Received 22 November 2022, accepted 18 January 2023, date of publication 23 January 2023, date of current version 31 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3239394

RESEARCH ARTICLE

Generalized Secure and Dynamic Decentralized Reputation System With a Dishonest Majority

KHALID MRABET[1], FAISSAL EL BOUANANI[1], (Senior Member, IEEE), AND HUSSAIN BEN-AZZA[2]
[1]ENSIAS College of Engineering, Mohammed V University in Rabat, Rabat 10000, Morocco
[2]ENSAM National High School of Arts and Trades, Moulay Ismail University at Meknes, Meknes 50500, Morocco

Corresponding author: Khalid Mrabet (khalid_mrabet@um5.ac.ma)

ABSTRACT Reputation management systems are essential for establishing trust among network users. They are tools for reinforcing cooperation and sanctioning malicious behavior. This importance becomes a requirement in decentralized environments such as mobile ad-hoc networks (MANETs), peer-to-peer systems (P2P), wireless sensor networks (WSNs), or decentralized social networks (DSNs) where there is no trusted third party to monitor and enforce good behavior among users. In this paper, we propose a dynamic decentralized reputation system that fits such network characteristics, namely decentralization, dynamism, and openness, without conceding on security. The novel system is a general-purpose system that uses blockchain to gather and supply global reputation information while remaining fully decentralized. Unlike previous works on decentralized reputation systems where reputation information is inconsistent and limited to users' direct experience and recommendations from peers (neighbors), our system gathers feedback from all over the network and stores reputation information on a distributed ledger fully accessible to all users. In terms of security, the proposed method achieves privacy utilizing secure multiparty computation, a cryptographic primitive that preserves feedback privacy even with a dishonest majority reaching $n-2$ malicious parties while requiring only $O(n)$ messages. The employed techniques enable the system to achieve unique characteristics like consistency, conservation, and verifiability in addition to privacy. The security analysis we provide confirms these properties, and the performed simulation shows the protocol's effectiveness.

INDEX TERMS Blockchain, privacy, reputation, secure multiparty computation.

I. INTRODUCTION

In a world increasingly interconnected, with sustained progress of services and people interacting online, the need for reputation systems is gaining momentum. Indeed, reputation systems help establish trust between users in situations where most online parties are strangers. They enable them to predict who is likely trustworthy even without knowing them. Based on feedback from past transactions, a reputation system typically collects, aggregates, and distributes data about a party, to characterize and predict its future behavior. Consequently, it facilitates community interaction and improves online services efficiency [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

Measuring a user's reputation amounts to estimating the community's confidence in him. Depending on the context, he is expected to behave in a specific manner or perform particular actions. In return, other users give their appreciation of the quality of those actions. If aggregated properly, those appreciations will provide a global score at the community level.

In e-commerce, for example, there are many marketplaces equipped with reputation systems. They collect feedback on products and sellers to help customers decide which product to purchase and from whom. They are good examples of centralized reputation systems. Such systems are possible because of the trust both vendors and consumers put in the marketplace operator.

In many contexts, however, such a trusted party does not exist. Examples include MANETs, Vehicular Ad hoc

Networks (VANETs), DSNs, P2P systems, and WSNs; This is the reason why decentralized reputation systems (DRS) like [2], [3], and [4] exist.

In MANETs, for example, a party uses his neighbors to route messages. It can collect reputation information on them by observing whether they forward the messages sent through them or not. However, when a new party joins the network, it has no reputation information on others. So in traditional decentralized reputation systems, the new party can ask its peers for their reputation information on a target party and use this as the basis to decide whether or not to interact with it.

One of the major concerns about this approach is feedback privacy. If feedback from some users is exposed eventually when their identities are public, they might be subject to retaliation or attacks and may receive low ratings as a reprisal.

To remedy this situation, several privacy-preserving DRS (PDRS) have been proposed by researchers [5], [6], [7], [8], [9]. In such systems, parties avoid providing their reputation information directly to the requesting party. Instead, they use a protocol that allows them to jointly compute a function based on their rating values (typically the sum or the average). They then send the outcome to the party who made the request. This way, they can keep their ratings private while getting desired results.

However, privacy is not the only concern with traditional DRS. Other problems negatively impact the core functions of these systems, making them ineffective.

1) *Availibility:* when the reputation information is not evenly available over the network, and thus the system is not fair to all users;
2) *Consistency:* The reputation information is not the same everywhere and thus inconsistent over the network since users are likely to get different reputation scores for the same target user at the same time based on their relationships and acquaintances;
3) *Conservation:* The reputation information is likely to be lost with users quitting the network since they store their ratings locally;
4) *Efficiency:* Another inherent drawback is efficiency, which relates to the fact that the reputation score has to be recalculated from scratch every time a user queries for it, which comes with a consequent cost in computation power and communication.

Motivated by these issues, the present work proposes using blockchain as a distributed database for reputation storage and update, jointly with secure multiparty computation for privacy. Unlike the previous works on the subject, the proposed system achieves some important and unique properties. Namely, *Availibility*, *Consistency*, *Conservation*, and *Verifiability* in addition to *Correctness* and *Privacy*.

Some of these properties can seem straightforward if looked at in the context of centralized reputation systems. Nevertheless, in the present context of dynamic decentralized reputation systems, they are challenging. Above that, our protocol is a fully decentralized general-purpose global reputation system, which to our best knowledge, was not achieved before in related works.

The proposed system is distinguished from previous ones by its non-reliance on a querying party to initiate reputation computation since source parties in our system calculate and submit their feedback directly to the blockchain rather than providing it on demand. This solution ensures that the effort is not repeated at each request and saves to a significant extent communication bandwidth and computation power.

This paper's contributions can be summarized as follows (see also Fig. 1):

- We propose an efficient and dynamic decentralized reputation system by combining secure multiparty computation [10] and non-interactive zero-knowledge proof [11] with blockchain technology [12]. Individual rating statistics are kept private, and only the aggregated reputation is made public under the proposed system. Furthermore, the blockchain architecture ensures that reputation data is consistent and available throughout the network.
- We design a blockchain-based architecture that implements the proposed reputation system to guarantee system transparency and verifiability. The proposed architecture reduces the on-chain storage and computation overhead with an off-chain phase. Security analysis demonstrates the reliability of the proposed system;
- After exploring different models used to represent and aggregate reputation in standard scenarios from the literature, we achieve a general-purpose reputation system that complies with the existing models. A system that is not limited, for example, to buyers/sellers or providers/clients settings where nodes are either raters or ratees. Instead, each party in the network has to act simultaneously as a rater and a ratee.

The rest of the paper is organized as follows. In Section II, we present some related works. Next, the problem and security definitions are provided in Section III. Section IV introduces the essential building blocks, while a review of the reputation representation and aggregation models in standard scenarios is presented in Section V. The sixth section details the proposed protocol phases, whereas the security analysis and performance tests, insights, and discussions, are provided in Section VII. Lastly, we conclude the work in Section VIII.

II. RELATED WORKS

There are two approaches to privacy-preserving decentralized reputation systems: The first approach concentrates on protecting user anonymity without guarding feedback. Whereas the second focuses on maintaining feedback confidentiality without necessarily hiding identities. Concretely, the distinction between the two is as follows:

- *User anonymity systems* assign to each user one or more pseudonyms that cannot be linked to his true identity so that feedback providers are kept anonymous by having their identities hidden in the system. These systems allow users to issue transactions and provide feedback

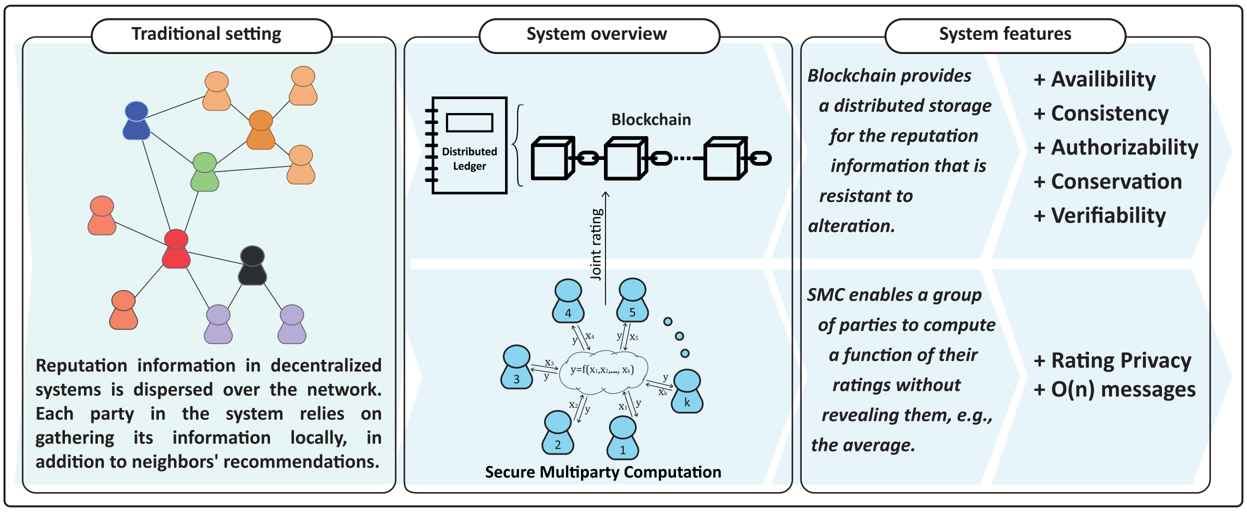
Traditional setting
Reputation information in decentralized systems is dispersed over the network. Each party in the system relies on gathering its information locally, in addition to neighbors' recommendations.
System overview
Distributed Ledger
Blockchain
Joint rating
y=f(x₁,x₂,..., xₖ)
Secure Multiparty Computation
System features
Blockchain provides a distributed storage for the reputation information that is resistant to alteration.
+ Availibility
+ Consistency
+ Authoritability
+ Conservation
+ Verifiability
SMC enables a group of parties to compute a function of their ratings without revealing them, e.g., the average.
+ Rating Privacy
+ O(n) messages

FIGURE 1. System contributions.

without concealing it, as it cannot be linked to their identities.

