A Reputation Management Framework for Knowledge-Based and Probabilistic Blockchains

Tara Salman
Dept. of Computer Science & Engineering
Washington University in St. Louis
St. Louis, USA
tara.salman@wustl.edu

Raj Jain
Dept. of Computer Science & Engineering
Washington University in St. Louis
St. Louis, USA
jain@cse.wustl.edu

Lav Gupta
Dept. of Math and Computer Science
University of Missouri
St. Louis, USA
lavgupta@missouri.edu

***Abstract*—Recently, leading research communities have been investigating the use of blockchains for Artificial Intelligence (AI) applications, where multiple participants, or agents, collaborate to make consensus decisions. To achieve this, the data in the blockchain storage have to be transformed into blockchain knowledge. We refer to these types of blockchains as *knowledge-based blockchains*. Knowledge-based blockchains are potentially useful in building efficient risk assessment applications. An earlier work introduced *probabilistic blockchain* which facilitates knowledge-based blockchains.**

This paper proposes an extension for the probabilistic blockchain concept. The design of a *reputation management framework,* suitable for such blockchains, is proposed. The framework has been developed to suit the requirements of a wide range of applications. In particular, we apply it to the detection of *malicious nodes* and reduce their effect on the probabilistic blockchains' consensus process. We evaluate the framework by comparing it to a baseline using several adversarial strategies. Further, we analyze the collaborative decisions with and without the malicious node detection. Both results show a sustainable performance, where the proposed work outperforms others and achieves excellent results.

Keywords—blockchains, knowledge-based blockchains, probabilistic blockchains, reputation management framework, Rated Proportional Multi-Configurable Exponential Weighted Average, RPMC-EWA, malicious node detection.

I. INTRODUCTION

Recent advances in networking, big data, cloud computing, and data analysis techniques have made several large-scale distributed applications possible. However, security and trustworthiness issues are major concerns for such applications. Blockchains, as distributed, peer-to-peer, and immutable networks can save these applications from their security and trustworthiness issues [1-5].

Lately, there have been some efforts toward utilizing blockchains for decision support applications such as voting, predictions, and collaborative Artificial Intelligence (AI) systems. In 2015, the Bitcoin foundation funded a new project for building efficient, anomalous, and secure voting systems [6-10]. All in all, these efforts have introduced *the transformation of blockchains from a data storage platform to a collaborative processing and decision-support platform*. We refer to this type of blockchains as *knowledge-based blockchains*, i.e., accumulating useful knowledge from the blockchains [11].

Our earlier work in [12] introduced a paradigm for achieving collaborative decision-making required for knowledge-based blockchains. We refer to that as the *probabilistic blockchain paradigm*. Probabilistic blockchains do not imply that the chain is probabilistic, as is the case in [13], but that the collaborative decision is made probabilistically, given many individual decisions made with imperfect information. As an example, one can think of voters in a voting system or different AI-capable nodes in a blockchain-based AI system as *agents* of the system. These agents are nodes that inspect events and make individual decisions, votes, or predictions. Based on these individual decisions, the blockchain network can achieve a collaborative decision about the event being investigated. For the rest of the paper, we will refer to this collaborative decision as ***the consensus*** made using the blockchain.

Probabilistic blockchains, as explained further in Section II, can be used for many risk assessment or prediction application. Their applications span both FinTech and non-FinTech domains including stock market predictions, asset investment, insurance, loan granting, credit scoring, malware, and intrusion detection, and hotel (or any asset) recommendation systems. In other words, the probabilistic blockchain framework is a step toward building effective blockchain-based AI systems.

This paper extends probabilistic blockchains and proposes a *reputation management framework based on agents' performance* in the systems. The framework is initially designed to suit multiple application domains with different requirements. Accordingly, first, we discuss several applications of the proposed framework for probabilistic blockchains. We then apply it specifically to *malicious node detection*. We show that the framework outperforms traditional reputation systems in malicious node detection and results in good consensus.

The combination of probabilistic blockchains and the reputation framework has several advantages for decision-making applications. The collaborative decision in probabilistic blockchains is decentralized, concurred, and secure against manipulation. The use of a reputation framework helps in distinguishing good/expert nodes from bad/misbehaving/nonexpert ones. It can be used to incentivize good nodes and allow them to have a higher impact on the decision. This motivates participating nodes to make correct decisions and prevents any effort to control the decision process. That is, the proposed combination advances traditional decision-making by providing more secure and reliable systems.

978-1-7281-4693-5/19/$31.00 ©2019 IEEE
DOI 10.1109/Blockchain.2019.00078

The rest of the paper is organized as follows: In Section II, we give a brief overview of blockchains and their extensions to probabilistic blockchains. In Section III, we put forward some requirements for reputation frameworks. In Section IV, we propose a particular reputation framework and prove that it meets the requirements. Section V presents some applications of the proposed framework to probabilistic blockchains. Section VI discusses the performance analysis, comparisons, and results. Finally, Section VII provides conclusions and summarizes the paper.

