\mathcal{L}^n, \mathcal{V}^n, \mathcal{R}^n)$ for correct processes and the probabilistic threshold automaton $\mathbf{PTA}^c = (\mathcal{L}^c, \mathcal{V}^c, \mathcal{R}^c)$ for common coins, we have that they share the same set of variables, i.e., $\mathcal{V}^n = \mathcal{V}^c$, and their sets of locations and rules are naturally disjoint. For simplicity, we write $\mathcal{L}$ for $\mathcal{L}^n \cup \mathcal{L}^c$, $\mathcal{V}$ for $\mathcal{V}^n$ and $\mathcal{R}$ for $\mathcal{R}^n \cup \mathcal{R}^c$. Note that $(\mathcal{L}, \mathcal{V}, \mathcal{R})$ does not form a probabilistic threshold automaton for this system.

C. Extended Probabilistic Counter Systems

The semantics of the probabilistic multi-round system is an infinite-state Markov decision process (MDP). Given an environment $\mathit{Env} = (\Pi, \mathit{RC}, N)$, a threshold automaton $\mathbf{TA}^n = (\mathcal{L}^n, \mathcal{V}^n, \mathcal{R}^n)$ for correct processes and a probabilistic threshold automaton $\mathbf{PTA}^c = (\mathcal{L}^c, \mathcal{V}^c, \mathcal{R}^c)$ for the common coin, we define the semantics, called counter system $\mathit{Sys}(\mathbf{TA}^n, \mathbf{PTA}^c)$ over *Env*, to be the infinite-state MDP $(\Sigma, I, \mathbf{Act}, \Delta)$, where

- Σ: the set of configurations,
- $I \subseteq \Sigma$: the set of initial configurations,
- $\mathbf{Act} = \mathcal{R} \times \mathbb{N}_0$: the set of actions labelled by round numbers,
- $\Delta : \Sigma \times \mathbf{Act} \to \mathtt{Dist}(\Sigma)$ is the probabilistic transition function.

a) Configurations: A configuration is a tuple $c = (\kappa, \mathbf{g}, \mathbf{p})$, where the function $c.\kappa : \mathcal{L} \times \mathbb{N}_0 \to \mathbb{N}_0$ describes the values of the location counters in each round, $c.\mathbf{g} : \mathcal{V} \times \mathbb{N}_0 \to \mathbb{N}_0$ defines the values of the variables in each round, and the vector $c.\mathbf{p}$ shows the values of the parameters. We denote the vector $(\mathbf{g}[x,k])_{x \in \mathcal{V}}$ of all variables in round k by $\mathbf{g}[k]$, and denote the vector $(\kappa[\ell,k])_{\ell \in \mathcal{L}}$ of location counters in round k by $\kappa[k]$.

TABLE I: The Rules of the Multi-round Threshold Automaton for MMR14

Rules	Guard	Update
r_1, r_2	*true*	-
r_3	*true*	b_0++
r_4	*true*	b_1++
r_6,r_{12}	$b_0 \geq t+1-f$	b_0++
r_5,r_{11}	$b_1 \geq t+1-f$	b_1++
r_7, r_9	$b_0 \geq 2t+1-f$	a_0++
r_8, r_{10}	$b_1 \geq 2t+1-f$	a_1++
r_{13}	$b_1 \geq 2t+1-f$	-
r_{14}	$b_0 \geq 2t+1-f$	-
r_{15}, r_{16}, r_{17}	$a_0 \geq n-t-f$	-
r_{18}, r_{19}, r_{20}	$a_1 \geq n-t-f$	-
r_{21}	$a_0+a_1 \geq n-t-f \wedge a_0 \geq 0 \wedge a_1 \geq 0$	-
r_{22}, r_{25}, r_{26}	$cc_0 > 0$	-
r_{23}, r_{24}, r_{27}	$cc_1 > 0$	-
r_a, r_b	*true*	-
r_c	*true*	cc_0++
r_d	*true*	cc_1++

A configuration is initial if all processes as well as the common coin are in initial locations of round 0, and all variables evaluate to 0. A threshold guard evaluates to true in a configuration c for a round k, written $c, k \models \varphi$, if for all its conjuncts $b \cdot x \geq \bar{a} \cdot \mathbf{p}^\top + a_0$, it holds that $b \cdot c.\mathbf{g}[x,k] \geq \bar{a} \cdot (c.\mathbf{p}^\top) + a_0$, and similarly for conjuncts of the other form.

b) Actions and Probabilistic Transition Function: An action $\alpha = (r,k) \in \mathbf{Act}$ stands for the execution of a rule $r \in \mathcal{L}$ in round k by a single process.

We say an action α unlocked in a configuration c if the guard of its rule evaluates to true in round k, that is, $c, k \models r.\varphi$. An action $\alpha = (r,k)$ is applicable to a configuration c if α is unlocked in c and the location counter value at round k for the source location of its rule is at least 1, formally, $c.\kappa[r.\mathit{from}, k] \geq 1$. When an action α is applicable to configuration c and ℓ is a potential destination location for the probabilistic action α, we denote the resulting configuration

as *apply*(α, c, ℓ). In this resulting configuration, the parameters remain unchanged, the variables are updated according to the update vector $\alpha.\mathbf{u}$, and the values of location counters are modified in a natural way: in round $\alpha.k$, the counter of source location $\alpha.$*from* decreases by 1, the counter of destination location ℓ increases by 1, and the others remain unchanged.

The probabilistic transition function Δ is defined such that for any two configurations c and c', and for any action α applicable to c, we have

$$\Delta(c,\alpha)(c') = \begin{cases} \alpha.\sigma_{to}(\ell) & \text{if } \textit{apply}(c,\alpha,\ell) = c' \\ 0 & \text{otherwise} \end{cases}$$

D. Non-probabilistic Extended Counter Systems

Given a probabilistic threshold automaton **PTA** over an environment, we can replace probability with non-determinism and get a non-probabilistic threshold automaton $\mathbf{TA}_{\text{PTA}}$.

Definition 1 *Given a* $\mathbf{PTA} = (\mathcal{L}, \mathcal{V}, \mathcal{R})$ *over an environment Env, its non-probabilistic threshold automaton is* $\mathbf{TA}_{\text{PTA}} = (\mathcal{L}, \mathcal{V}, \mathcal{R}_{np})$ *over Env where the set of rules* $\mathcal{R}_{np}$ *is defined as* $\{r_\ell = (\textit{from}, \ell, \varphi, \mathbf{u}) \mid r = (\textit{from}, \delta_{to}, \varphi, \mathbf{u}) \in \mathcal{R} \wedge \ell \in \mathcal{L} \wedge \delta_{to}(\ell) > 0\}$.

In words, $\mathcal{R}_{np}$ contains all *Dirac* rules in $\mathcal{R}$, and turns all probabilistic branches of non-*Dirac* rules into non-deterministic *Dirac* rules. We write **TA** for $\mathbf{TA}_{\text{PTA}}$ when the automaton **PTA** is clear from the context, e.g., the non-probabilistic threshold automaton of $\mathbf{PTA}^c$ is $\mathbf{TA}^c$.