- *Feedback confidentiality systems*, on the other hand, assign to each user a single pseudonym and keep feedback confidential. They do not conceal who is providing feedback; Instead, they hide the feedback value. These systems, in theory, should not reveal information other than the aggregated reputation score.

The present work follows the second approach as it considers it more realistic and practical because complete anonymity is not always possible to achieve in real-world scenarios. For example, on e-commerce sites, even if we can preserve anonymity online, the exchange of physical items sold on them would likely reveal identities. From this perspective, feedback-confidentiality systems are a better alternative to enable users to submit truthful feedback without fear of retaliation.

The major part of traditional works in PDRS (e.g. [5], [7], [8], [9]) consider the setting where a querying party P_q wants to interact with a target party P_t, but is not sure that P_t is trustworthy. Either P_q lacks information about P_t past behavior, or its experience with P_t is too limited or outdated. Let $\{P_1^{(t)}, P_2^{(t)}, \ldots, P_N^{(t)}\}$ be the set of parties having reputation information on P_t called witnesses or source parties. In this case, P_q can consult a selected subgroup of source parties, namely $\{P_{i_1}^{(t)}, P_{i_2}^{(t)}, \ldots, P_{i_n}^{(t)}\}$ $(n \leq N)$ that will run a protocol to compute the reputation score of P_t securely and send the result to P_q.

Following this setting, authors in [5] presented one of the earliest works in the field, with a protocol based on random witness selection and additive secret sharing. The protocol came with three different security strengths and proved the existence of witness selection schemes that guarantee at least two honest witnesses with a high probability. Although the protocol is fully decentralized and designed for general use, it cannot compute and store global reputation scores. Indeed, each party in the system has to store its gathered information locally, and reputation is based only on neighbors' feedback. In addition, the system requires exchanging $O(n^2)$ messages at each request for reputation.

Authors in [6] and [8] built upon the work of [5] with the k-shares reputation protocol, respectively for the semi-honest and malicious adversarial model. Their protocol improves communication costs by requiring only $O(n)$ messages. In addition, it maximizes the probability of keeping reputation information private by allowing users to select only witnesses with good reputation scores avoiding those they do not trust.

The prior protocols [5], [6], [8] are not suitable for dynamic networks, and several issues arise when attempting to use them. In such networks, the number of available source parties, i.e., currently part of the network, could be much smaller than if the network was entirely static. Indeed, once a party leaves, all its reputation information is no longer accessible since each party stores its information locally. Also, reputation is likely to be computed with a different set of present parties each time a party requests it. This leads reputation information to get inconsistent and changes at each request.

To alleviate the issue in dynamic networks where parties can enter and quit the network at will, authors in [9] proposed a protocol that allows parties leaving the network to delegate their reputation information to prevent its loss. For privacy reasons, the leaving user has to split the entrusted information over a group of users (secret-sharing) before leaving. Of course, this operation does not go without an increase in computation and communication costs, and if a delegation group member leaves, its information has to be re-delegated. Naturally, recovering and reconstructing the delegated information becomes more challenging with the number of parties involved in inflating and the data becoming fragmented.

However, even if the issue of reputation information loss was partially resolved, it is clear that all the proposed solutions above are incapable of computing and storing global reputation scores.

We can say the same about the work in [13] proposing a decentralized reputation model for VANETs that aims to enable nodes to detect malicious vehicles and avoid interacting with them based on their trust scores. It uses a different model based on the Bayesian filter to measure trust scores accurately, although the reputation information gathered is partial and mostly shared locally.

To structure this section better, we assess and classify related works by the following criteria:

1) Full Decentralization: where reputation systems do not use central entities to collect, compute or serve reputation scores; instead, the information is distributed over parties that share it to evaluate the trustworthiness of potential transaction partners.
2) General-Purpose Reputation Systems are those designed for usage in various network environments. They are not restricted to settings like service providers/consumers in online marketplaces or servers/clients in IoT. Instead, they must fit various networks, including P2P, MANETs, or WSN.
3) Global Reputation Systems are systems that collect ratings from all over the network and aggregate them into global reputation scores accessible to all users.

It is worth mentioning that many proposed systems in the literature are not general-purpose systems like the ones mentioned above. They are rather tailored for specific contexts such as online marketplaces [14], [15], [16], or internet of things (IoT) [17], [18], [19], [20] where the network is split into two distinct groups: ratees and raters. Indeed, users in online marketplaces are either consumers, thus raters, or service providers, thus ratees, while in IoT, they are either server nodes or clients. However, in different contexts such as P2P, MANETs, VANETs, DSNs, and WSNs, users have to play both the role of a rater and a ratee.

We highlight that such proposed systems are too specific or incompatible with a fully distributed setting like P2P, MANETs, or WSN.

Among them, we find:

PrivBox [21], a verifiable reputation system for online marketplaces. A system designed so consumers can rate retailers and submit their feedback in an encrypted form using homomorphic encryption to a public bulletin board (PBB): a sort of central database. The system makes the reputation information publicly accessible and verifiable without disclosing individual ratings due to the possibility of computing on ciphertexts with homomorphic encryption. However, the protocol leaves the reputation computation task to anyone (seller or buyer) who wants to compute the reputation of a particular retailer. The system employs zero-knowledge proofs to prove that ratings are well-formed.

Another system from [22] called PrivRep and built on [21] uses a similar PBB in addition to a Reputation Engine (RE), responsible for computing reputation from the homomorphically encrypted feedback. The RE is an entity controlled and run by the marketplace that supports the task instead of relegating it to regular users. It reserves the right to reject some of the feedback if judged untrustworthy. PrivRep assumes two central entities: RE for computation and PBB for storage, which undermines its decentralized aspect.

Same for [23] where authors proposed a similar system for the Social Internet of things with a PBB mentioning the possibility of implementing it as a blockchain or as a mirrored server.

In [15], authors proposed a blockchain-based cross-platform reputation system for e-commerce named RepChain; the system interconnects e-commerce platforms and enables them to share their users' reputations through a consortium blockchain. Even though the system is not entirely decentralized, as each platform relies on its centralized entity, the top layer interconnecting platforms is decentralized thanks to blockchain.

Authors in [19] propose a solution to Blockchain usage limitations in the Internet of Things (IoT) reputation systems, especially their lack of scalability. They introduce a distributed ledger combining Tangle and Blockchain as a reputation framework. Combining Tangle with Blockchain is destined to provide maintainability of the former and scalability of the latter. Consequently, the proposed ledger could handle a more significant number of IoT devices and transactions.

In the same direction, authors in [24] proposed a two-layered blockchain-based reputation system for VANETs consisting of a local one-day message blockchain and a global vehicle reputation blockchain. The proposed model efficiently manages local traffic information through the local one-day blockchain, reducing the memory overhead of vehicles. According to the vehicle's actions, its reputation score is updated and stored permanently in the global reputation blockchain.

Blockchain has had a wide range of applications due to its outstanding features like security and reliability, especially in distributed settings [20], [23], [24], [25]. Among other applications is Fog computing, where blockchain may achieve secure decentralized reputation systems and identity management [26].

Authors in [20] proposed a system for the Internet of Things (IoT) that considers geospatial information in reputation management as the trustworthiness of a device can be affected by many factors, including its geographical location. The proposed solution has a cloud-fog-edge architecture where the fog layer employs a blockchain to form a decentralized network among fog nodes allowing decentralized and transparent management. The location-based system part stores geographical information in Smart Contracts and enables reputation values to vary according to geographical location.

To our best knowledge, From literature assessment and classification of related works according to five criteria, namely: Full Decentralization, General-Purpose, Global Reputation, Privacy, and employed Technologies (The comparative study is summarized in Table 1), we can confidently say that part of the related works are fully decentralized reputation systems that are general-purpose but do not maintain global reputation scores and rely on locally stored information. Their reputation information is partial and inconsistent over the network because it is limited to users' direct experience and recommendations from neighbors and acquaintances. The other part of related works is proposed in specific settings that achieve global reputation and some form of decentralization but are not general-purpose systems. Our objective in this work is naturally to fill the gap in global reputation systems that are general-purpose and fully decentralized.

III. PROBLEM FORMULATION

In this section, we formulate a model for our system and present the security and adversarial model.

A. SYSTEM MODEL

In our system, there are two entities, namely users and the blockchain. Users have two roles to play:

- Regular User uniquely identified by its public key *pk* or its address *a* can interact with other peers and leave a rating score for whom he interacted. He maintains a lightweight copy of the ledger locally;
- Validator or Miner is a regular user with a higher reputation score that enables it to contribute to maintaining the public ledger with other miners based on a consensus protocol (see IV-A).

At a high level, the system works as follows. Parties join the network by generating a pair of keys associating the public key to a unique identifier like a Media Access Control address, an IP, or a hash. All communication between parties is done via secure channels. The latter are emulated by employing signature and encryption. Specifically, we use ECDSA signature [27], and ECIES encryption schemes [28].