A. Related Work

Reputation management frameworks have been investigated by the blockchain community. Their main purpose is to build reputation-based mining techniques that allow good nodes to create blocks and gain rewards. Gai et al. have discussed a Proof of Reputation (PoR) mining algorithm where the miner is chosen based on his reputation [14]. PoR has been applied to GoChain blockchain platform that is built on top of the Ethereum platform [15]. Another Proof of Reputations (PoRe) concept has been proposed by Qin in et al. [16]. PoRe combined with Proof of Work (PoW) and proof of stake has been applied to RPChain, a blockchain-based academic social network service.

RepuCoin, [17], is a blockchain system that uses miners' reputation to define the mining power in PoW algorithm. RepChain, [18], is another blockchain reputation system that uses reputations to decide on the leader and validators in leader-based mining algorithms. Agents will high reputations get a high chance to be selected as leaders. Thus, they get the opportunity to create blocks and gain rewards. Similar approaches have been applied in [19] and [20] with other mining algorithms.

These works have shown promising results in terms of security analysis and higher system throughput. Thus, reputation-based mining techniques have the potential of solving major transaction throughput and security issues found in other mining techniques.

To the best of the authors' knowledge, malicious node detection has not been investigated for blockchain applications. As knowledge-based blockchains are a relatively new concept, their malicious agents' detection has not been investigated. Traditional blockchain applications use the chain as a storage unit. Thus, malicious nodes do not affect the system except when adding or removing transactions from the database. This is made difficult by appropriate mining techniques, and so malicious node detection is not useful in traditional blockchains.

II. Probabilistic Blockchains Background

This section gives a brief background of the transition from blockchains to probabilistic blockchains. It discusses the main design and architectural changes introduced in that paradigm.

We assume some prior knowledge of the blockchain technology, the motivation to use it, and its main characteristics. Also, the analysis of the probabilistic blockchains has been discussed in [12] and is, therefore, out of the scope of this paper.

A. Traditional Blockchains Architecture

A blockchain consists of two main components: a network and a database. The blockchain database is a data structure that comprises transactions, blocks, and a chain. The system interactions, e.g., money or digital exchanges, are stored in the transactions. Transactions are verified and bundled together to form a block using a predefined technique. Several blocks are linked to each other where each block points to the hash of its predecessor. This creates a chain of blocks called the blockchain database. The blockchain network is a set of distributed nodes that maintain the database and are connected via a network.

B. Motivation For Probabilistic blockchains

In any collaborative decision-making application, there are many users that provide individual decisions about the events being inspected. As an example, consider a stock market prediction application. There are many companies that analyze stocks and predict their future performance. We could use the blockchain to store these forecasts. Moreover, the blockchain can be used to process these forecasts and achieve a consensus. A similar situation applies to group decision-making inside large corporations, where employees indicate their opinion about an important issue and the managers can benefit from the summary of those opinions. We call these users or employees as **agents**.

To achieve these applications, the blockchains should be extended in three directions. First, transactions and blocks should be able to store probabilistic decisions made by agents. Second, the blockchain network should be able to summarize decisions and achieve probabilistic consensus without the needs for external parties. Third, these consensuses should be visible to others and possibly updated whenever needed.

It should be noted that even if the traditional blockchains are able to store decisions, they are not being used to do so. Only simple deterministic "yes" or "no" decisions are made to check if 'a transaction is valid or not' or 'being present in the chain or not.' Therefore, the processing of transactions and blocks is still missing. Moreover, summarizing decisions and achieving consensus within the blockchain is not feasible.

C. Probabilistic Blockchains

Unlike traditional blockchains, probabilistic blockchains allow transactions to be simply the opinions of users. The blocks contain all such transactions along with a summary of the opinions. The summary may be a statistical summary like the mean or may be the result of a machine learning algorithm.

The transactions that were used to store contracts or bitcoin exchanges in traditional blockchains are used to record decisions in probabilistic blockchains. Each decision is made by an *agent i* for an *event j* and has a *probability* $p_{i,j}$. Therefore, a transaction in probabilistic blockchains has an event id, an agent id, a decision, and a probability $p_{i,j}$.

As with traditional blockchains, miners, or block formers, verify these transactions and form a candidate block containing several transactions. The added function for probabilistic blockchains consists of summarizing the decisions enclosed in the new block. The candidate block should have a summary of one or several events that are included in that block. The main differences between a traditional and a probabilistic blockchain database are illustrated in Fig. 1.

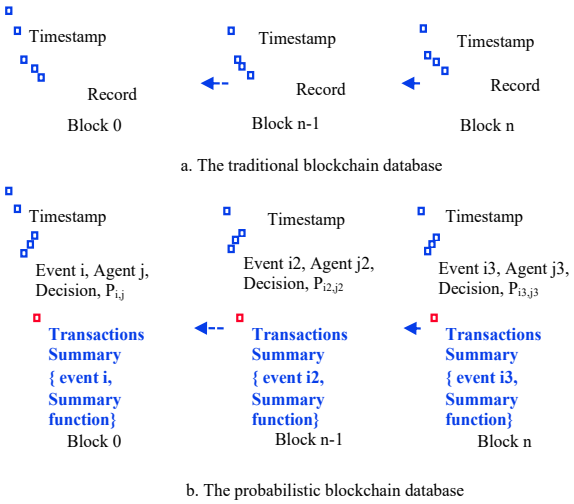

Fig. 1. Traditional and probabilistic blockchain database

The event summaries act as *the blockchain consensus* of the inspected events in that block. This consensus is based on a function that is application dependent and predefined in the system. We refer to this function as *the consensus function*. This function should be representative, incremental, easy to calculate, and difficult to manipulate. The property of being 'representative' indicates that the summary reflects the decision of a group rather than that of a single agent. For example, 'min' or 'max' summary are not considered proper consensus functions because they could easily be impacted by a single agent. The property of being 'incremental' indicates the new summary can be calculated by adding the newly added decisions for the same event to the previous summary value. This is required so that miners do not have to retrieve all decisions made for the same event in prior blocks. The function should be easy to calculate, as the mining processes are already complicated. Besides, it should be difficult to manipulate so that a malicious agent is not able to control the consensus. Any function satisfying these four conditions can be used. Examples of good functions include mean, the second moment, first n moments, or even the result of a sophisticated machine-learning algorithm.