Given an environment *Env* = (Π, RC, N), a threshold automaton $\mathbf{TA}^n$ for correct processes and a probabilistic threshold automaton $\mathbf{PTA}^c$ for the common coin, we can first get the non-probabilistic $\mathbf{TA}^c$, and then define an infinite non-probabilistic counter system $\textit{Sys}_\infty(\mathbf{TA}^n, \mathbf{TA}^c)$ over *Env*, to be the tuple $(\Sigma, I, \mathbf{Act}', R)$. The set of configurations and initial configurations Σ, I are defined as in Sect. III-C. An action $t \in \mathbf{Act}'$ is a tuple $(r,k) \in (\mathcal{R}^n \cup \mathcal{R}^c_{np}) \times \mathbb{N}_0$, and R is the transition relation. Two configurations c_0, c_1 are in the transition relation, i.e., $(c_0, c_1) \in R$ if and only if there exists an action t such that $t(c_0) = c_1$.

In the non-probabilistic counter system, a (finite or infinite) sequence of transitions is called schedule, and it is often denoted by τ. A schedule $\tau = t_1, t_2, \ldots, t_{|\tau|}$ is applicable to a configuration c if there is a sequence of configurations $c_0, c_1, \ldots, c_{|\tau|}$ such that for every $1 \le i \le |\tau|$ we have that t_i is applicable to c_{i-1} and $c_i = t_i(c_{i-1})$. Given a configuration c_0 and a schedule τ, we denote by *path*(c_0, τ) a path $c_0, t_1, c_1, \ldots, t_{|\tau|}, c_{|\tau|}$ where $t_t(c_{i-1}) = c_i$ for every $1 \le i \le |\tau|$. Similarly we define an infinite schedule τ and an infinite path also denoted by *path*(c_o, τ).

An infinite path is fair if no transition is applicable forever from some point on. Equivalently, when a transition is applicable, eventually either its guard becomes false, or all processes leave its source location.

E. Adversaries

The non-determinism is usually resolved by a so-called adversary. We denote by Σ^+ the set of all non-empty sequences of configurations. An adversary is a function $a : \Sigma^+ \to \mathbf{Act}$, which given a sequence of configurations $\xi \in \Sigma^+$ selects an action applicable to the last configuration of ξ. Given a configuration c and an adversary a, we generate a family of paths, depending on the outcomes of non-Dirac transitions, and we denote this set by $paths(c, a)$. An adversary a is fair if all paths in $paths(c, a)$ are fair.

The Markov Decision Process (MDP) $\textit{Sys}(\mathbf{TA}^n, \mathbf{PTA}^c)$ over *Env* together with an initial configuration c and an adversary a induce a Markov chain, written as $\mathcal{M}^c_a$. We denote by $\mathbb{P}^c_a$ the probability measure over infinite paths starting at c in the latter Markov chain.

An adversary a is round-rigid if it is fair, and if every sequence of actions it produces can be decomposed into a concatenation of sequences in the form $s_0 \cdot s_1 \cdot s_2 \ldots$, where the sequence s_k contains only actions of round k. We denote the set of all round-rigid adversaries by $\mathcal{A}^R$.

The atomic propositions AP_k and stutter equivalence discussed in this work follow the definitions in [5].

IV. Randomized Distributed Consensus protocols

The consensus problem was first introduced by Lamport et al. [15]. It can be stated in a basic, generic manner: One or more processes may propose some value. How do we get a collection of correct processes to agree on exactly one of those proposed values?

In binary cases, assuming each correct process p_i proposes a value $v_i \in \{0, 1\}$, each of them has to decide a binary value.

Definition 2 (Consensus) *Consensus is reached if the following properties hold:*

- *Agreement: No two correct processes decide different values.*
- *Validity: A decided value was proposed by a correct process.*
- *Termination: Each correct process decides.*

The famous FLP impossibility shows that no deterministic consensus protocol can be possible in asynchronous settings as soon as one node may crash. Ben-or [1] and Rabin [2] are the first to show that the impossibility can be circumvented via randomness. In this paper, we focus on randomized binary consensus protocols in asynchronous systems, where randomization is provided by common coins. *Randomized Consensus* is then defined by *Agreement, Validity*, plus the following *Almost-sure Termination* property: Each correct process decides with probability 1. For round-based protocols, consider the event $\mathcal{E}(i, r)$: process p_i decides by round r. Then this termination property is re-stated as: for any correct process p_i, we have $\lim_{r \to +\infty} \mathbb{P}(\mathcal{E}(i, r)) = 1$.

Now we can express the specifications in $\text{LTL}_{-\text{X}}$ as follows:

- Agreement: no two correct processes decide differently.

For both $v \in \{0,1\}$, the following holds: $\forall k, k' \in \mathbb{N}_0$.

$$\mathbf{A}(\mathbf{F} \bigvee_{\ell \in \mathcal{D}_v^n} \kappa[\ell,k] > 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{D}_{1-v}^n} \kappa[\ell',k'] = 0) \quad (\textit{Agree})$$

- Validity: if all correct processes have v as the initial value, then no process decides $1-v$. For both $v \in \{0,1\}$, the following holds: $\forall k \in \mathbb{N}_0$.

$$\mathbf{A}(\mathbf{G} \bigwedge_{\ell \in \mathcal{I}_v^n} \kappa[\ell,0] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{D}_v^n} \kappa[\ell',k] = 0) \quad (\textit{Valid})$$

- Almost-sure Termination under Round-rigid Adversaries: For every initial configuration s and every round-rigid adversary a, the following holds:

$$\mathbb{P}_a^s[\ \exists k \in \mathbb{N}_0.\ \mathbf{G} \bigwedge_{\ell \in \mathcal{F}^n \setminus \mathcal{D}^n} \kappa[\ell,k] = 0\ ] = 1 \quad (\textit{Termin})$$

V. Verification of Randomized Consensus

A. Towards verifying non-probabilistic properties

For the specifications of safety properties, i.e., *Agreement* and *Validity*, we observe that they are both non-probabilistic properties and concern about locations in the threshold automata of correct processes. *Agreement* contains two round variables k and k', and *Validity* considers round 0 and k. We would like to check these specifications in the ByMC tool, which allows the properties to use only one round number. Therefore, we introduce two round invariants that refer to one round and prove that these two round invariants imply the consensus properties *Agreement* and *Validity* as follows.

The first round invariant claims that in every round and in every path, once a correct process decides $v \in \{0,1\}$ in a round, no correct process ever enters a location from $\mathcal{F}_{1-v}^n$ in that round. Formally, $\forall k \in \mathbb{N}_0$.

$$\mathbf{A}(\mathbf{F} \bigvee_{\ell \in \mathcal{D}_v^n} \kappa[\ell,k] > 0 \ \rightarrow\ \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}_{1-v}^n} \kappa[\ell',k] = 0) \quad (\textit{Inv1})$$

The second round invariant claims that in every round and in every path, if no correct process starts a round with a value $v \in \{0,1\}$, then no correct process ever ends that round with v. Formally, $\forall k \in \mathbb{N}_0$.

$$\mathbf{A}(\mathbf{G} \bigwedge_{\ell \in \mathcal{I}_v^n} \kappa[\ell,k] = 0 \ \rightarrow\ \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}_v^n} \kappa[\ell',k] = 0) \quad (\textit{Inv2})$$

Round switch lemma is useful in the following reasoning. It states that in every round and in every run, if no process ever enters a final location with value v, then in the next round, there will be no process in any initial location with value v.