Parties can perform actions specific to the network and interact with each other. At the end of the interaction between, for example, two parties A and B, B will get an authenticated piece of data from A attesting to the exchange. It can then rate A accordingly by joining its source parties list, computing a joint rating with other peers in the list, and submitting it to the blockchain. Finally, rating transactions for the same party aggregate as a reputation score on the blockchain.

B. ADVERSARIAL MODEL

In the *semi-honest* adversarial model, users always follow the protocol. They execute the protocol according to the specifications and do not deviate from them. However, they may passively attempt to learn the inputs of honest users by using intermediate information received during the protocol execution or any other information they can gain through legitimate means.

Under the *malicious* model, things are different. Users may deviate from the protocol and attempt actively to achieve their objectives through arbitrary behavior and extra-protocol activities following diverse strategies.

A dishonest user may act alone or in agreement with other malicious users to achieve his goals. When multiple users of the same type work together, we refer to them as collusion. We call an adversary any coalition of dishonest users.

The present system is secure in the semi-honest and malicious adversarial model. For the on-chain phase, i.e., executed on the blockchain, the system is safe from any adversary with computation power/stake below 51% of the network. For the off-chain phase, it is secure with a dishonest majority up to $n-2$.

C. SECURITY OBJECTIVES/REQUIREMENTS

The main security objectives of the proposed system are privacy and integrity. Privacy consists of preserving the confidentiality of feedback and any information related to it, while integrity aims to maintain the functionalities of the reputation system unaltered. It includes, for example, preventing a malicious user from manipulating the reputation aggregation function to forge an unmerited good reputation. In the following, we enumerate privacy and integrity sub-objectives in detail, as well as some properties essential to the system:

- *Privacy* This property guarantees feedback confidentiality protection so that it is neither disclosed explicitly nor derived from any public or intermediate information gained by the adversary during the interaction. Even more, the adversary is unable to learn any additional information beyond the final reputation score;
- *Correctness* As the reputation system relies on regular users and validators to perform some computation. Correctness property requires that the adversary is unable to mislead the system to erroneous results;
- *Authorizability*. The requirement is that only users who have had an interaction with the ratee can rate him. This property prevents users who had not transacted with a ratee from assigning him unjustified feedback for some motive, whether for tarnishing his reputation or for false promotion (attacks such as bad-mouthing and self-promotion);
- *Verifiability* The requirement that any user should be able to identify all published feedback linked to some identity and verify its basis on authentic transactions from real partners. Moreover, he could verify that the reputation score was computed correctly from the related feedback;
- *Availibility* guarantees that every party on the network can access all reputation scores at will. For example, malicious users can attempt to take down the system by different means and attack or prevent honest users from accessing it. The system has to grant all legitimate users equitable access to public reputation scores at any time;
- *Consistency* is the requirement that if different parties request a reputation score at the same time, they get the same reputation score everywhere in the network until new ratings come into the system if we take into account the propagation time;
- *Conservation* This property requires that even if a party leaves the network, its ratings have to remain part of the

TABLE 1. A Comparative study of privacy-preserving decentralized reputation systems (FD: Fully Decentralized; GP: General-Purpose; GR: Global Reputation).

Paper	Ref.	FD	GR	GP	Privacy	Technologies
Pavlov & al.	[5]	✓	✕	✓	✓	Verifiable secret sharing, Discrete log commitment
Hasan & al.	[8]	✓	x	✓	✓	Zero-knowledge proofs, Homomorphic encryption
Clark & al.	[9]	✓	x	✓	✓	Secure Multiparty Computation, Delegation
Najafi & al.	[13]	✓	x	✓	x	Bayesian filter
Debe & al.	[17]	x	✓	x	x	Blockchain, Smart Contracts
Liu & al.	[18]	x	✓	x	✓	Blockchain, Zero-Knowledge Proof, Signature
Azad & al.	[21]	x	✓	x	✓	Non-Interactive Zero-Knowledge Proof, Homomorphic Encryption, Public Bulletin Board
Bag & al.	[22]	x	✓	x	✓	Non-Interactive Zero-Knowledge Proof, Homomorphic Encryption, Public Bulletin Board, Reputation Engine
Azad & al.	[23]	x	✓	x	✓	Non-Interactive Zero-Knowledge Proof, Homomorphic Encryption, Public Bulletin Board
Li & al.	[15]	x	✓	x	✓	Consortium Blockchain for multiple e-commerce platforms
Lee & al.	[24]	✓	✓	x	✓	Two Layered Blockchain, Digital signature.
Weerapanpisit & al.	[20]	x	✓	x	x	Blockchain, Smart Contracts
Our Protocol.		✓	✓	✓	✓	Blockchain, Secure Multiparty Computation, Non-Interactive Zero-Knowledge Proof, Semi-Homomorphic Encryption

system and that the reputation score is a function of all the previously submitted ratings in the network.

IV. BUILDING BLOCKS

In the following, $\mathbb{F}$ denotes the finite field $\mathbb{F}_p$ with p a sufficiently large prime number.

A. BLOCKCHAIN

A blockchain is similar to a distributed public database. Every user can read from this database, but it takes more to write to it. For that, users have to reach a majority consensus over the network before the writing is accepted [29]. Any action that modifies this database is called a transaction. Transactions are typically gathered and recorded in the form of blocks and then broadcast among all users. Once the network reaches a consensus over a block and accepts it, it becomes difficult to alter it. Blockchain is reputed for recording transactions efficiently in a verifiable and permanent way. For this reason, it is considered an append-only database.

Blockchain can also be seen as an ordered list of blocks $(\mathrm{B})_{(0 \leq i < l)}$ (see Fig. 2) chronologically constructed from the first block $\mathrm{B}_{(0)}$ called genesis block to the last block $\mathrm{B}_{(l)}$ which is the current valid block. Other blocks will add to this list in regular periods. The result of all transactions in all blocks at a given moment constitutes the state of the blockchain.

Blockchain is resistant to modification by design. It is typically managed by a P2P network adhering to a communication protocol.

A *transaction* T is a single instruction used typically to transfer a sum of coins (virtual money) securely. Each *block* $\mathrm{B}_{(i)}$ is composed of the header $\mathrm{H}_{(i)}$ and the body that comprises a series of transactions $\mathrm{T}_{(i)} = (T_1, \ldots, T_n)$ in the form of a Merkle tree. The header $\mathrm{H}_{(i)}$ includes a collection of relevant pieces of information.

$$\mathrm{B}_{(i)} = (\mathrm{H}_{(i)}, (T_1, \ldots, T_n)).$$

As mentioned above, the *state* is the resultant of all transactions in all blocks at a specified time. It is an auxiliary

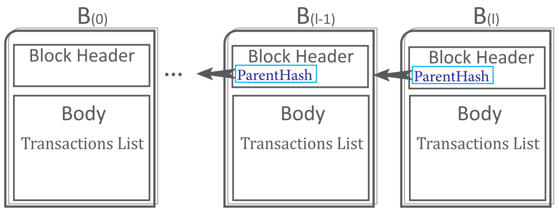

FIGURE 2. Blockchain structure.

database that summarizes the blockchain state at a given moment. The state database introduced with Ethereum [27], [30] is a modified Merkle tree implemented as a mapping between parties' addresses and their account states. It is maintained, updated, and linked to every new valid block (see Fig. 3). Unlike Ethereum, Bitcoin does not implement an equivalent structure [31].

We denote the state database σ and use a party address a to reference its account denoted by σ^a. In our context, the **account state** will include, among others, the following fields relevant to our system:

- Nonce: The number of transactions sent from this address so far, denoted $\sigma_n^{(a)}$.
- Reputation: The reputation score, denoted $\sigma_r^{(a)}$.
- Weight: The number of feedback received so far, denoted $\sigma_w^{(a)}$.
- Source Parties List: The list of source parties addresses, denoted $\sigma_l^{(a)}[\,]$.

One can implement these additional fields easily using Smart Contracts on existing blockchains that support them. However, they can also be implemented as standard state account fields and integrated directly into the blockchain framework. In the present work, we adopt the latter approach throughout the paper, except for the performance evaluation section, where we use smart contracts, as they lend themselves easily to experimentation.

1) CONSENSUS

Creating new blocks is controlled by a mechanism that varies between blockchain technologies. The Proof-of-Work was

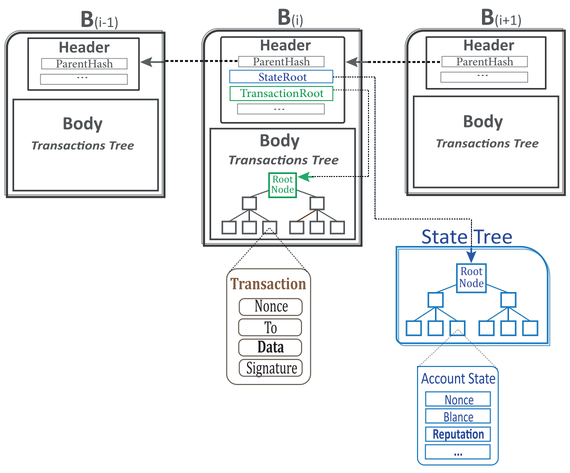

FIGURE 3. Blocks-state diagram.

the first mechanism used to reach consensus [32] on newly created blocks. A concept introduced with Bitcoin [12] and followed by several alternative coins launched using similar ideas.

The mechanism of Proof-of-Work allows miners to write to the blockchain only if they prove that they did a certain amount of work. The work consists of an extensive search for partial collision using hash functions. This mining process demands a lot of computing energy [33].