After the candidate block is formed, it is broadcasted and verified by other blockchain nodes, as with the traditional blockchains. The added functions to block verifications are the correctness and validity of the consensus values. Thus, any block forming and verification technique (i.e., the mining process) that is used in traditional blockchains can also be used in probabilistic blockchains.

D. Need for Reputation Management

One of the issues in any group decision-making is that not all agents contributing to the group decision may be treated equally. Some agents, who have had a good history of correct decisions in the past, may be given a higher weight. In other words, the reputation of those contributing to the decision is important. Thus, this paper proposes a reputation assignment framework for the decision-making in probabilistic blockchains.

E. Probabilistic Blockchains with Reputation

The proposed reputation management system can be applied to any decision making or prediction engine with the same objectives. However, the use of probabilistic blockchain combined with the reputation framework has several advantages over the traditional systems. First, blockchains provide a distributed and immutable database where individual decisions are securely stored. Probabilistic blockchains add to that by allowing consensus decision to have the same features. Further, probabilistic blockchains come with many security advantages including resiliency to malicious agents, resiliency to malicious miners, Distributed Denial of Service (DDoS) protection, and fraud mitigation. The details of these can be found in [15].

A reputation management framework results in better consensus decision as good behaving agents have higher contributions to the decision. In addition to that, probabilistic blockchains have its own advantage to the framework. It can ensure that the framework is correctly followed and agents' reputations are not centralized. This is true since other nodes validate the reputations as will be discussed. That is, reputation and probabilistic blockchains collectively collaborate to build efficient decision-making engines.

III. Reputation Management Framework Requirements

Some requirements are needed to be met to properly design a reputation management framework suitable for probabilistic blockchains. This section will discuss seven such requirements and why they are necessary. It lays the foundation for the framework proposed in the next section. Our basic premise is that the requirements should favor better-performing agents and provide fairness among similar agents. Further, they should be configurable to be used for different applications.

A. Continuity

A reputation function should be continuous such that it gives a different value for any change in an agent's performance. That is, for any correct or incorrect decision made by an agent, a suitable reputation should be given. This allows the agent to have a reputation value at any point in the system. More importantly, it enables this reputation value to represent the agent's performance at any time.

B. Boundedness Function

A reputation function should be bounded between 0 and 1. That is, a good agent should eventually have a reputation of 1 while a malicious agent should have a reputation of 0. This allows the reputation to be expressed as a percentage. For example, an agent with a reputation of 0.9 is mostly trusted while an agent with a reputation of 0.1 is not trusted.

Boundedness helps to intuitively understand the reputation value based on the agent's performance. Moreover, it puts forward an expectation of how the agent will perform in the future. An agent with a 0.9 reputation is more likely to make correct decisions than an agent with a 0.1 reputation.

C. A Reputation Starts at 0.5

An agent that is just starting and has not contributed to the decision-making process should have a reputation of 0.5. This

makes the agent half trusted and gives it a 50% expectation to perform well in the future. It should be noted that this reputation value is lower than an excellent behaving agent but at the same time higher than a malicious or poorly performing agent.

D. Time-Dependent Function

The reputation function should be time-dependent, giving an agent the chance to enhance its reputation as he makes correct decisions. This indicates that an agent that has made wrong decisions recently should have a lower reputation than an agent that has made wrong decisions much earlier. This is desirable as the second agent has enhanced its performance while the first has made it worse. Thus, the second is expected to perform better in the near future and should have a higher reputation.

As an example, consider two agents where the first agent made 98 correct decisions followed by 2 wrong decisions recently. The second agent made 2 wrong decisions followed by 98 correct decisions. Hence, the second one had better recent performance and is expected to perform well. Intuitively, it should have a higher reputation.

Examples of reputation functions that provide this property is exponential moving average. We will start with exponentially moving average and modify it in the next section.

E. Configurable Increase and Decrease

The increase in reputation should be configurable to suit various application domains. Some applications, e.g., critical decisions, require a slow increase in the reputations so that new agents are not trusted easily. Other applications, e.g., recommendation systems, require moderate or fast growth. Similarly, a decrease in reputation should be configurable to suit different applications. Critical decision-making applications require a fast decrease to penalize misbehaving agents faster.

Further, it should be possible to have different increase and decrease factors to account for applications that require different configurations. For example, critical applications require a slow increase with a rapid decrease.

We use the reputation framework for malicious node detection, which requires a slow increase and a fast decrease. Nevertheless, it should be noted that configurable parameters are always desirable of any reputation management framework.