Lemma 1 (Round switch) *For every Sys= $Sys_\infty(\mathbf{TA}^n, \mathbf{TA}^c)$ and every $v \in \{0,1\}$, we have: $\forall k \in \mathbb{N}_0$.*

$$\mathbf{A}(\mathbf{G} \bigwedge_{\ell \in \mathcal{F}_v^n} \kappa[\ell,k] = 0 \ \rightarrow\ \mathbf{G} \bigwedge_{\ell' \in \mathcal{I}_v^n} \kappa[\ell',k+1] = 0) \quad (\textit{RS})$$

Proof *It follows from the definitions of $\mathcal{I}_v$, $\mathcal{F}_v$ and $\mathcal{B}_v$.* □

Proposition 1 *If Sys $\models$ (Inv1)$\wedge$(Inv2), then Sys $\models$ (Agree)$\wedge$(Valid).*

The proof is omitted due to length constraints.

We utilize a similar method as [5] to check one-round properties. The main idea is to prove that there exists a counterexample to the property in the multi-round system if and only if there is a counterexample in a single-round system. To this end, we prove the following two theorems in the non-probabilistic extended counter systems $Sys_\infty(\mathbf{TA}^n, \mathbf{TA}^c)$.

The first theorem states that every finite schedule can be reordered into a round-rigid one that is stutter equivalent regarding $\mathrm{LTL}_{-\mathrm{X}}$ formulas over proposition from AP_k, for all rounds k, therefore it is sufficient to reason about the round-rigid schedules.

Theorem 1 *For every configuration c and every finite schedule τ applicable to c, there is a round-rigid schedule τ' such that the following holds:*

- *Schedule τ' is applicable to configuration c,*
- *τ' and τ reach the same configuration when applied to c, i.e., $\tau'(c) = \tau(c)$,*
- *for every $k \in \mathbb{N}_0$, we have that path(c,τ') and path(c,τ) are stutter equivalent w.r.t. AP_k.*

The next theorem provides a simple way to check specifications with one round number, which include both round invariants (*Inv1*) and (*Inv2*). It allows us to check specifications using single-round systems. First we need to build the single-round threshold automata for $\mathbf{TA}^n$ and $\mathbf{TA}^c$.

Definition 3 *Given a $\mathbf{TA}{=}(\mathcal{L},\mathcal{V},\mathcal{R})$ over Env, its single-round threshold automaton is $\mathbf{TA}_{rd}{=}(\mathcal{L}\cup\mathcal{B}',\mathcal{V},\mathcal{R}_{rd})$, where $\mathcal{B}'{=}\{\ell'\colon\ \ell \in \mathcal{B}\}$ are copies of border locations, and the set of transition rules $\mathcal{R}_{rd}{=}\ (\mathcal{R}\setminus\mathcal{S}) \cup \mathcal{S}' \cup \mathcal{R}_{loop}$, where $\mathcal{R}_{loop}{=}\ \{(\ell',\ell',\mathit{true},\mathbf{0})\}$ are self-loops at locations in $\mathcal{B}'$, and $\mathcal{S}'{=}\{(\mathit{from},\ell',\mathit{true},\mathbf{0})\colon\ (\mathit{from},\ell',\mathit{true},\mathbf{0}) \in \mathcal{S}$ with $\ell' \in \mathcal{B}'\}$ consists of modifications of round-switch rules.*

Intuitively, given a multi-round $\mathbf{TA}$, we can remove its round-switch rules, add two more border locations $\mathcal{B}'$ and rules that direct final locations to $\mathcal{B}'$ and self-loops. In this way, we get a single-round threshold automaton $\mathbf{TA}_{rd}$ starting from $\mathcal{B}$ and ending with $\mathcal{B}'$, which models in some round a correct process starts and ends with a binary value, respectively. Given the single-round threshold automata for correct processes ($\mathbf{TA}_{rd}^n$) and common coin ($\mathbf{TA}_{rd}^c$), in a similar way, we can construct a single-round counter system $Sys^k(\mathbf{TA}_{rd}^n, \mathbf{TA}_{rd}^c)$ to model the system in round k.

Theorem 2 *Let TA be non-blocking, and let all fair executions of $Sys^0(\mathbf{TA}_{rd}^n, \mathbf{TA}_{rd}^c)$ terminate w.r.t. all possible initial configurations. If $\varphi[k]$ is a $\mathrm{LTL}_{-\mathrm{X}}$ formula over AP_k for a round variable $k \in \mathbb{N}_0$, the following points are equivalent:*

- *$Sys_\infty(\mathbf{TA}^n, \mathbf{TA}^c) \models \forall k \in \mathbb{N}_0.\mathbf{A}\varphi[k]$*
- *$Sys^0(\mathbf{TA}_{rd}^n, \mathbf{TA}_{rd}^c) \models \mathbf{A}\varphi[0]$ with respect to the initial configurations Σ^u, where Σ^u is the union of all renamed*

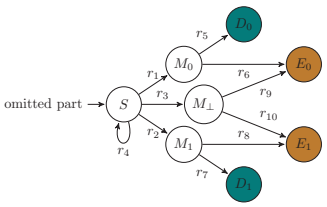


Fig. 5: The Common Part in the Threshold Automata of (C)

initial configurations from all rounds.

The proofs are omitted due to length constraints.

B. Round-rigid probabilistic termination

Considering the differences in the design of common-coin-based consensus protocols, we can roughly divide them into three categories. For each category of protocols, they have similar sufficient conditions for their almost-sure termination.