With the advent of PPCoin [34] further developed by BlackCoin [35], NXT [36] and NeuCoin [37], a new family of systems was born replacing the Proof-of-Work with the concept of "Proof-of-Stake"

From its side, Proof-of-Stake enables every participant to randomly gain the right to write to the public database with a probability proportional to his *stake*, i.e., the number of coins he owns. This approach cancels the extra computing cost of mining and leads to a much faster block-creation process. It seems perfectly reasonable today to maintain distributed consensus using Proof-of-Stake. The thing we consider more appropriate in our context than using Proof-of-Work.

During the block creation process, the validators perform various tasks. Among them the verification and recording of new transactions. A new block typically contains transactions that have occurred since the last block, and when effectively added to the blockchain, the underlying transactions are definitive. Subsequently, all the concerned accounts states are updated to reflect the changes.

B. MESSAGE AUTHENTICATION CODE

Loosely speaking, a message authentication code (MAC) is a piece of data used to authenticate a message. Like digital signatures, its purpose is to confirm that the message came from the stated sender (authenticity) and has not been changed (integrity). It allows the receiver to detect any change to the message content. However, unlike digital signatures, MAC values are generated and verified using the same secret key, which implies that the sender and receiver must agree on a key before initiating communications.

Formally, a MAC scheme is a triple of polynomial-time algorithms (Gen, Mac, Verif) such that:

- $\mathrm{Gen}(1^\lambda)$ is the key generation algorithm that takes λ the security parameter as input and samples a key $k \in K$ from the key space uniformly at random $k \leftarrow \mathrm{Gen}(1^\lambda)$;
- $\mathrm{Mac}_k(m)$, the Mac algorithm returns a tag $t = \mathrm{Mac}_k(m)$ on input key $k \in K$ and message $m \in M$;
- $\mathrm{Verif}(k, m, t)$ is the verifying algorithm that takes an input $k \in K, m \in M$, and $t \in T$, verifies the authenticity of the message and returns accepted when the message and the tag match $t = \mathrm{Mac}_k(m)$ and rejected otherwise;
- The following equality

$$\Pr\left[\begin{array}{c} k \leftarrow \mathrm{Gen}(1^\lambda), \\ \mathrm{Verif}(k, x, \mathrm{Mac}_k(x)) = accepted \end{array}\right] = 1$$

holds.

A MAC scheme is said to be *secure* (unforgeable) if for every polynomial time adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} k \leftarrow \mathrm{Gen}(1^\lambda), \\ (x, t) \leftarrow \mathcal{A}^{\mathrm{Mac}_k(\cdot)}(1^\lambda), \\ x \notin Query(\mathcal{A}^{\mathrm{Mac}_k(\cdot)}, 1^\lambda), \\ \mathrm{Verif}(k, x, t) = accepted \end{array}\right] < \epsilon(\lambda)$$

where $\mathcal{A}^{\mathrm{Mac}_k(\cdot)}$ denotes that $\mathcal{A}$ has access to the oracle $\mathrm{Mac}_k(\cdot)$, and $Query(\mathcal{A}^{\mathrm{Mac}_k(\cdot)}, 1^\lambda)$ denotes the set of the queries made by $\mathcal{A}$ on Mac() knowing λ. The function $\epsilon(\lambda)$ is *negligible* in the security parameter λ, i.e., for every nonzero polynomial function $poly()$ there exists λ_0 such that $|\epsilon(\lambda)| < \left|\frac{1}{poly(\lambda)}\right|$ for all $\lambda > \lambda_0$.

1) ONE-TIME MAC

We refer to a MAC as a one-time MAC if it uses the key at most once. It can be built from primitives like universal hashing and pairwise independent hash functions.

The simplest one-time MAC from pairwise independent hash functions is defined by a random key $\mathrm{k} = (a, b)$, and a MAC tag $\mathrm{Mac_k}(m) = (am + b) \mod p$, where p is prime.

It is shown that one-time MACs are unconditionally secure and that even quantum resources do not offer any advantage over them [38].

C. SEMI-HOMOMORPHIC ENCRYPTION

In this subsection, we introduce the Semi-Homomorphic Encryption Scheme (SHE). A cryptographic primitive that satisfies our system needs in terms of homomorphic encryption while being lighter and more efficient than other homomorphic primitives. SHE [39] is a public-key cryptosystem that satisfies a relaxed version of the *additive homomorphic property*. SHE is a tuple of algorithms (Gen, Enc, Dec) where:

- $\mathrm{Gen}(1^\lambda, p)$ is a randomized algorithm that takes as input a security parameter λ and a modulus p. It outputs a public/secret key pair (pk, sk) and a set of parameters $\mathbb{P} = (p, M, R, \mathcal{D}^d_\sigma, \mathbb{G})$ where, M and R are integers, and $\mathcal{D}^d_\sigma$ is a randomized algorithm producing always d-vectors

r of random integers such that $||\mathrm{r}||_\infty \leq \sigma$ for some $\sigma < R$ except with negligible probability. Finally, $\mathbb{G}$ is the domain of ciphertexts, an additive abelian group;
- $\mathrm{Enc}_{pk}(x, \mathrm{r})$ is a deterministic algorithm that takes as input an integer $x \in \mathbb{Z}_p$ and a vector $\mathrm{r} \in \mathbb{Z}^d$ and outputs a ciphertext $C \in \mathbb{G}$. We sometimes write $\mathrm{Enc}_{pk}(x)$ when it is not important to specify the randomness explicitly;
- $\mathrm{Dec}_{sk}(C)$ is a deterministic algorithm that takes as input a ciphertext $C \in \mathbb{G}$ and outputs $x' \in \mathbb{Z}_p \cup \{\perp\}$;
- $\mathrm{Enc}_{pk}(x, r) + \mathrm{Enc}_{pk}(x', r) = \mathrm{Enc}_{pk}(x + x', r + r')$.

A semi-homomorphic encryption scheme SHE is *correct* if:

$$\Pr\begin{bmatrix} (pk,\ sk,\ \mathbb{P}) \leftarrow \mathrm{Gen}(1^\lambda, p), \\ x \in \mathbb{Z}_p,\ |x| \leq M; \\ \mathrm{r} \in \mathbb{Z}^d,\ ||\mathrm{r}||_\infty \leq R: \\ \mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(x,\ \mathrm{r})) \neq x \mod p \end{bmatrix} < \varepsilon(\lambda)$$

with probabilities taken over randomness in Gen and Enc, and $\epsilon(\lambda)$ a negligible function.

SHE is IND-CPA-*Secure* (i.e. achieves indistinguishability under chosen plaintext attack) if for all probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ on the following experiment verifies:

$$\mathrm{Adv}^{\mathrm{CPA}}(\mathcal{A}, \lambda) = \left| \Pr\begin{bmatrix} (pk,\ sk,\ \mathbb{P}) \leftarrow \mathrm{Gen}(1^\lambda, p); \\ (m_0, m_1, state) \leftarrow \mathcal{A}_1(1^\lambda, pk) \\ m_0, m_1 \in \mathbb{Z}_p; \\ b \leftarrow \{0, 1\};\ C \leftarrow \mathrm{Enc}_{pk}(m_b); \\ b' \leftarrow \mathcal{A}_2(1^\lambda, state, C): \\ b = b' \end{bmatrix} \text{-}1/2 \right| < \varepsilon(\lambda)$$

with the probability taken over randomness in Gen, Enc, and $\mathcal{A}$.

Paillier Cryptosystem: We obtain a semi-homomorphic scheme straightforward from Paillier's cryptosystem [40] by setting the adequate parameters:
- The secret key as two large primes $sk=(p_1, p_2)$, and the public key as $pk=N=p_1p_2$, and $M=(N-1)/2$, $R=\infty$, $d=1$, $\mathcal{D}^d_\sigma = U_{\mathbb{Z}^*_{N^2}}$ (Uniform distribution), $\sigma=\infty$ and $\mathbb{G}=\mathbb{Z}^*_{N^2}$;
- $\mathrm{Enc}_{pk}(x, r)=(N+1)^x r^N \mod N^3$ where $x \in Z_N$ and r is random in $\mathbb{Z}^*_{N^2}$;
- The decryption function is simply redefined as $\mathrm{Dec}'(c) = \mathrm{Dec}(c) \mod p$ to get the reconstructed plaintexts modulo p.

We highlight that the Paillier cryptosystem is written a bit differently from the above SHE definition, which writes $\mathbb{G}$ additively, while $Z^*_{N^2}$ is written with multiplicative notation. Also for Paillier we have $\mathrm{Enc}_{pk}(x, r) + \mathrm{Enc}_{pk}(x', r) = \mathrm{Enc}_{pk}(x + x', r \cdot r')$ instead of $\mathrm{Enc}_{pk}(x + x', r + r')$. However, this makes no difference.

Several other candidates like the lattice-based scheme Brakerski-Gentry-Vaikuntanathan (BGV) [41], Regev's cryptosystem [42], [43] or the Subset Sum Cryptosystem [44] can be used interchangeably with the Paillier scheme suggested here.

D. COMMITMENT

Commitment schemes are common ingredients in cryptographic protocols. As their name suggests, they are used to enable a party to commit itself to a value while keeping it secret [45]. Once a party has committed to someone, it cannot change it later when the commitment is "opened," and other parties cannot gain knowledge about it before the opening. Commitment schemes are the digital form of non-transparent sealed envelopes.

A commitment scheme is a two-phase protocol with two players: a sender S and a receiver R. It must satisfy two requirements to be secure:

(1). Hiding (or Secrecy): At the end of the Commit phase, the receiver R does not gain any knowledge of the secret.

(2). Binding: Given the commitment from the Commit phase, there exists at most one value that the receiver can accept in the Open phase as a legal opening of the commitment.