F. Slower Reputation Recovery

The increase in the reputation value after a wrong decision should be slower than a normal increase. That is if two agents with a reputation of 0.5 where agent A is just starting and agent B had made several good and bad decisions. Agent B should have a slower increase than the agent A, which is just starting.

The stipulation given above is required to penalize malicious nodes and prevent their fast reputation recovery. It also helps in preventing attack strategies where a malicious agent gives wrong decisions randomly or using a specific pattern.

G. Proportional Update

When a node makes a wrong decision, its reputation decrease should be proportional to how good it has performed so far. To explain this, consider two nodes, Agent A and Agent B, who have each made a wrong decision at the same time. Agent A previously made 1000 good decisions while Agent B made only 100. This means that Agent A made 1 in 1000 mistakes while Agent B made 1 in 100. It is intuitive that Agent A should have a higher reputation than agent B. The same thing applies to the increase in reputation recovery. The proportional update helps to provide fairness and a higher reputation for the node that has been in the system longer.

IV. Proposed Reputation Management Framework

The seven requirements discussed in the previous section should be reflected in any reputation formula used. In this section, we propose a reputation formulation and prove that it meets these requirements. We propose this formulation for a reputation framework to be used for probabilistic blockchain as will be discussed later.

A. Reputation Formulation

We start with a function that follows an Exponentially Weighted Average (EWA) to calculate agents' reputations. Thus, $R_{i,t}$, the reputation of Agent i at time t, may be calculated as follows:

$$R_{i,t} = \begin{cases} 0.5 & t = 0 \\ \alpha X + (1-\alpha) R_{i,t-1} & t \neq 0 \end{cases} \quad (1)$$

Where α is a configuration parameter that is between 0 and 1. X is a value that is generally either -1 or 1, deciding whether to decrease or increase the new reputation. X can also be a fractional value as we will see in our reputation function. For better notation, we will use R_t to represent $R_{i,t}$.

The parameter α affects the weight given to the agent's old performance. As α decreases, more weight is given to the prior performance, thus, the change in R_t value will decrease. This indicates that for a slow increase or decrease, α should be small.

The EWA formulation above is not sufficient to meet all seven previously discussed requirements. More specifically, EWA formulation does not provide boundedness (between 0 and 1), different configurable increase and decrease, and proportional decrease properties. In the following, we discuss how to improve the formulation.

a) ***Boundedness:*** The EWA formulation is [-1, 1] bounded while we require [0, 1] bound to represent a percentage. To satisfy our bounds, we need to change the output from [-1, 1] to [0, 1]. Thus, the new *percentage EWA formulation* can be represented as follows:

$$R_t = \frac{\begin{cases} 0 & t=0 \\ \alpha X + (1-\alpha)(2R_{t-1}-1) & t \neq 0 \end{cases}}{2} + 0.5 \quad (2)$$

Note that X is generally either -1 or 1, and $\alpha \leq 1$. It can be seen that with this translation, if R_{t-1} is in the interval [0, 1], R_t is also in the same interval [0, 1]. Values of R_t for three values of R_{t-1} are shown in the table below:

	R_t	
R_{t-1}	X=-1	X=+1
0	0	α
0.5	0.5(1-α)	0.5(1+α)
1	1-α	1

b) ***Configurable increase and decrease:*** The EWA formula has one configuration parameter α to account for both

the increase and decrease. However, as discussed earlier, some applications require different increase and decrease parameters. For example, loss due to a wrong decision may be much higher than gain due to the correct decision. To satisfy such cases, we consider α as an increase parameter and add another parameter β for the decrease. Thus, the new *multi-configurable percentage EWA formula* can be as follows:

$$R_t = \frac{\begin{cases} 0 & t=0 \\ \alpha X+(1-\alpha)(2R_{t-1}-1) & t>0, \text{correct decision} \\ \beta(-X)+(1-\beta)(2R_{t-1}-1) & t>0, \text{wrong decision} \end{cases}}{2} + 0.5 \quad (3)$$

*c) **Proportional update:*** As X generally has a value of either 1 or -1, i.e., decrease or increase, the first term in EWA will be the same for all agents. That is, if two agents have the same reputation before making a wrong decision, their reputation will be the same regardless of their prior performance. However, as discussed earlier, proportional decreases and increases provide higher fairness to agents who have participated longer in the system. To achieve this property, we set X to the proportion of good or bad behavior of the agent up to this point. Formally:

$$R_t = \frac{\begin{cases} 0 & p=0, n=0, t=0 \\ \alpha\frac{p}{p+n}+(1-\alpha)(2R_{t-1}-1) & t>0, \text{correct decision} \\ \beta\frac{-n}{p+n}+(1-\beta)(2R_{t-1}-1) & t>0, \text{wrong decision} \end{cases}}{2} + 0.5 \quad (4)$$

Where p is the number of correct decisions and n is the number of wrong decisions before time t.

We refer to the reputation formula in Eq. 4 as *Rated Proportional Multi-Configurable EWA (RPMC-EWA)* formula. We propose using RPMC-EWA in calculating the reputation for each agent as will be seen.

It should be noted that we started with EWA and moved towards meeting our requirements. The same thing can be done for other known starting formulas. We choose EWA as it meets most requirements and does not rely on any underlying distribution assumption.