Specifically, let $Sys = Sys_\infty(\mathbf{TA}^n, \mathbf{TA}^c)$ and $Env = (\Pi, RC, N)$ be the counter system and environment of the protocol, and nc is the number of modeled correct processes, we can classify the protocols in the following way:

- (A) the protocol does not have a "decide" action, that is, there are no accepting locations $\mathcal{D}$ in $\mathbf{TA}^n$, the threshold automata of correct processes;
- (B) the protocol has a "decide" action and all its messages and conditions contain only binary values, that is, there are accepting locations $\mathcal{D}$ in $\mathbf{TA}^n$, and for every pair of shared variables on the same messages $s_0, s_1 \in \Gamma$ and every final configuration $c_f \in \mathcal{F}$, $c_f.s_0 + c_f.s_1 \leq nc$;
- (C) the protocol has a "decide" action and uses a "Binary Crusader Agreement" primitive, that is, there are accepting locations $\mathcal{D}$ in $\mathbf{TA}^n$, and there is a common part in $\mathbf{TA}^n$ as shown in Fig. 5;

1) Sufficient conditions for (A) protocols: In this category of protocols, there is no "decide" action, and the *Almost-sure Termination* property is stated as "the probability of not all correct processes having the same value in R round is $O(2^{-R})$". The proof target of *Almost-sure Termination* property for (A) protocols can be formalized as follows. For every initial configuration c and every round-rigid adversary a, the following holds:

$$\mathbb{P}_a^c[\ \exists k \in \mathbb{N}_0, \exists v \in \{0,1\}.\ \mathbf{G} \bigwedge_{\ell \in \mathcal{F}^n_{1-v}} \kappa[\ell,k] = 0\ ] = 1 \quad (1)$$

We need two sufficient conditions to prove their *Almost-sure Termination* property under round-rigid schedules:

- $(C1)$ states the existence of a positive probability lower bound for all processes ending round k with the same final values. Formally, if there is a probability bound $p \in (0,1]$, such that for every round-rigid adversary a, every $k \in \mathbb{N}_0$, and every configuration c_k that is initial for round k, it holds that

$$\mathbb{P}_a^{c_k}[\ \exists v \in \{0,1\}.\ \mathbf{G} \bigwedge_{\ell \in \mathcal{F}^n_{1-v}} \kappa[\ell,k] = 0\ ] \geq p\ .$$

- $(C2)$ states that if all correct processes start round k with the same value $v \in \{0,1\}$, then they will all end with v in that round. Formally, $\forall v \in \{0,1\}, \forall k \in \mathbb{N}_0$.

$$\mathbf{A}(\mathbf{G} \bigwedge_{\ell \in \mathcal{I}^n_v} \kappa[\ell,k] = 0 \ \rightarrow\ \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}^n_v} \kappa[\ell',k] = 0)\ .$$

Combining conditions $(C1)$ and $(C2)$, under every round-rigid adversary, from any initial configuration of round k, the probability that all correct processes end with the same value in that round is at least p, and it holds for any future round. Thus, the probability not to have the same value within n rounds is at most $(1-p)^n$, which tends to 0 when n tends to infinity.

Proposition 2 *If Sys $\models (C1)$ and Sys $\models (C2)$,then Sys $\models$ (1).*

The proof is trivial.

2) Sufficient conditions for (B) protocols: The protocols in this category can decide a binary value, and their messages contain only binary values. Their desired *Almost-sure Termination* property requires that all correct processes decide the same value $v \in \{0,1\}$ with probability 1, and it can be stated as follows:

For every initial configuration c and every round-rigid adversary a, the following holds:

$$\mathbb{P}_a^c[\ \exists k \in \mathbb{N}_0, \exists v \in \{0,1\}.\ \mathbf{G} \bigwedge_{\ell \in \mathcal{F}^n \setminus \mathcal{D}^n_v} \kappa[\ell,k] = 0\ ] = 1 \quad (2)$$

Similarly we have to introduce three sufficient conditions to prove this specification:

- $(C1)$ states the existence of a positive probability lower-bound for all processes ending round k with the same final values. Formally, if there is a probability bound $p' \in (0,1]$, such that for every round-rigid adversary a, every $k \in \mathbb{N}_0$, and every configuration c_k that is initial for round k, it holds that

$$\mathbb{P}_a^{c_k}[\ \exists v \in \{0,1\}.\ \mathbf{G} \bigwedge_{\ell \in \mathcal{F}^n_{1-v}} \kappa[\ell,k] = 0\ ] \geq p\ .$$

- $(C2')$ states the existence of a positive probability lower-bound for all correct processes deciding the same value $v \in \{0,1\}$ in round k if all correct processes start round k with the same value v. Formally, if there is a probability bound $p' \in (0,1]$, such that for every round-rigid adversary a, every $k \in \mathbb{N}_0$, and every configuration c_k that is initial for round k, it holds that:
$\forall v \in \{0,1\}, \forall k \in \mathbb{N}_0$.

$$\mathbb{P}_a^{c_k}[\mathbf{G} \bigwedge_{\ell \in \mathcal{I}^n_{1-v}} \kappa[\ell,k] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell \in \mathcal{F}^n \setminus \mathcal{D}^n_v} \kappa[\ell',k] = 0] \geq p'.$$

Note that local-coin-based protocols with "decide" steps in [5] share the same sufficient condition $(C2)$ for probabilistic termination with category (A) protocols, while category (B) protocols require a probabilistic condition $(C2')$. The difference lies in the guard of such "decide" steps: in a local-coin-based protocol, a correct process decides a binary value

when it receives enough messages of certain type; however, in category (B) (and also (C)) protocols, it additionally requires the value equals the common coin (see Line 10 in Fig. 1).

Proposition 3 *Assume that there are accepting locations $\mathcal{D}$ in $\mathbf{TA}^n$, and for every pair of shared variables on the same messages $s_0, s_1 \in \Gamma$ and every final configuration $c_f \in \mathcal{F}$, $c_f.s_0 + c_f.s_1 \leq nc$. If Sys $\models (C1)$ and Sys $\models (C2')$, then Sys $\models (2)$.*

Proof *Let us fix the environment Env $= (\Pi, RC, N)$, an initial configuration c_0 of Sys and a round-rigid adversary a.*

Two possible options may occur along a path $\pi \in$ paths(c_0, a):

(a) $\exists v \in \{0,1\}.\pi \models \mathbf{G}(\bigwedge_{\ell \in \mathcal{F}_v^n} \kappa[\ell, 0] = 0)$,
(b) $\forall v \in \{0,1\}.\pi \models \mathbf{F}(\bigvee_{\ell \in \mathcal{F}_v^n} \kappa[\ell, 0] > 0)$,

In words, either round 0 ends with a final configuration where all correct processes have the same value, or round 0 ends with a final configuration where both 0 and 1 present. In case (a), we have $\pi \models \mathbf{G}(\bigwedge_{\ell \in \mathcal{F}_v^n} \kappa[\ell, 0] = 0)$ and by $(C1)$, for round $k = 0$, the probability that this case happens is at least p.

Consider the next round. By Round switch lemma (RS), we have that in round 1 all correct processes start with the same value v, two possible options may occur:

(c) $\pi_0 \models \mathbf{G} \bigwedge_{\ell \in \mathcal{F}^n \setminus \mathcal{D}_v^n} \kappa[\ell', 1] = 0$,
(d) $\pi_0 \models \mathbf{F} \bigvee_{\ell \in \mathcal{F}^n \setminus \mathcal{D}_v^n} \kappa[\ell', 1] > 0$,

In words, either all correct processes decide v in this round, or it ends with a final configuration where not all correct processes decide. According to $(C2')$, for round 1 the probability that case (c) happens is at least p', if case (a) happens in round 0.

Combine the first two rounds together. If case (a) happens in round 0 and case (c) happens in round 1, all correct processes decide the same binary value at the end of round 1, and its probability is at least $p \cdot p'$. Therefore the probability that not all correct processes decide the same binary value at the end of first two rounds is at most $(1 - p \cdot p')$.