The commitment scheme presented below for the present system uses a hash function $\mathcal{H}$ (a Random Oracle) and some randomness:

```
Protocol 1: Commitment Protocol $\Pi_{\mathrm{Commit}}$
Commit: 1) In order to commit to $x$, $S$ sets $y \leftarrow x||r$, where
           $r$ is chosen at random uniformly in a determined domain, and queries the Random Oracle $\mathcal{H}$ to get $c \leftarrow \mathcal{H}(y)$.
        2) $S$ then sends $c$ to $R$.
Open:   1) In order to open a commitment $c$, where $c = \mathcal{H}(x||r)$, player $S$ sends $y = x||r$.
        2) $R$ calls $\mathcal{H}$ on $y$ and check whether $\mathcal{H}(y) = c$.
           He accepts if and only if this check passes.
```

E. SECRET SHARING

Secret sharing is a method for splitting a secret among a group of parties such that they cannot reconstruct the secret unless a sufficient number of them unite. A dealer in a secret sharing scheme gives each party a share. But a single one is of no use on its own. For example, a t-out-of-n (t, n)-secret sharing scheme is secure if it distributes n shares so that any subset of parties with less than t shares cannot reconstruct the secret. Moreover, it cannot gain more information about it.

In our context, parties use the secret sharing technique to make joint computations on their private ratings without revealing them. Particularly two techniques are used further in SMC (see IV-G):
- The first one is an authenticated version (see IV-B) of the trivial (n, n)-additive secret sharing scheme defined as follows:

  For a secret x, the dealer D samples randomly $x^{(i)} \overset{\$}{\leftarrow} \mathbb{F}$ for $1 \leq i \leq n-1$ and sets the nth share $x^{(n)}$ such that $\sum_{i=1}^{n} x^{(i)} = x$. The result is an (n,n)-additive secret sharing of x:

$$[x]=(x^{(1)}, \ldots, x^{(n)}) \;/\; \sum_{i=1}^{n} x^{(i)}=x.$$

- The second technique used in our system based on a semi-homomorphic encryption scheme (see IV-C) allows two parties to secret-share the product of their inputs without revealing them. It works as follows:
  One party P_1 sends an encryption $\mathrm{Enc}_{pk_1}(a)$ of its input a under its own public key to another party P_2, which replies by $C=b\cdot\mathrm{Enc}_{pk_1}(a)-\mathrm{Enc}_{pk_1}(x_2)$, where b denotes party's P_2 input, and x_2 is chosen at random. Since a semi-homomorphic encryption scheme supports the multiplication of a known value with a ciphertext, hence the decryption of C is $x_1=b \cdot a - x_2$, which makes $[a \cdot b]=(x_1, x_2)$ an additive secret sharing of $a \cdot b$.

1) AN AUTHENTICATED SECRET SHARING

The idea behind authenticated secret sharing is to use one-time MACs to prevent parties from lying about their shares when they are supposed to generate them correctly. In this regard a random key $\kappa\in\mathbb{F}$ is issued, and a MAC for $a\in\mathbb{F}$ defined by $Mac_\kappa(a)=\kappa \cdot a \mod p$ is used.

For some parties $P_1, \ldots, P_n$, if κ is the MAC key and $[\kappa]=(\kappa^{(1)}, \ldots, \kappa^{(n)})$ its secret sharing, we assume that every party P_j $1\leq j\leq n$ keeps its share $\kappa^{(j)}$ secret and that κ the global key is unknown to all parties. They can easily achieve this result by generating their random shares independently and committing to them so that κ results naturally as the sum. The protocol Π_{Rand} does exactly that: $[\kappa]\leftarrow\Pi_{\mathrm{Rand}}$.

Protocol 2: The Protocol Π_{Rand}

1) Every party P_i samples $r^{(i)} \overset{\$}{\leftarrow}\mathbb{F}$ commit to it and keeps it private.
2) Take $r=\sum_{i=1}^{n} r^{(i)}$ such that $[r]=(r^{(1)}, \ldots, r^{(n)})$ and r is random and unknown to all parties until the shares are opened.

Definition 0.1: We call an authenticated secret sharing of $x\in\mathbb{F}$ under a global key κ the ordered set:

$$[[x]]=(x^{(1)}, \ldots, x^{(n)}, m(x)^{(1)}, \ldots, m(x)^{(n)}, \kappa^{(1)}, \ldots, \kappa^{(n)})$$

Each player P_i holds its authenticated share tuple $[[x]]^{(i)}=(x^{(i)}, m(x)^{(i)}, \kappa^{(i)})$ such that: $x=\sum_{i=1}^{n} x^{(i)}$, $m(x)=\mathrm{Mac}_\kappa(x)=\sum_{i=1}^{n} m(x)^{(i)}$, and $\kappa=\sum_{i=1}^{n}\kappa^{(i)}$.
If a party P_i, for example, wants to generate the authenticated shares for its private input x, then it has to share x, $m(x)$ under κ:

- For κ, nothing needs to be done. It is already shared since every party P_j has its private key share κ_j. If not $[\kappa]\leftarrow\Pi_{\mathrm{Rand}}$ is executed.
- Sharing x is also straighforward like explained above $[x]=(x^{(1)}, \ldots, x^{(n)})$ such that $\sum_{i=1}^{n} x^{(i)}=x$.
- For $m(x)=\mathrm{Mac}_\kappa(x)=\kappa \cdot x$, we know for sure that P_i cannot share it on its own since it does not know κ. So it needs some interaction and the semi-homomorphic technique mentioned above:
  For that each party P_j sends an encryption $\mathrm{Enc}_{pk_j}(\kappa_j)$ of its key share under its own public key pk_j to P_i, which replies by $c_j=x \cdot \mathrm{Enc}_{pk_j}(\kappa_j) - \mathrm{Enc}_{pk_j}(e_j)$, where e_j is chosen at random. Since semi-homomorphic encryption schemes allow the addition of ciphertexts and multiplication of ciphertexts by cleartext, hence if P_j decrypts c_j, it gets $\mathrm{Dec}_{sk_j}(c_j)=x \cdot \kappa_j - e_j$. P_j then sets its share to $m(x)^{(j)}=x \cdot \kappa_j - e_j$. P_i takes $m(x)^{(i)}=x \cdot \kappa_i + \sum_{j\neq i} e_j$ so that $\sum_{j=1}^{n} m(x)^{(j)}=x \cdot \kappa$. This leads to $[m(x)]=(m(x)^{(1)}, \ldots, m(x)^{(n)})$ an additive secret sharing of $m(x)=x \cdot \kappa$.

F. ZERO-KNOWLEDGE PROOFS

A Zero-knowledge (ZK) proof allows a verifier $\mathcal{V}$ to check the validity of a statement claimed by a prover $\mathcal{P}$ without revealing anything other than the claim being true.

An example of ZK proof of knowledge is the Schnorr-like protocol based on the standard 3-move Σ_- protocol. Its goal is to prove knowledge of x in a field $\mathbb{F}$, such that $f(x)=y$ without revealing x. The protocol works as follows:

1) The prover $\mathcal{P}$ samples a random $s\overset{\$}{\leftarrow}\mathbb{F}$ and sends a commitment $a=f(s)$ for s to the verifier.
2) The verifier $\mathcal{V}$ then samples a random $c\overset{\$}{\leftarrow}\mathbb{F}$ and sends it to $\mathcal{P}$.
3) $\mathcal{P}$ replies with $z=s+c\cdot x$. Finally $\mathcal{V}$ checks whether $f(z)=a+c\cdot y$.

where f is homomorphic for the field operations.

- *Correctness* the protocol is correct if f is homomorphic for the field operations.
- *Honest-Verifier Zero-Knowledge* or Security of the Prover is achieved by simulating (a, c, z) from any c by sampling $z\overset{\$}{\leftarrow}\mathbb{F}$ and computing $a=f(z)-c\cdot y$.
- *Special Soundness* or Security for the Verifier is obtained by extracting the secret from two different transcripts (z, c), (z', c') with $c\neq c'$ which can be done by computing $x=(z-z')\cdot(c-c')^{-1}$ (the operations are possible in a field).

The Non-interactive Zero-Knowledge (NIZK) [11] version of this scheme is achieved through Fiat-Shamir transformation [46] using a secure cryptographic hash function (Random Oracle Model). Instead of the verifier issuing the challenge c, It is simply redefined as $c=\mathcal{H}(pp||y||a)$, where pp is some public parameter. The hash function $\mathcal{H}$ should be a secure cryptographic hash function, e.g., SHA or SHA3, with a bitlength at least equal to the order of the considered subgroup.

Protocol 3: Non-Interactive Zero-knowledge Proof of knowledge Π_{NIZKPoK}

1) The prover $\mathcal{P}$ samples a random $s\overset{\$}{\leftarrow}\mathbb{F}$ compute a commitment $a=f(s)$ to s, and the challenge $c=\mathcal{H}(pp||y||a)$ then sends (a, c, z) where $z=s+c\cdot x$.

2) The verifier $\mathcal{V}$ computes $c = \mathcal{H}(pp||y||a)$ and checks whether $f(z) = a + c \cdot y$.

Π_{NIZKPoK} is used extensively in our system with function f set to the $\text{Enc}_{pk}()$, the encryption algorithm of a SHE protocol.