B. RPMC-EWA Meets the Requirements

The discussions in the previous subsection prove boundedness, configurable increase and decrease, and proportional update properties of RPMC-EWA. Next, we show that RPMC-EWA meets other requirements discussed earlier.

RPMC-EWA is a continuous function, and any decision made will result in a slightly different reputation value. It is bounded between 0 and 1, as discussed earlier. It starts at 0.5 following the first condition in Eq. 4. It has a configurable increase/decrease since both α and β are configurable. It follows a proportional decrease as $\frac{p}{p+n}$ was introduced for this. Further, RPMC-EWA is time-dependent as it inherits EWA and will be shown in Lemma 1 and it provides a slower reputation recovery as will be shown in Lemma 2.

Lemma 1: *RPMC-EWA satisfies the time-dependent requirement.*

This property is inherited from the EWA formula [21]. To show this, we consider two agents, A and B. Agent A made a mistake at time t while Agent B made a mistake at $t-1$. Earlier decisions were the same. Thus, the reputation at $t-2$ is the same for both agents. If *RPMC-EWA* is time-dependent, Agent A should have a lower reputation that Agent B.

$$R_{A,t} = \beta\frac{-n}{p+n}+(1-\beta)\,R_{A,t-1} = \beta\frac{-n}{p+n}+(1-\beta)\,(\alpha\frac{p}{p+n}+(1-\alpha)R_{A,t-2})$$

$$R_{B,t} = \alpha\frac{p}{p+n}+(1-\alpha)\,R_{B,t-1} = \alpha\frac{p}{p+n}-(1-\alpha)\,(\beta\frac{-n}{p+n}+(1-\beta)R_{B,t-2})$$

$\frac{-n}{p+n}$ is almost equal to 0, and we denote it by $-\delta$. Similarly, $\frac{p}{p+n}$ is almost equal to 1, and we denote by Δ. Therefore,

$$\begin{aligned} R_{A,t} &= \beta(-\delta)+(1-\beta)\,R_{A,t-1} = \beta(-\delta)+(1-\beta)\,(\alpha(\Delta)+(1-\alpha)R_{A,t-2}) \\ &= -\beta\delta+\alpha\Delta+(1-\alpha)R_{A,t-2}-\beta\alpha\Delta-\beta(1-\alpha)R_{A,t-2} \\ &= -\beta\delta+\alpha\Delta-\alpha\beta\Delta+[(1-\alpha)-\beta(1-\alpha)]R_{A,t-2} \\ &= -\beta\delta+\alpha\Delta-\alpha\beta\Delta+[(1-\beta)-\alpha(1-\beta)]R_{A,t-2} \end{aligned}$$

$$\begin{aligned} R_{B,t} &= \alpha(\Delta)+(1-\alpha)\,R_{B,t-1} = \alpha(\Delta)+(1-\alpha)\,(\beta(-\delta)+(1-\beta)R_{B,t-2}) \\ &= \alpha\Delta-\beta\delta+(1-\beta)R_{B,t-2}+\alpha\beta\delta-\alpha(1-\beta)R_{B,t-2} \\ &= \alpha\Delta-\beta\delta+\alpha\beta\delta+[(1-\beta)-\alpha(1-\beta)]R_{B,t-2} \end{aligned}$$

The last term is the same for both agents and, therefore:

$$R_{A,t} - R_{B,t} = \alpha\beta(\Delta - \delta)$$

Hence, the reputation of Agent A is higher than the reputation of Agent B. Thus, the agent who made the most recent mistake will have the lowest reputation.

Lemma 2: *RPMC-EWA meets the slower reputation recovery property*

To prove this, we look at the value of the term $\frac{p}{p+n}$ in Eq. 4. For an agent that has just started, and has made a correct decision, n will be 0. Thus, the above term will be 1, which is the maximum increase achieved. For an old agent, who has committed mistakes earlier, $n \neq 0$. Thus, the term will be less than 1. Obviously, the second case will have a slower increase. Further, as n increases, the recovery will be even slower.

Fig. 2 illustrates this with a case scenario where Agent A has just started, Agent B made 10% wrong decisions earlier, Agent C made 15% wrong decisions earlier, and Agent D made 25% wrong decisions earlier. All agents are making good decisions since the simulation started. Note that $\alpha=0.005$ *and* $\beta=0.3$.

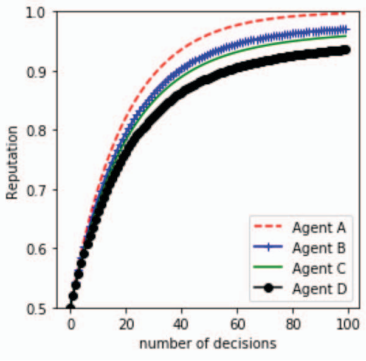

Fig. 2: *RPMC-EWA slower reputation property*

V. RPMC-EWA AND PROBABILISTIC BLOCKCHAINS

This section discusses RPMC-EWA can be used in probabilistic blockchains. Further, it highlights RPMC-EWA uses for the probabilistic blockchain paradigm.

A. RPMC-EWA in Probabilistic Blockchains

The addition for probabilistic blockchain to deploy RPMC-EWA is in the consensus calculation and validation process. For the calculation, the miner gets the last reputations for all participating agents and use it in calculating the new consensus in the block. For validation, the validators also get the reputations for all agents and recalculate the proposed consensus to ensure its correctness. This guarantees that the miner is not controlling the consensus nor manipulating the reputations.