By iterating the reasoning, consider the event $\mathcal{E}(2r)$: not all correct processes decide the same binary value at the end of first $2r$ rounds. Thus for round $2n$, we have that

$$\mathbb{P}_a^{c_0}[\mathcal{E}(2n)] \leq (1 - p \cdot p')^n$$
$$\lim_{n \to +\infty} \mathbb{P}_a^{c_0}[\mathcal{E}(2n)] = 0 \tag{3}$$

The limit when n tends to infinity yields that the probability of not having round-rigid termination is 0. In conclusion, all correct processes decide the same binary value with probability 1. □

3) Sufficient conditions for (C) protocols: The *Almost-sure Termination* property of (C) protocols shares the same formula as (2). However, they are based on *Binary Crusader Agreement*, a weaker version of the binary consensus, which introduces the third value $\perp$ for "not sure about 0 or 1", and the value $\perp$ is usually encoded by sending/receiving valid messages containing both values 0 and 1.

Fig. 5 shows the common structure of a single-round non-probabilistic threshold automaton for correct processes in the (C) protocols. Besides the final locations $\mathcal{F} = \{E_0, E_1, D_0, D_1\}$, there are 3 locations representing the output of *Binary Crusader Agreement* primitive. We denote the set of locations $\{M_0, M_1, M_\perp\}$ by $\mathcal{M}$. When receiving enough messages containing either a binary value or $\perp$ tagged with type M, it enters a location in $\mathcal{M}$ based on the number of messages received (encoded by variables m_0, m_1 and $m_\perp$). There are two templates for the guard of r_3: $m_0 + m_1 \geq n - t - f$, and $m_0 + m_1 + m_\perp \geq n - t - f$. The transitions $r_5 - r_{10}$ are all coin-based rules. r_5, r_8, r_9 share the same coin-based guard $cc_0 > 0$, which stands for the common coin result 0, while r_6, r_7, r_{10} share the same coin-based guard $cc_1 > 0$.

Here we introduce the property *binding*, which is proposed in [7]. Abstractly, a protocol obtains the binding property if no matter what the adversary does, it is forced to choose (bind to) in the present in a way that restricts all future outcomes of the protocol.

Definition 4 (Binding) *Let time τ be the first time such that there is a party that is correct and enters a location in $\mathcal{M}$ at time τ. At time τ, there is a value $b \in \{0,1\}$ such that no correct party enters M_{1-b} in any extension of this execution.*

Formally, given a finite path π, we denote the set of all paths that take π as its prefix by Exts(π). For every initial configuration c_0, every round-rigid adversary a and every round k, the following holds:

$$\forall \pi \in \textit{paths}(c_0, a).\ \pi \models \mathbf{F} \bigvee_{\ell \in \mathcal{M}} \kappa[\ell, k] > 0 \ \rightarrow$$
$$(\exists b \in \{0,1\}, \forall \pi' \in \textit{Exts}(\pi).\ \pi' \models \mathbf{G}\kappa[M_{1-b}, k] = 0)$$

The *binding* property states that for every execution prefix that some correct processes enter a location in $\mathcal{M}$ in round k, there is a single binary value b such that no correct process can enter location M_{1-b} in this round in any future extension of this prefix. Note that this is an instance of a *hyperproperty* because it characterizes sets of executions, i.e., all possible extensions of a prefix, instead of individual executions as in standard safety or liveness properties.

It is natural to require that if a correct process enters M_0 in round k then no correct process ever enters M_1, that is, $\mathbf{A}(\mathbf{F}\kappa[M_0, k]{>}0 \rightarrow \mathbf{G}\kappa[M_1, k]{=}0)$ (and similarly for M_1). However, it is different for the cases where a correct process enters $M_\perp$, because the transition r_3 from S to $M_\perp$ concerns the total number of messages instead of the exact number of each message, and the propositions on the numbers of messages are not supported in the threshold automata approach. Therefore, we propose a solution to further refine the model, particularly the transition rule to $M_\perp$, and encode the *binding* property by five sufficient conditions.

Assume that $r_3 = (S, M_\perp, \varphi, \mathbf{0})$, that is, it is a transition rule from location S to location $M_\perp$, guarded by formula φ, and it keeps the shared variables unchanged. We can remove r_3 and instead add three locations N_0, N_1, and $N_\perp$ as well as the following transition rules:

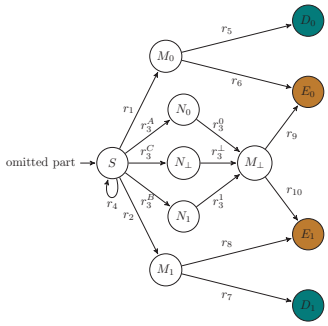


Fig. 6: A Refined Model of the Common Part

- $r_3^A = (S, N_0, \varphi \wedge m_0 > 0, \mathbf{0})$
- $r_3^B = (S, N_1, \varphi \wedge m_1 > 0, \mathbf{0})$
- $r_3^C = (S, N_\perp, \varphi \wedge m_0 = 0 \wedge m_1 = 0, \mathbf{0})$
- $r_3^i = (N_i, M_\perp, \mathit{true}, \mathbf{0})$, for $i \in \{0, 1, \perp\}$

It is easy to observe that the modification does not block the automaton. By reasoning about the counter values of location N_0, N_1 and $N_\perp$, we can infer the number of messages received, which is impossible in the original automaton.

Here we list the sufficient conditions for *binding* property as follows:

- *(CB0)*: if a correct process enters M_0 in round k, then no correct process ever enters M_1, that is, $\forall k \in \mathbb{N}_0$. $\mathbf{A}(\mathbf{F}\kappa[M_0, k] > 0 \rightarrow \mathbf{G}\kappa[M_1, k] = 0)$;
- *(CB1)*: if a correct process enters M_1 in round k, then no correct process ever enters M_0, that is, $\forall k \in \mathbb{N}_0$. $\mathbf{A}(\mathbf{F}\kappa[M_1, k] > 0 \rightarrow \mathbf{G}\kappa[M_0, k] = 0)$;
- *(CB2)*: if a correct process enters N_0 and then $M_\perp$ in round k, then no correct process ever enters M_1, that is, $\forall k \in \mathbb{N}_0$. $\mathbf{A}(\mathbf{F}\kappa[N_0, k] > 0 \rightarrow \mathbf{G}\kappa[M_1, k] = 0)$;
- *(CB3)*: if a correct process enters N_1 and then $M_\perp$ in round k, then no correct process ever enters M_0, that is, $\forall k \in \mathbb{N}_0$. $\mathbf{A}(\mathbf{F}\kappa[N_1, k] > 0 \rightarrow \mathbf{G}\kappa[M_0, k] = 0)$;
- *(CB4)*: if a correct process enters $N_\perp$ and then $M_\perp$ in round k, then no correct process ever enters M_0 or M_1, that is, $\forall k \in \mathbb{N}_0$. $\mathbf{A}(\mathbf{F}\kappa[N_\perp, k] > 0 \rightarrow \mathbf{G}\bigwedge_{\ell \in \{M_0, M_1\}} \kappa[\ell, k] = 0)$;