G. SECURE MULTIPARTY COMPUTATION

Secure Multiparty Computation (SMC) allows a group of parties to compute a function jointly over their private inputs. For some parties $P_1, \ldots, P_n$, each party P_i holding a secret input x_i and f an agreed-on function that takes n inputs, An SMC protocol Π enables $P_1, \ldots, P_n$ to compute $y = f(x_1, \ldots, x_n)$ while satisfying the following conditions:

- Correctness: the protocol Π computes the correct value of y;
- Privacy: The parties $P_1, \ldots, P_n$ cannot learn additional information from the protocol but y.

Usually, the desired function f is represented by the equivalent boolean or arithmetic circuit.

Currently, to realize SMC one has two paradigms to choose from, secret sharing [47], [48], [49], [50] and garbled circuits [51], [52], [53]. Both paradigms come with their strengths, and both have their lines of development. Since secret sharing is currently the most suitable to evaluate arithmetic circuits, we adopt it for our protocol.

The proposed system uses a general SMC protocol in the preprocessing model [47], [48], [49], [54] built upon the blocks presented above. It is a two-phase protocol inspired by the Overdrive protocol [49], an efficient improvement of SPDZ protocols [47], [48]. The preprocessing phase executed individually precedes the online phase that involves interaction. From its side, the online phase where players share inputs via additive secret sharing uses cheap information-theoretic primitives. While in the other side, the preprocessing phase uses a homomorphic encryption technique to generate random triples independently from inputs for further use in the online phase.

The SMC protocol guarantees privacy and correctness by using commitment and zero-knowledge proofs and authenticating shared values with information-theoretic MACs as presented in subsection IV-E1.

Formally, every P_i executes $\Pi_{[\![\cdot]\!]}$.Input (see protocol 5) on its private input $x_i \in \mathbb{F}$, to generate and share $[\![x_i]\!] = (x_i^{(1)}, \ldots, x_i^{(n)}, m(x_i)^{(1)}, \ldots, m(x_i)^{(n)}, \kappa^{(1)}, \ldots, \kappa^{(n)})$ the authenticated secret sharing of x_i, where each player P_j s.t. $1 \leq j \leq n$ holds a share tuple $[\![x_i]\!]^{(j)} = (x_i^{(j)}, m(x_i)^{(j)}, \kappa^{(j)})$ verifying: $x_i = \sum_{j=1}^{n} x_i^{(j)}$, $\kappa \cdot x_i = \sum_{j=1}^{n} m(x_i)^{(j)}$, and $\kappa = \sum_{j=1}^{n} \kappa^{(j)}$.

In the online phase, the main task is to evaluate a circuit over secret inputs, which the parties do by performing operations like addition and multiplication on authenticated shared values, c.f., protocol 6.

To open the result $[\![y]\!]$ at the end of calculations, all players P_i broadcast their shares $y^{(i)}$, commit and then open $m(y)^{(i)} - \kappa^{(i)} y$. Afterward, they check if the sum of the latter is equal to zero, as shown in the Π_{MACCheck} protocol 4.

Since the addition is linear, it can be done via local computation. However, multiplying two private values $[\![x]\!]$, $[\![y]\!]$ requires some interaction between parties. To compute $[\![x \cdot y]\!]$ a fresh random triple $[\![a]\!], [\![b]\!], [\![c]\!]$ has to be available to execute Beaver's trick [55]. This is the goal of the pre-processing phase which prepare independent random tuples $([\![a]\!], [\![b]\!], [\![c]\!])$ with $c = a \cdot b$ and a, b for the purpose.

Once parties have a fresh tuple $([\![a]\!], [\![b]\!], [\![c]\!])$ they follow Beaver's method by opening $\alpha = [\![x - a]\!]$ and $\beta = [\![y - b]\!]$ and then get the authenticated product by setting $[\![x \cdot y]\!] \leftarrow [\![c]\!] + \alpha[\![b]\!] + \beta[\![a]\!] + \alpha \cdot \beta$.

Protocol 4: Π_{MACCheck}
Each party in $\{P_i\}_{1 \leq i \leq n}$ uses y, $m(y)^{(i)}$, $\kappa^{(i)}$ in the following way:

1) Computes $\delta^{(i)} \leftarrow m(y)^{(i)} - \kappa^{(i)} y$;
2) Execute Π_{Commit}.Commit on $\delta^{(i)}$ to receive and broadcast c_i;
3) Broadcast $\delta^{(i)}$ to all parties by executing Π_{Commit}.Open on $(\delta^{(i)}, c_i)$;
4) If $\sum_{i=1}^{n} \delta^{(i)} \neq 0$ then abort and output $\perp$; otherwise continue.

---

1) ONLINE PHASE

Protocol 5: $\Pi_{[\![\cdot]\!]}$

- **Initialize:** Each party P_i does the following:
  1) Sample a MAC key $\kappa^{(i)} \xleftarrow{\$} \mathbb{F}$;
  2) Initialize two instances of Π_{NIZKPoK} with every other party P_j (one as prover, one as verifier), receiving $(\text{pk}_i, \text{sk}_i)$ and pk_j;
  3) Using Π_{NIZKPoK}, send an encryption $\text{Enc}_\kappa(\kappa^{(i)})$ to every other party.
- **Input:** On input x_j from P_j:
  1) For the input x_j, P_j samples randomly $x_j^{(i)} \xleftarrow{\$} \mathbb{F}$ for each party P_i. Then P_j sets its corresponding share $x_j^{(j)}$ accordingly such that $\displaystyle\sum_{i=1}^{n} x_j^{(i)} = x_j$;
  2) For every party P_i:
     a) P_j computes $C_j^{(i)} = \text{x}_\text{j} \cdot \text{Enc}_{pk_i}(\kappa^{(i)}) - \text{Enc}_{pk_i}(\text{e}_j^{(i)})$ for random $\text{e}_j^{(i)}$;
     b) P_j sends $\{x_j^{(i)}, C_j^{(i)}\}$ to P_i;
     c) P_i decrypts $d_j^{(i)} = \text{Dec}_{\text{sk}_i}(C_j^{(i)})$.
  3) P_j sets its MAC share associated to x_j as $m(x_j)^{(j)} \leftarrow \displaystyle\sum_{i \neq j} e_j^{(i)} + x_j \cdot \kappa^{(j)}$ and each party P_i does so for $m(x_j)^{(i)} \leftarrow d_j^{(i)}$;
  4) All parties store their authenticated shares $(x_j^{(i)}, m(x_j)^{(i)}, \kappa^{(i)})$ of $[\![x_j]\!]$.
- **Compute:** To compute $[\![y]\!] = f([\![x_1]\!], \cdots, [\![x_n]\!])$, every party P_i computes from its authenticated shares $(x_j^{(i)}, m(x_j)^{(i)}, \kappa^{(i)})$ of $[\![x_j]\!]$, $1 \leq j \leq n$:

1) $y^{(i)}=f(x_1^{(i)}, \cdots, x_n^{(i)})$;
2) $m(y)^{(i)}=f(m(x_1)^{(i)}, \cdots, m(x_n)^{(i)})$.

Elementary operations are detailed in Protocol 6, $[\![\cdot]\!]$-operations.

- **Open:** To open $[\![y]\!]$, from all parties, each P_i broadcasts the share $y^{(i)}$. Then each party reconstructs $y=\sum_{i=1}^{n} y^{(i)}$;
- **Check:** All parties run Π_{MACCheck} protocol with y, $m(y)^{(i)}$, $\kappa^{(i)}$.

There is a clear distinction in SMC operations between public and private operands. We can always perform calculations on public operands the usual way. However, private operands need to be shared and authenticated before any operation. After secret-sharing private operands between parties, a party can perform calculations only on the shares it got and thus cannot get the global result. Instead, it gets a partial result that is meaningless on its own, but if gathered with others, it serves to reconstruct (open) the global one acting as a share for it.

Protocol 6: $[\![\cdot]\!]$-operations

- *Addition:* Given two private inputs $[\![a]\!]$ and $[\![b]\!]$ returns: $[\![a]\!] + [\![b]\!]=(a^{(1)} + b^{(1)}, \ldots, a^{(n)} + b^{(n)}; m(a)^{(1)} + m(b)^{(1)}, \ldots, m(a)^{(n)} + m(b)^{(n)}; \kappa^{(1)}, \ldots, \kappa^{(n)})$. (can be computed only by local operations);
- *Add a public value:* Given a private inputs $[\![a]\!]$ and a public one δ, returns $[\![a]\!] + \delta=[\![a + \delta]\!]=(a^{(1)} + \delta, a^{(2)}, \ldots, a^{(n)}; m(a)^{(1)} + \delta \cdot \kappa^{(1)}, \ldots, m(a)^{(n)} + \delta \cdot \kappa^{(n)}; \kappa^{(1)}, \ldots, \kappa^{(n)})$;
- *Multiply by a public value:* Given a private inputs $[\![a]\!]$ and a public one λ, returns $\lambda \cdot [\![a]\!]=(\lambda \cdot a^{(1)}, \ldots, \lambda \cdot a^{(n)}; \lambda \cdot m(a)^{(1)}, \ldots, \lambda \cdot m(a)^{(n)}; \kappa^{(1)}, \ldots, \kappa^{(n)})$;
- *Multiply:* To multiply $[\![x]\!]$ and $[\![y]\!]$ the parties do the following, namely:
  1) They take a triple $([\![a]\!], [\![b]\!], [\![c]\!])$ from the preprocessing phase,
  2) $\alpha=[\![x]\!] - [\![a]\!]$ and $\beta=[\![y]\!] - [\![b]\!]$ are opened,
  3) They set $[\![x]\!] \cdot [\![y]\!]=[\![c]\!] + \alpha[\![b]\!] + \beta[\![a]\!] + \alpha \cdot \beta$.