After the block is committed, all nodes including update agents' reputations for future blocks. The number of good and bad behavior for all agents have to be stored at all nodes in order to calculate and update the reputation. However, it should be noted that these reputations are not stored in the chain and kept off-chain for block size limitation purposes.

B. RPMC-EWA Use for Malicious Node Detection

The first application of the reputation framework discussed in the previous section is the malicious node detection. This application is the focus of performance analysis in this paper. In such an application, agents with low reputation are considered malicious. A starting node has a reputation of 0.5. Therefore, we consider an agent with a reputation lower than 0.5 as malicious or a poorly performing agent. In some applications, such agents may be excluded from the decision process.

For this application, it is preferable to increase agents' reputations gradually as they make good decisions and decrease it rapidly as they make bad decisions. The slow increase is required, so that newly joined agents are less favorable than agents with a long history of good behavior.

It should be noted that nodes that commit only a few mistakes may also be detected as malicious. This is true in decision-making applications as these agents affect the consensus. Also, if bad agents keep enhancing their performance, their reputation will rise above 0.5, eventually. Hence, they will again be considered as normal agents.

Before calculating the consensus, miners should calculate the reputation for agents and exclude malicious agents from the decision process. However, if their transactions are valid, the transaction will be added to the block. This acts as a record and allows all blockchain nodes to see other agents' performances. In addition, having these records helps prove these nodes' reputation recovery in case they enhance their performance.

C. RPMC-EWA Use for Consensus Calculation

In [15], a simple average was used to calculate the consensus. Having introduced the concept of reputation in this paper, a weighted average approach can be used with weights reflecting the agent's reputation. It allows good agents to contribute more to the consensus. Agents with higher weights have a higher probability of performing well in the future. In contrast, agents that had poor performance recently but not yet detected as malicious should contribute less to the consensus.

Note that, the increase and decrease in agent's reputations for this application is not restricted. However, as we are dealing with decision-making applications, the slow increase, and the rapid decrease is still recommended.

The reputations of agents can be used to calculate the weighted average. Miners consider all participating agents' reputations when calculating the consensus. These reputations should be normalized so that their sum is equal to 1. Then, the weighted average can be used to calculate the consensus.

VI. Performance Evaluation

In this section, we evaluate the performance of RPMC-EWA for probabilistic blockchain malicious node detection. We start by discussing a baseline reputation algorithm that will be used for comparison and the attack strategies to be used for evaluation. Then, we present two evaluation settings along with their results to show the performance of RPMC-EWA.

A. Beta Reputation Algorithm

The beta reputation algorithm is one of the traditional and most widely adopted reputation formulation that is used in reputation systems. It is derived from Posteriori probabilities of binary events that can be represented by a beta distribution. The expected value of the beta distribution is given $\alpha/(\alpha+\beta)$ where α and β are shape parameters that are greater than 1 [22].

Given that $\alpha=p+1$ (i.e., correct decisions) and $\beta=n+1$ (i.e., wrong decisions), the resulting expected value for the reputation follows the expected value of the beta distribution. That is, an agent's reputation at time t can be calculated as follows:

$$R_t = \frac{p+1}{p+n+2}$$

B. RPMC-EWA Algorithm Setting

For the malicious node detection, we need a slow increase and rapid decrease setting. To be consistent with this requirement RPMC-EWA will use $\alpha=0.005$ and $\beta=0.3$, for the rest of the paper, unless otherwise stated.

C. Attack Strategies

We assume that a normal agent is performing perfectly and making continuous correct decisions. If a node makes wrong decisions, it contributes to the decrease of its reputation. Thus, the assumption made above considers the worst-case scenario, where the attacker's target is the perfectly behaving node. It is a worst-case scenario because the agent's reputation is maximized as it has not made any mistake yet.

We followed four strategies for an attacker to use. These are discussed as follows:

Continuous-flipping strategy*:* This is the most straightforward strategy where a malicious node continuously flips its decision. This strategy is easy to detect since the number of wrong decisions will be growing steadily without changes to the number of correct decisions. Thus, it is rarely followed in real life, but we discuss it as a base case scenario.

Pattern-flipping-1 strategy : In this strategy, the malicious node will flip its decisions based on a predefined pattern. This results in an eventual increase in both correct and wrong decisions. Although it is harder to detect than the first strategy, it can also be detected by analyzing the agent history. For this case, a malicious agent is flipping the decision every four decisions. That is, the malicious agent makes three correct decisions then a wrong decision.

Pattern-flipping-2 strategy: In this strategy, the agent make 10 correct decisions and one wrong.

Random-flipping strategy: In this strategy, a malicious node will randomly flip its decisions resulting in an eventual increase in both correct and wrong decisions. It is harder to detect than others and thus, followed mostly in real attack scenarios.

D. RPMC-EWA and Beta Algorithms Performance

We compare RPMC-EWA algorithm with the beta reputation algorithm as a baseline reputation algorithm. An agent is detected as malicious if its reputation falls below 0.5, i.e., its starting reputation. We vary the number of correct decisions made before the agent turned malicious and evaluate whether the algorithm can detect the agent as malicious. The evaluation metrics used are the detection accuracy and the decision number at which the agent is detected as malicious. The latter is only measured when the algorithm can detect.