Proposition 4 *If TA^n has such a subpart as Fig. 6 and Sys satisfies all (CB0)-(CB4), then Sys satisfies Binding property.*

Proof *Let time τ be the first time such that there is a party that is correct and enters a location in $\mathcal{M}$ at time τ. There are five possible options for the correct process:*

- *it enters M_0 through the transition rule r_1, then by (CB0) we can set the binary value $b = 0$ and no correct process can enter location $M_{1-b} = M_1$, that is, $\mathbf{G}\kappa[M_1] = 0$;*
- *it enters M_1 through the transition rule r_2, then by (CB1) we can set the binary value $b = 1$ and no correct process can enter location $M_{1-b} = M_0$, that is, $\mathbf{G}\kappa[M_0] = 0$;*
- *it enters $M_\perp$ through the transition rule r_3^A, then by (CB2) we can set the binary value $b = 0$ and no correct process can enter location $M_{1-b} = M_1$;*
- *it enters $M_\perp$ through the transition rule r_3^B, then by (CB3) we can set the binary value $b = 1$ and no correct process can enter location $M_{1-b} = M_0$;*
- *it enters $M_\perp$ through the transition rule r_3^C, then by (CB4) we can set the binary value $b = 0$ and no correct process can enter location $M_{1-b} = M_1$;*

To conclude, at time τ, there must be a value $b \in \{0, 1\}$ such that no correct party enters M_{1-b} in any extension of this execution. □

Proposition 5 *Assume that there are accepting locations $\mathcal{D}$ in **TA**^{n}, and there is a subpart in **TA**^{n} as shown in Fig. 6. If Sys satisfies Binding property as well as $(C2')$, then Sys $\models$ (2).*

Proof *Fix the environment Env $= (\Pi, RC, N)$, a round-rigid adversary a and an initial configuration c_0. Assume that the common coin is ϵ-Good, that is, for any value $v \in \{0, 1\}$, the common coin result equals v with probability $\geq \epsilon > 0$.*

Consider any round k, let time τ_k be the first time such that there is a process that is correct and enters a location in $\mathcal{M}$ at time τ_k. By binding property, at time τ_k we can have a binary value b such that no correct process can ever enter M_{1-b} in this round. Because all coin-based rules $(r_5 - r_{10})$ start from a location in $\mathcal{M}$, no correct process has ever executed a coin-based rule at time τ_k. In other words, at time τ_k the common coin of round k is not yet tossed, therefore the common coin result is independent of the binary value b. Note that the common coin result has at least ϵ possibility of being value b. If no correct process can ever enter M_{1-b} in round k and the common coin result is b, all correct processes will have the same value b at the end of round k. To conclude, by binding property, there is a positive probability lower bound ϵ for all correct process ending round k with the same final value, and it coincides the condition $(C1)$.

Follow the same aforementioned reasoning, and we prove that with probability 1 all correct processes decide the same binary value. □

Corollary 1 *Assume that there are accepting locations $\mathcal{D}$ in **TA**^{n}, and there is a subpart in **TA**^{n} as shown in Fig. 5. If Sys satisfies all (CB0)-(CB4) conditions, as well as $(C2')$, then Sys $\models$ (2).*

Proof *It follows directly by Propositions 4 and 5.* □

4) Reducing probabilistic to non-probabilistic specifications: Note that the sufficient conditions $(C2)$, as well as *(CB0)-(CB4)*, are non-probabilistic specifications with one round number, so that we can check them using the method in Sect. V-A; while $(C1)$ and $(C2')$ are probabilistic, we need to further reduce its verification to verification of non-probabilistic specifications.

The method for local coins in [5] requires that there exists at most one "coin-based" rule in a threshold automaton and it must lie at the end of a round. We remove those restrictions on the "common-coin-based" rules. In fact, it is even allowed that a single-round path in a threshold automaton contains multiple "common-coin-based" rules. This makes the extended model

more expressive, while it keeps good properties, such as the following lemma.

Lemma 2 *Let Env be the environment, $\mathcal{A}^R$ be the set of round-rigid adversaries and $\mathcal{I}$ be the set of initial configurations over* Env. *In the single-round probabilistic counter system Sys($\mathbf{TA}^n_{rd}, \mathbf{TA}^c_{rd}$), for every* LTL$_{-\mathrm{X}}$ *formula φ over atomic proposition AP, the following statements are equivalent:*

- $\exists p > 0, \forall c \in \mathcal{I}, \forall a \in \mathcal{A}^R.\ \mathbb{P}^c_a(\varphi) \geq p$,
- $\forall c \in \mathcal{I}, \forall a \in \mathcal{A}^R, \exists \pi \in \mathit{paths}(c, a).\ \pi \models \varphi$

The proof is omitted due to length constraints.

By applying Lemma 2, we observe that the probabilistic sufficient conditions are equivalent to non-probabilistic specifications with the existential quantifier in the single round system Sys($\mathbf{TA}^n_{rd}, \mathbf{TA}^c_{rd}$). We can further turn them into non-probabilistic specifications that can be checked by ByMC, and in the end prove the sufficient conditions $(C1)$ and $(C2')$.

VI. Experiments

We have applied our approach to eight randomized fault-tolerant consensus protocols that make use of common coins, including:

1) Randomized Byzantine consensus (`Rabin83`) in [2], the first common coin-based randomized consensus protocol. It tolerates Byzantine faults when $t < n/10$.
2) Randomized Byzantine consensus (`CC85(a)`) in [16], which proposes a simple implementation of common coin and has an optimal resilience condition $n > 3t$.
3) Randomized Byzantine consensus (`CC85(b)`) in [16], which is an adaptation of `Rabin83` and raises the bound of Byzantine faults to $t < n/6$.
4) Randomized Byzantine agreement (`FMR05`) in [17], which contains one communication step in each round, and can resist up to $n/5$ Byzantine faults.
5) Randomized Byzantine agreement (`KS16`) in [18], which builds on Bracha's [19] and replaces the local coin in each process with a common coin. Its resilience condition remains $n > 3t$.
6) The protocol (`MMR14`) by Mostéfaoui et al. [13], which contains an attack by adaptive adversary resulting in non-termination.
7) In Miller's post [9] there is a discussion to fix the bug of `MMR14`, and the fixed version (`Miller18`) was later used in the Dumbo protocol [20].
8) Randomized Byzantine agreement (`ABY22`) in [7] based on binding crusader agreement, which tolerates t Byzantine faults when $n > 3t$.

For each protocol, we build two versions of one-round threshold automata for probabilistic and non-probabilistic properties. We input both automata into ByMC, which implements the parameterized model checking techniques.