We demonstrate further in subsection VII-A when discussing the proposed system correctness that:

Theorem 1: The operations presented in protocol 6 are fully compatible with the authenticated secret sharing representation, namely:

$$[\![a]\!] + [\![b]\!] = [\![a + b]\!]$$
$$[\![a]\!] + \delta = [\![a + \delta]\!]$$
$$\lambda \cdot [\![a]\!] = [\![\lambda a]\!]$$
$$[\![x]\!] \cdot [\![y]\!] = [\![x \cdot y]\!].$$

Also, we highlight that all operations above can be performed locally without interaction except for the multiplication of two private inputs. This operation requires the preprocessing phase. Likely in all known reputation systems so far, aggregation functions do not need it (c.f. V). Despite this, we included this phase in the appendix for generality's sake and eventual use in the future.

As mentioned above, operations with only public operands do not require special treatment. Only calculations that involve private operands are relevant for SMC. We concentrate on the following section on this type of operation when reviewing reputation aggregation functionalities by identifying relevant parts and leaving others as is.

V. REPUTATION REPRESENTATION AND COMPUTATION IN USUAL SCENARIOS

In the following, we investigate formats and models used to represent and compute reputation from ratings in typical scenarios. We aim to determine precisely the types of functionalities a reputation system in such scenarios has to calculate over ratings and present how the proposed protocol implements them.

From the SMC viewpoint, ratings are inputs, and reputation is output. However, the SMC protocol is more costly than regular computation. For this reason, we prefer isolating parts of these functionalities that must be securely computed using SMC to preserve privacy from parts that can be computed publicly (in clear) without compromising privacy.

Regarding rating representation, several formats are employed in the literature to describe and exchange rating information. The following types are used traditionally to represent them:

- *The Binary* format is used in the case ratings are either positive or negative regardless of their values, and they are stored using boolean values [56], [57], [58].
- *Discrete* is employed when ratings take several values on a scale. They are usually stored using discrete integer values [58].
- *Continuous*, where the information is stored as a floating point number [57], [58].
- *String:* The information is stored in textual form, allowing a wide range of data to be maintained [56], [59].

Regarding reputation computation, there are several approaches to aggregating ratings into a reputation score. The simplest way consists of counting the number of positive and negative ratings for an entity and summing the two numbers [60]. The final score serves to rank all entities in the system.

A slightly better way is to average all ratings to get a single score for each entity. Furthermore, weighting ratings can enhance this approach by providing more importance to, for example, recent ratings or those from more reputable sources [61]. Additionally, normalization can be used to evaluate reputation on a specific scale.

In the following, we sum up the different approaches in the literature and specify how our protocol applies to each one:

A. COUNTING

Counting, as introduced above, relies on standard techniques like *summation*, *averaging*, and *weighting* to compute reputation. Some authors [62] adopt a more natural way for humans to represent ratings [60] like Very Trustworthy, Trustworthy, Untrustworthy, and Very Untrustworthy. Such representations are hard to work with before converting them to numerical values. If converted, the counting method is then applied (average, for example), and the result is converted back using look-up tables.

According to the adopted counting method, applying SMC to compute either the sum, the average, or the weighted average of parties' private ratings is straightforward. It does not need any adaptation or change. It suffices to run $\Pi_{[\![.]\!]}$ (protocol 5) on ratings as inputs.

Example: Let us say parties P_1, P_2, and P_3 want to compute the reputation score of a target party P_t from their ratings x_1, x_2, and x_3 respectively, and that the reputation function is the average weighted by the raters' actual reputation scores, namely w_1, w_2, and w_3. In this case the reputation of P_t will be:

$$w_t = f(x_1, x_2, x_3) = \frac{w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3}{3}$$

However, P_1, P_2, and P_3 do not want to reveal their ratings, and there is no trusted party to do the computation for them. So, they run $[\![\cdot]\!] \cdot \mathrm{Input}()$ on their private ratings so that each party P_i gets a share of the three ratings, namely:

- P_1 gets $(x_1^{(1)}, m(x_1)^{(1)}, \kappa^{(1)})$, $(x_2^{(1)}, m(x_2)^{(1)}, \kappa^{(1)})$, and $(x_3^{(1)}, m(x_3)^{(1)}, \kappa^{(1)})$.
- P_2 gets $(x_1^{(2)}, m(x_1)^{(2)}, \kappa^{(2)})$, $(x_2^{(2)}, m(x_2)^{(2)}, \kappa^{(2)})$, and $(x_3^{(2)}, m(x_3)^{(2)}, \kappa^{(2)})$,
- and P_3 gets $(x_1^{(3)}, m(x_1)^{(3)}, \kappa^{(3)})$, $(x_2^{(3)}, m(x_2)^{(3)}, \kappa^{(3)})$, and $(x_3^{(3)}, m(x_3)^{(3)}, \kappa^{(3)})$.

The shares $(x_i^{(j)}, m(x_i)^{(j)}, \kappa^{(j)})$, $1 \leq i, j \leq 3$ are randomly generated and shared using Beaver's method and SHE according to protocol $[\![\cdot]\!] \cdot \mathrm{Input}()$ and they verify by construction:

- $x_1 = x_1^{(1)} + x_1^{(2)} + x_1^{(3)}$ and $m(x_1) = m(x_1)^{(1)} + m(x_1)^{(2)} + m(x_1)^{(3)}$;
- $x_2 = x_2^{(1)} + x_2^{(2)} + x_2^{(3)}$ and $m(x_2) = m(x_2)^{(1)} + m(x_2)^{(2)} + m(x_2)^{(3)}$;
- $x_3 = x_3^{(1)} + x_3^{(2)} + x_3^{(3)}$ and $m(x_3) = m(x_3)^{(1)} + m(x_3)^{(2)} + m(x_3)^{(3)}$;
- $\kappa = \kappa^{(1)} + \kappa^{(2)} + \kappa^{(3)}$ and $m(x_i) = \kappa \cdot x_i$ for $1 \leq i \leq 3$ where κ is the MAC key.

The next step is $[\![\cdot]\!] \cdot \mathrm{Compute}()$ where each party P_i for $1 \leq i \leq 3$ computes locally:

$$w_t^{(i)} = \frac{w_1 \cdot x_1^{(i)} + w_2 \cdot x_2^{(i)} + w_3 \cdot x_3^{(i)}}{3},$$

and

$$m(w_t)^{(i)} = \frac{w_1 \cdot m(x_1)^{(i)} + w_2 \cdot x_2^{(i)} + w_3 \cdot m(x_3)^{(i)}}{3}.$$

We can verify easily that $w_t = w_t^{(1)} + w_t^{(2)} + w_t^{(3)}$. For the general correctness demonstration, the reader can refer to section VII.

Next, all P_i for $1 \leq i \leq 3$ commit and broadcast $w_t^{(i)}$, and compute w_t by executing $[\![\cdot]\!] \cdot \mathrm{Open}()$, which calculates $w_t = w_t^{(1)} + w_t^{(2)} + w_t^{(3)}$ after requiring parties to commit to their shares $w_t^{(i)}$ before broadcasting them. As $w_t^{(i)}$ does not reveal any information about x_i for $1 \leq i \leq 3$, it is safe to broadcast them. However, to prevent parties from changing $w_t^{(i)}$ values according to others' values during the protocol execution commitment is required.

The last step for parties before accepting w_t as the correct reputation score is to verify that no party has sent false shares or wrong values to manipulate results or induce other parties in error. For that every party P_i calls Π_{MACCheck} on $w_t, m(w_t)^{(i)}, \kappa^{(i)}$. If the check is positive, they are sure that the computation is correct.

The Counting scenario is summarized in table 2.

TABLE 2. Counting scenario summary.

Approach	*Counting*
Systems	eBay [63] SuperTrust [64] TrustGuard [65] PET [66] PeerTrust [67]
Representation	Binary / Discrete / Continuous
functionality	Sum, Average, Weighted Average

B. PROBABILISTIC

Another aggregation method involves using a probability model to predict future behavior from the knowledge of prior events by fitting the latter as input and computing the likelihood of a hypothesis to hold like "a party x will behave correctly" [58], [61], [68], [69], [70], [71]. Probabilistic systems are generally based on the Bayesian model. They take binary ratings as input (i.e., positive or negative) and compute reputation scores by statistically updating a beta probability density function (PDF). The beta distribution takes two free parameters, α, and β, respectively, the number of positive and negative feedback. The reputation score is the tuple of Beta PDF parameters $(\alpha; \beta)$, or the Expectation value of the beta PDF, and optionally accompanied by the Variance or a Confidence parameter. Besides, the tuple $(\alpha; \beta)$ must be stored and updated with new rating occurrences. Otherwise, we cannot update the Expectation value or the Variance without the two parameters. The Beta PDF denoted by $\mathrm{Beta}(p|\alpha; \beta)$ is expressed using the gamma function Γ as:

$$\mathrm{Beta}(p|\alpha; \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1},$$

where $0 \leq p \leq 1$ and $\alpha; \beta > 0$. The probability expectation of the beta distribution is given by: $\mathbb{E}(p) = \alpha/(\alpha + \beta)$.

From the above, we can represent and store the reputation score as the tuple $(\alpha; \beta)$ without deviating from the adopted approach or losing privacy. Since the tuple $(\alpha; \beta)$ does not reveal any of the individual ratings, the Beta PDF can be applied to it by the system in a second step as convenient. On the contrary, if we store just the beta PDF value without the tuple $(\alpha; \beta)$, it will be impossible to recalculate its value when new ratings enter the system.