The results of the above experiments are shown in Table I. "Decision #" is the decision at which the agent turned malicious while "Detection decision #" is the decision at which the agent is detected as malicious. As can be seen, RPMC-EWA outperforms beta reputation in all cases. Beta reputation is able to detect a malicious agent only when in continuous flipping strategy and some cases of random flipping strategy. On the other hand, RPMC-EWA performs well in all strategies except for pattern-flipping-2 strategy.

Table I. Results of RPMC-EWA and Beta algorithms performance for malicious node detection

Setting		Beta reputation		RPMC-EWA reputation	
Strategy	**Decision #**	**Accuracy %**	**Detection decision #**	**Accuracy %**	**Detection decision #**
Continuous flipping	1	100	3	100	2
	50	100	99	100	54
	100	100	199	100	106
	500	100	999	100	511
	1000	100	1999	100	1011
Random flipping	1	100	20 *	100	2 *
	50	10	92 *	100	59 *
	100	0	**	100	113 *
	500	0	**	100	525 *
	1000	0	**	100	1030 *
Pattern flipping 1	1	0	**	100	2
	50	0	**	100	72
	100	0	**	100	167
	500	0	**	100	586
	1000	0	**	100	1165
Pattern flipping 2	***	0	**	0	**

* an average value
**Cannot be calculated
*** For all numbers

Beta's bad performance is due to a small decrease in reputation when an agent makes any mistake. That is, an agent reputation will be going up and down by a small amount; thus, never reaches 0.5 except if the agent continuously flips or makes wrong decisions. In contrast, RPMC-EWA with rapid or even moderate decrease and slow increase can achieve 0.5 reputation faster as the decrease is much higher than increase.

E. Malicious Node Detection and the Consensus

We also evaluate the effect of malicious node detection on the consensus in probabilistic blockchains. As beta algorithm did not perform well on malicious node detection, we excluded it in this experiment. We compare the consensus with and without malicious node detection.

The work in [14] has shown that probabilistic blockchain becomes resilient against malicious nodes as soon as these nodes are less than half of the total number of agents. However, this resiliency applies to the consensus interpretation rather than the consensus probability of an event. For example, the probability in the existence of a malicious node is 80% while honest agents have provided an individual probability of 100%. With malicious node detection, it is possible to get the exact consensus probability as decisions made by malicious nodes are excluded.

As with the first experiment above, we use the same four attack strategies. Ten agents are participating with one malicious. Honest nodes give the same decisions with a probability of 1 while a malicious node flips the decision. Different probabilities may be used; however, we took this case in the context of this paper. To harden the detection, an agent is turned malicious only for a limited time (Decision 300 to Decision 500). We evaluate the consensus with and without malicious node detection as demonstrated in Fig. 3.

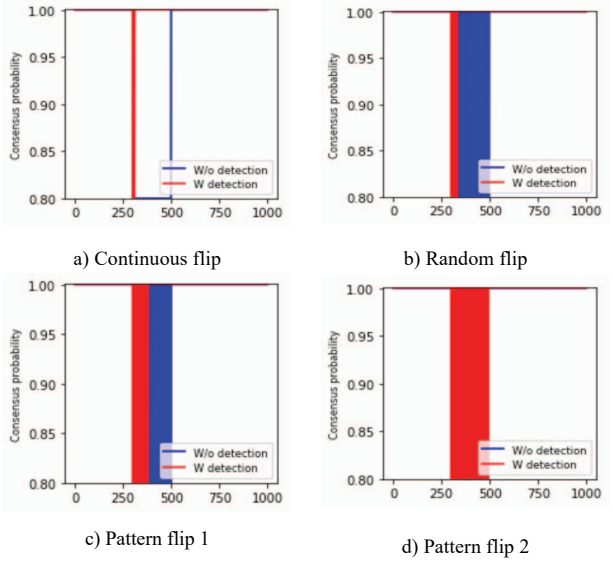

Fig. 3: Consensus Probability with and without malicious node detection

As can be seen, the malicious node was detected and excluded from the consensus except for pattern-flipping-2. Considering a higher β facilitates the detection in the last case. In the case of continuous flip strategy, the node was detected right after it turned malicious. Whereas, in case of the random flip pattern, the malicious node was detected after it had made around 25 decisions after turning malicious. For pattern flip 1, the malicious node is detected after approximately 50 decisions.

F. Analysis Limitation and Further works

The analyses made in this paper have several known limitations to be handed in future works. First, *all analyses are*

done off-chain as the reputation framework are yet to be integrated with probabilistic blockchain. Since the changes are only in mining, it is expected that these results will hold for blockchain. However, *the reputation framework integration with a blockchain platform* is needed to validate the results. The challenge is in tracking the reputations and making sure that there is a consensus among nodes.

In addition, *the framework computation analysis and overhead* are still to be evaluated. The *effects of consensus and reputation calculation on throughput and delay* need to be investigated. These effects are critical to ensure that the proposed system does not add substantial overhead to the blockchains. Further, *comparing the proposed reputation framework to already existing blockchain-based reputation* solutions is also desirable for the proposed approach.