Table III gives a summary of the properties and conditions that were verified in our experiments. The round invariants inv1 and inv2 are sufficient conditions for *Agreement* and *Validity*, and the others are sufficient conditions for *Almost-sure termination*. Given the set of all possible locations $\mathcal{L}$ and a subset $S = \{\ell_1, \ldots, \ell_n\}$, we adopt the shorthand notation for LTL$_{-\mathrm{X}}$ formulae: $\mathrm{EX}\{S\}$ stands for $\bigvee_{\ell \in S} \kappa[\ell] \neq 0$, that is, at least one threshold automaton enters a location in S; $\mathrm{ALL}\{S\}$ stands for $\bigwedge_{\ell \in \mathcal{L} \setminus S} \kappa[\ell] = 0$, that is, all threshold automata enter the locations in S. Take (*Inv2*) as example, its formula states that for every execution where all correct processes propose value 0 in a round, it is always true that no process can end with value 1 in that round.

Table II shows the computational results of the experiments: column $|\mathcal{L}|$ and $|\mathcal{R}|$ give the numbers of automata locations and rules. In the columns for properties, "nschemas" stands for the number of checked schemas of executions. The computational times are given in seconds if not specified, and "CE" stands for a reported counterexample. The benchmarks of category (C) are challenging for the technique of Konnov et al. [4]: the sizes of their threshold automata are too large, and as a result their experiments timeout for a 24-hour limit. Therefore we check the first six rows with a laptop with Intel Core i7-12650H, while the last two experiments are conducted on a computing server of 216 cores AMD EPYC 7702 in parallel mode. We ran each experiments 10 times and listed the average values for nschemas and time.

As can be seen in Table II, the benchmarks of categories (A) and (B) have small sizes and they can be quickly verified for the three properties. A violation of *Binding* sufficient conditions is found for `MMR14` within 10 seconds at best.The found counterexample contains the system settings, e.g. $n = 193$ and $t = 64$, an initial configuration as well as a sequence of actions. It reports that the resulted state does not satisfy the formula of *(CB2)*. We analyze its execution and find that it follows the pattern of the designed attack in [9]: firstly some correct processes access the common coin, then the adaptive adversary obtains the coin value v and manipulates the messages, and finally a correct process gets its new *est* value as $1 - v$. By repeating the last actions we can complete the duality in the attack. `Miller18` and `ABY22`, two fixed versions of `MMR14`, both pass the check, while it takes over 11 and 10 hours to check all their sufficient conditions for *Almost-sure Termination* properties.

Table II reveals a positive correlation between the number of checked schemas and the verification time, and the worst-case schemas depend on the size of the automaton and the structure of the formula. Given that a threshold automaton is tightly coupled with its protocol, reducing the number of locations and transition rules is not straightforward without sacrificing essential meaning. Another critical factor affecting the maximum number of schemas is the presence of milestones. Intuitively, when a milestone is reached, certain threshold guards (inequations) are always true/false in the future executions. Interestingly, different types of messages lead to varying numbers of milestones. As an illustrative example, we modify `ABY22` to create 5 threshold automata of the same size but with different milestone counts. The maximum numbers of schemas are then calculated and presented in Table IV.

TABLE II: Benchmarks of 8 Different Common Coin-Based Protocols

Automaton				Agreement		Validity		A.S. Termination	
Name	category	$\|\mathcal{L}\|$	$\|\mathcal{R}\|$	nschemas	time	nschemas	time	nschemas	time (total)
Rabin83	(A)	7	17	6	0.25	2	0.20	8	0.43
CC85(a)	(B)	9	18	342	4.93	42	0.50	171.5	2.70
CC85(b)	(B)	10	17	6	0.25	2	0.20	8	0.32
FMR05	(B)	10	16	6	0.23	2	0.21	2	0.32
KS16	(B)	11	26	18	0.75	5	0.31	15	0.76
MMR14	(C)	17	29	28918	298.90	1442	8.74	-	CE
Miller18(mpi)	(C)	22	48	$> 10^6$	605	253534	226	$> 10^8$	11h46m47s
ABY22(mpi)	(C)	22	49	$> 10^6$	583	106098	71	$> 10^8$	10h13m14s

TABLE III: Properties Checked for Value 0

Label	Formula
(*Inv1*)	$\mathbf{A}\ \mathbf{F}(\text{Ex}\{D_0\}) \rightarrow \mathbf{G}\ (\neg\text{Ex}\{E_1, D_1\})$
(*Inv2*)	$\mathbf{A}\ \text{ALL}\{I_0\} \rightarrow \mathbf{G}\ (\neg\text{Ex}\{E_1, D_1\})$
$(C1)$	$\mathbf{A}\ \mathbf{F}(\text{Ex}\{D_0, E_0\}) \rightarrow \mathbf{G}(\neg\text{Ex}\{D_1, E_1\})$
(CB0)	$\mathbf{A}\ \mathbf{F}(\text{Ex}\{M_0\}) \rightarrow \mathbf{G}(\neg\text{Ex}\{M_1\})$
...	...

TABLE IV: Maximum Numbers of Schemas for Threshold Automata with Different Milestones

Name	Formula	nmilestones	max-nschemas
ABY22	*(CB0)*	10	98182294
ABY22-1	*(CB0)*	9	15129955
ABY22-2	*(CB0)*	8	2650445
ABY22-3	*(CB0)*	7	257126
ABY22-4	*(CB0)*	6	28918
ABY22	(*Inv2*)	10	7479057
ABY22-1	(*Inv2*)	9	1298630
ABY22-2	(*Inv2*)	8	253534
ABY22-3	(*Inv2*)	7	28395
ABY22-4	(*Inv2*)	6	3592

VII. Related Work

In recent years, there have been steady advance in creating ever more complex distributed consensus protocols, in order to optimize for performance while preserving its guarantees of fault-tolerance. HoneyBadger [21] is the first practical asynchronous atomic broadcast protocol. The Dumbo family of protocols [20], [22] build upon HoneyBadger to obtain further improvements in performance and latency. All these protocols use asynchronous Byzantine agreement as a substep. The increasing complexity of these protocols make it more urgent to develop scalable formal techniques for their verification.

There have been a long line of work on verification of fault-tolerant distributed protocols. Earlier work [3], [23] make use of Cadence SMV and the probabilistic model checker PRISM to verify protocols consisting of 10-20 processes. For parameterized verification, the key development is the proposal of threshold automata, and theoretical results reducing the correctness in the general case to model checking on a finite system [4], [11]. These results are implemented in the ByMC tool [12] for verifying protocols in practice.

Bertrand et al. [5] extend the framework of threshold automata with probabilistic transitions, namely probabilistic threshold automata, and reduce the safety and liveness verification of a multi-round randomized protocol to checking on a one-round automaton. In that paper, round-rigid adversaries are assumed in the proof of termination. This restriction is relaxed to that of weak adversaries in [24].

In the direction of compositional reasoning, the work by Bertrand et al. [6] verified the consensus protocol of Red Belly Blockchain by reducing the protocol into two parts: an inner broadcast protocol and an outer decision protocol, verified each part separately and then composed the correctness results.

Besides the use of threshold automata, there is also work using Ivy and TLA proof system to verify a simplified version of HotStuff [25], a protocol for repeated distributed consensus used in permissioned blockchains [26]. The work by Attiya et al. [27] focused on verifying Byzantine fault-tolerant protocols based on directed acyclic graphs (DAGS), some of which also make use of randomness in the form of common coins.

VIII. Conclusion

In this paper, we proposed an extension of probabilistic threshold automata that supports the use of common coins, that can be used to model fault-tolerant randomized distributed protocols for reaching consensus. We reduce the correctness properties of consensus protocols: agreement, validity, and termination to queries on single-round non-probabilistic threshold automaton, which can be checked using ByMC. Key parts of the queries correspond to the binding condition preventing an adaptive adversary from delaying decision by the protocol indefinitely. Using this framework, we verified eight consensus protocols that make use of common coins, and were able to reproduce an attack found in earlier work.

Our work can be viewed as a further step toward verifying complex Byzantine consensus protocols, such as HoneyBadger [21] and Dumbo [20], both of which make use of asynchronous Byzantine agreement. Future work includes the development of composition reasoning techniques as well as reasoning about cryptographic primitives that are part of those protocols. Finally, we intend to remove the restriction of round-rigid adversaries in the proof of termination, perhaps by using methods similar to those proposed in [24].

Acknowledgment

This work is supported by CAS Project for Young Scientists in Basic Research, Grant No.YSBR-040, ISCAS New Cultivation Project ISCAS-PYFX-202201, and ISCAS Basic Research ISCAS-JCZD-202302.

This work is part of the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant no. 101008233.

REFERENCES

[1] M. Ben-Or, "Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract)," in *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*. Montreal Quebec Canada: ACM, 1983.

[2] M. O. Rabin, "Randomized byzantine generals," in *24th Annual Symposium on Foundations of Computer Science*. Tucson, Arizona, USA: IEEE, 1983.

[3] M. Z. Kwiatkowska and G. Norman, "Verifying randomized byzantine agreement," in *Formal Techniques for Networked and Distributed Systems - FORTE 2002*. Houston, Texas, USA: Springer, 2002.

[4] I. V. Konnov, M. Lazic, H. Veith, and J. Widder, "A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms," in *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017*. Paris, France: ACM, 2017.

[5] N. Bertrand, I. Konnov, M. Lazic, and J. Widder, "Verification of randomized consensus algorithms under round-rigid adversaries," *Int. J. Softw. Tools Technol. Transf.*, vol. 23, no. 5, pp. 797–821, 2021.

[6] N. Bertrand, V. Gramoli, I. Konnov, M. Lazic, P. Tholoniat, and J. Widder, "Holistic verification of blockchain consensus," in *36th International Symposium on Distributed Computing, DISC 2022*, Augusta,Georgia, USA, 2022.

[7] I. Abraham, N. Ben-David, and S. Yandamuri, "Efficient and adaptively secure asynchronous binary agreement via binding crusader agreement," in *PODC '22: ACM Symposium on Principles of Distributed Computing*. Salerno, Italy: ACM, 2022.

[8] 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.

[9] A. Miller, "Bug in aba protocol's use of common coin," last Accessd 8 Sep 2023. [Online]. Available: https://github.com/amiller/HoneyBadgerBFT/issues/59

[10] I. Abraham, G. Gueta, D. Malkhi, L. Alvisi, R. Kotla, and J. Martin, "Revisiting fast practical byzantine fault tolerance," *CoRR*, vol. abs/1712.01367, 2017. [Online]. Available: http://arxiv.org/abs/1712.01367

[11] I. V. Konnov, H. Veith, and J. Widder, "On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability," *Inf. Comput.*, vol. 252, pp. 95–109, 2017. [Online]. Available: https://doi.org/10.1016/j.ic.2016.03.006

[12] I. Konnov and J. Widder, "Bymc: Byzantine model checker," in *Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems - 8th International Symposium, ISoLA*. Limassol, Cyprus: Springer, 2018.

[13] A. Mostefaoui, H. Moumen, and M. Raynal, "Signature-free asynchronous byzantine consensus with t $<$ n/3 and o(n2) messages," in *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, 2014.

[14] A. Mostéfaoui, H. Moumen, and M. Raynal, "Signature-free asynchronous binary byzantine consensus with t $<$ n/3, o(n2) messages, and O(1) expected time," *J. ACM*, vol. 62, no. 4, pp. 31:1–31:21, 2015.

[15] L. Lamport, R. E. Shostak, and M. C. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982. [Online]. Available: https://doi.org/10.1145/357172.357176

[16] B. Chor and B. A. Coan, "A simple and efficient randomized byzantine agreement algorithm," *IEEE Trans. Software Eng.*, vol. 11, no. 6, pp. 531–539, 1985.

[17] R. Friedman, A. Mostéfaoui, and M. Raynal, "Simple and efficient oracle-based consensus protocols for asynchronous byzantine systems," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 1, pp. 46–56, 2005.

[18] V. King and J. Saia, "Byzantine agreement in expected polynomial time," *J. ACM*, vol. 63, no. 2, pp. 13:1–13:21, 2016.

[19] G. Bracha, "Asynchronous byzantine agreement protocols," *Inf. Comput.*, vol. 75, no. 2, pp. 130–143, 1987.

[20] 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. [Online]. Available: https://doi.org/10.1145/3372297.3417262

[21] 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, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 31–42. [Online]. Available: https://doi.org/10.1145/2976749.2978399

[22] 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

[23] M. Z. Kwiatkowska, G. Norman, and R. Segala, "Automated verification of a randomized distributed consensus protocol using cadence SMV and PRISM," in *Computer Aided Verification, 13th International Conference*. Paris, France: Springer, 2001.

[24] N. Bertrand, M. Lazic, and J. Widder, "A reduction theorem for randomized distributed algorithms under weak adversaries," in *Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings*, ser. Lecture Notes in Computer Science, F. Henglein, S. Shoham, and Y. Vizel, Eds., vol. 12597. Springer, 2021, pp. 219–239. [Online]. Available: https://doi.org/10.1007/978-3-030-67067-2_11

[25] L. Jehl, "Formal verification of hotstuff," in *Formal Techniques for Distributed Objects, Components, and Systems - 41st IFIP WG 6.1 International Conference, FORTE 2021, Held as Part of the 16th International Federated Conference on Distributed Computing Techniques, DisCoTec 2021, Valletta, Malta, June 14-18, 2021, Proceedings*, ser. Lecture Notes in Computer Science, K. Peters and T. A. C. Willemse, Eds., vol. 12719. Springer, 2021, pp. 197–204. [Online]. Available: https://doi.org/10.1007/978-3-030-78089-0_13

[26] 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*, P. Robinson and F. Ellen, Eds. ACM, 2019, pp. 347–356. [Online]. Available: https://doi.org/10.1145/3293611.3331591

[27] H. Attiya, C. Enea, and S. Nassar, "Faithful Simulation of Randomized BFT Protocols on Block DAGs," in *34th International Conference on Concurrency Theory (CONCUR 2023)*, Dagstuhl, Germany, 2023.