The *security of the proposed approach,* along with probabilistic blockchains needs to be further analyzed theoretically and practically. Finally, *applying the proposed approach to other applications and attack scenarios* can add to the applicability of the proposed approach for decision making.

VII. Conclusion

Knowledge-based blockchains are gaining significant attention in building blockchain-based AI applications. In our earlier work, probabilistic blockchains, a blockchain paradigm to make consensus decisions within the blockchains, is proposed. The paradigm has significant potentials in building efficient AI systems in FinTech and non-FinTech applications. In this paper, we extend probabilistic blockchains and propose a reputation management framework for agents' reputation in probabilistic blockchains. The framework is applied to malicious node detection where malicious agents are excluded from the blockchain consensus. Results showed that the proposed approach outperforms the traditional beta reputation system in detecting malicious nodes. Further, detecting and excluding malicious nodes show a better performance in consensus evaluation by probabilistic blockchains. It has been shown that the proposed reputation framework can also be used for consensus calculation, where the agents' contributions to the consensus depend on their prior performances. Both applications are to be investigated in future extensions of the proposed work.

Acknowledgment

This publication was made possible by NPRP grant #NPRP11S-0109-180242 from the Qatar National Research Fund (a member of Qatar Foundation) and the National Science Foundation Grants No. CNS-1547380 and CNS-1718929.The findings achieved herein are solely the responsibility of the author[s].

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Accessed 3 June 2019].

[2] M. Pilkington, Blockchain technology: principles and applications, Research handbook on digital transformations, 2016.

[3] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), Munich, 2016.

[4] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292-2303, 2016.

[5] B. Betts, "Blockchain and the promise of cooperative cloud storage," Computerweekly, August 2016. [Online]. Available: http://www.computerweekly.com/feature/Blockchain-and-the-promise-of-cooperative-cloud-storage. [Accessed February 2019].

[6] P. McCorry, C. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in International Conference on Financial Cryptography and Data Security, Malta, 2017, pp. 357-375.

[7] L. Shen, "Ethereum-Based Blockchain Betting Platform Augur Just Launched. Here's Why It's Not Married to Ether," 9 July 2018. [Online]. Available: http://fortune.com/2018/07/09/augur-coin-crypto-rep-vitalik-buterin/. [Accessed 3 June 2019].

[8] M. Swan, "Blockchain Thinking : The Brain as a Decentralized Autonomous Corporation [Commentary]," IEEE Technology and Society Magazine, vol. 34, no. 4, pp. 41-52, Dec 2015.

[9] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang and J. Han, "When Intrusion Detection Meets Blockchain Technology: A Review," IEEE Access, vol. 6, pp. 10179-10188, 2018.

[10] T. Golomb, Y. Mirsky and Y. Elovici, "CIoTA: Collaborative IoT Anomaly Detection via Blockchain," in Network and Distributed Systems Security Symposium (NDSS), San Diego, CA, 2018.

[11] R. Jain, "Extending Blockchains for Extending Blockchains for Risk Management Risk Management and Decision Making and Decision Making," 9 November 2018. [Online]. Available: https://www.cse.wustl.edu/~jain/talks/ftp/pbc_ibf.pdf. [Accessed 3 June 2019].

[12] T. Salman, R. Jain and L. Jupta, "probabilistic Blockchains: A Blockchain Paradigm for Collaborative Decision-Making," in The 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, New York, November 2018.

[13] K. Yamada and H. Saito, "What's So Different about Blockchain? — Blockchain is a Probabilistic State Machine," in IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW), Nara, 2016, pp. 168-175.

[14] F. Gai, B. Wang, W. Deng and W. Peng, "Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network," in Database Systems for Advanced Applications, Gold Coast, Australia, 2018.

[15] Github, "The official GoChain client. https://gochain.io," gochain, [Online]. Available: https://github.com/gochain-io/gochain. [Accessed 3 June 2019].

[16] D. Qin, C. Wang and Y. Jiang, "RPchain: A Blockchain-Based Academic Social Networking Service for Credible Reputation Building," in International Conference on Blockchain, Seattle, WA, USA, 2018.

[17] J. Yu, D. Kozhaya, J. Decouchant and P. E. Verissimo, "RepuCoin: Your Reputation is Your Power," IACR Cryptology ePrint Archive 2018, 2018.

[18] C. Huang, Z. Wang, H. Chen, Q. Hu, Q. Zhang, W. Wang and X. Guan, "RepChain: A Reputation based Secure, Fast and High Incentive Blockchain System via Sharding," arXiv preprint arXiv:1901.05741.

[19] C. Tang, L. Wu, G. Wen and Z. Zheng, "Incentivizing Honest Mining in Blockchain Networks: A Reputation Approach," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. (to appear), 2019.

[20] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim and J. Zhao, "Toward Secure Blockchain-Enabled Internet of Vehicles: Optimizing Consensus Management Using Reputation and Contract Theory," IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2906-2920, March 2019.

[21] Weki, "Exponential_moving_average," Wekipedia, [Online]. Available: https://en.wikipedia.org/wiki/Moving_average#Exponential_moving_average. [Accessed 22 Febreuary 2019].

[22] A. Josang and R. Ismail, "The Beta Reputation System," in 15th Bled Electronic Commerce Conference, Bled, Slovenia, 2002.