Let us encode ratings as vectors in $\mathbb{F}^2$, namely, the positive rating as $(1, 0)$ and the negative as $(0, 1)$, and then take $(\alpha; \beta)$ as the sum of previous ratings; therefore α and β correspond to the number of positive and negative feedback respectively. The aggregation function here is simply the sum.

For example, adding a negative rating to the reputation tuple translates literally to $(\alpha; \beta) + (0, 1)=(\alpha + 0; \beta + 1)= (\alpha; \beta+1)$ according to the usual operations in vector space $\mathbb{F}^2$.

We also highlight that the SMC protocol applies to vector inputs without any change. The only exception is for randomly generated shares that now have the form: $x^{(i)}= (a^{(i)}, b^{(i)})$, $1\leq i\leq n$ for an input $x=(a, b)$ with $a=\sum_{i=1}^{n} a^{(i)}$, $b=\sum_{i=1}^{n} b^{(i)}$, and $x=\sum_{i=1}^{n} x^{(i)}$ respecting the operations in $\mathbb{F}^2$:

- $(a, b) + (c, d)=(a + c, b + d)$
- $\lambda \cdot (a, b)=(\lambda a, \lambda b)$

A summary of the probabilistic scenario is presented in table 3.

TABLE 3. Probabilistic scenario summary.

Approach	*Probabilistic*
Systems	Beta [72] Travos [69] Buchegger's [73]
Representation	positive rating := (1,0); negative rating := (0,1);
functionality	Sum
Other non SMC computation	$\mathrm{beta}(p\vert\alpha;\beta)$, $E(p)=\alpha/(\alpha+\beta)$

C. FUZZY

Some authors [74], [75], [76] also explored Fuzzy logic for aggregating reputation, using fuzzy rules to combine several criteria in the reputation score. Fuzzy logic is used to process ratings while allowing these systems to work with uncertainty [60], [61]. For example, the FuzzyTrust [74] system aggregates local trust scores collected from all peers to produce a global reputation for each peer. The system uses fuzzy inference to get the reputation aggregation weights from three variables: the peer reputation, the transaction date, and the transaction amount. It calculates the global reputation according the following formula: $R_i=\frac{\sum_{j\in S} w_j t_{ji}}{\sum_{j\in S} w_j}$ where R_i is the global reputation of peer i, S the set of peers with whom peer i has transacted, t_{ji} the local score of peer i rated by peer j, and w_j the aggregation weight of t_{ji}.

We note that the aggregation function, in this case, is R_i, the weighted average of $\{t_{ji}\}_{j\in S}$. The system infers aggregation weights $\{w_j\}_{j\in S}$ using fuzzy logic from three public variables, which makes them also public, unlike the local scores t_{ji} meant to stay private.

Based on the above, SMC can be applied without change. The Fuzzy scenario is summarized in table 4.

D. FLOW

Flow computes reputation by processing the flow of transitive trust [3], [60], [61], [78], [79], [80]. For instance, in [3], each peer i can store the number of satisfactory transactions it has had with peer j as $\mathrm{sat}(i, j)$, and the number of unsatisfactory ones as $\mathrm{unsat}(i, j)$. Then s_{ij} is defined as $s_{ij}=\mathrm{sat}(i, j) - \mathrm{unsat}(i, j)$. The local trust value s_{ij}, is then normalized as $c_{ij}=\frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}$ to ensures, all values are between 0 and 1. The peer i then aggregates the normalized local trust values of his friends towards j, $t_{ij}=\sum_k c_{ik}c_{kj}$, by asking for their opinions and weighting their opinions by the trust he places in them. The system enables a peer i to extend from his acquaintances to his friends' friends and beyond. At each circle of friends, the local trust scores they give are weighted by the trust scores their intermediate friends put on them. The Eigentrust system has modeled this transitive process elegantly using matrix calculation.

TABLE 4. Fuzzy scenario summary.

Approach	*Fuzzy*
Systems	FuzzyTrust [74] REGRET [76] P2PRep [77]
Representation	Discrete
functionality	Weighted average

Of course, this is not a global reputation score in the proper sense of the word, but the more the circles of friends are large, the more accurate and stable the score obtained.

The very reason for the existence of this system is the inability to aggregate local scores directly into a global score and the absence of the necessary infrastructure for that. Otherwise, it would be more than enough to take $t_j=\sum_k t_k c_{kj}$ if the system allowed it. Therefore, the reputation aggregation function of such a system reduces to a weighted average, and even the transitive aspect will disappear if the system supports global aggregation. Also, the approach for SMC remains the same in this case.

We present a summary of the Flow scenario in table 5.

TABLE 5. Flow scenario summary.

Approach	*Flow*
Systems	Credence [80] EigenTrust [3] NICE [81] PowerTrust [79]
Representation	Binary
functionality	Weighted average

E. OUR APPROACH FOR REPUTATION AGGREGATION

Let us assume in the beginning that a party a has formed an opinion on the behavior and quality of action of a target party t.

We state that whatever the approach adopted for the reputation aggregation and the type used to represent the rating of t by party a, we can always convert it to a numerical form like shown above and consequently encode it as a string of bits with adequate precision, e.g., 32bits or 64bits. This form lends itself easily to operations like summation, averaging, weighting, normalization, and other operations. At this end, we choose p big enough to encode all possible ratings and a wide range of shares values as elements of $\mathbb{Z}_p$ (take $p>2^{32}$, $p>2^{64}$).

Let $\rho_t^{(a)} \in \mathbb{Z}_p$ denote party's a rating on party t. A party a is said to be *a source party* for party t if a has feedback on t. The set of all source parties of a party t is given as $\mathcal{S}_t$.

Definition 1.1 (Reputation): . Let $\mathcal{S}_t = \{a_1 ... a_n\}$ be the set of source parties t. Let Rep denote the reputation function, such that $Rep : (\mathbb{Z}_p)^n \rightarrow \mathbb{R}$. The reputation of party t is given as:

$$\mathcal{R}_t = Rep(\rho_t^{(a_1)}, \ldots, \rho_t^{(a_n)}).$$

VI. THE REPUTATION PROTOCOL

A. PROTOCOL OUTLINE

The protocol assumes that after the target party performs some action, the interacting parties will get a piece of data denoted *Trace* as proof of interaction. They can join the list of source parties and compute reputation jointly based on their ratings based on the quality of that action. Afterward, they submit the resulting score to the blockchain for integration.

If a party is new or has a bad reputation, it cannot join the list of source parties and thus cannot rate other parties. Of course, it can continually improve its reputation by behaving well and receiving positive feedback from its peers. However, only parties with reputation scores exceeding a predefined threshold Tr can rate others.

The protocol comprises three phases presented below; see also, Fig. 4:

1) *Joining the list of source parties, and subgroups formation:* It is mostly an on-chain phase where the list of source parties is updated in the target party's state account;
   a) Interacting parties join the list of source parties;
   b) Joining parties are placed in a queue and then assigned to subgroups of k where the first coming is the first served;
   c) Each party gets a corresponding subgroup U of k parties from the list;
2) *Running multiparty computation on private ratings:* It is an off-chain phase where each source party in U:
   a) generates shares for its private rating;
   b) sends shares to peers and receives theirs;
   c) evaluates the reputation function on its shares, sign, and commits to the result;
   d) All parties broadcast their signed results, open them, and check the sum of their MACs;
   e) lastly, they submit the signed results to the blockchain as a joint transaction;
3) *Reputation update on the blockchain:* performed by miners. The miner that gained the right to form the current block according to the PoS performs the following actions:
   a) checks source parties membership;
   b) verifies results authenticity;
   c) Updates reputation data.

B. PROTOCOL SPECIFICATION

1) PHASE 1: JOIN THE TARGET SOURCE PARTIES' LIST

Protocol 7: Phase 1:
1) Every interacting party issues a JOIN transaction to be added to the list of t's source parties $\sigma_l^{(t)}[\,]$ in the form $<Nonce, t, Join, Trace, Signature>$;
2) The miner adds the address of a party requesting to join the list of t's source parties to $\sigma_l^{(t)}[\,]$ if:
   a) the *Trace* it sent is valid and not used before in previously recorded JOIN transactions;
   b) and its reputation score is $> T_r$;
3) Each party gets an assigned subgroup of source parties according to its order in the queue.

2) PHASE 2: SECURE MULTIPARTY COMPUTATION

Any source party can initiate this phase by requesting interaction with other parties in U or issuing the transaction at the end. Let us refer to that party by a_1 for simplicity and to the set of selected parties by $U = \{a_1, \ldots, a_k\}$.

Protocol 8: Phase 2:
1) a_1 sends requests to selected participants in U to start SMC computation;
2) All parties call $[\![\cdot]\!] \cdot$ Initialize() (see protocol 5);
3) Every party a_i calls $[\![\cdot]\!] \cdot$ Input() on its private input $\rho_t^{(a_i)}$ so that each party a_j gets a share of it $[\![\rho_t^{(a_i)}]\!]^{(j)} = ((\rho_t^{(a_i)})^{(j)}, m(\rho_t^{(a_i)})^{(j)}, \kappa^{(j)})$;
4) Each a_i evaluates $[\![y]\!]^{(i)} = Rep([\![\rho_t^{(a_1)}]\!]^{(i)}, \ldots, [\![\rho_t^{(a_k)}]\!]^{(i)})$
5) All a_i broadcast $[\![y]\!]^{(i)}$ and compute y by calling $[\![\cdot]\!] \cdot$ Open();
6) Call Π_{MACCheck} on $y, m(y)^{(i)}, \kappa^{(i)}$;
7) If the Π_{MACCheck} was successful, then a_1 issues a RATE transaction in the form

$$Tx_U = <(\sigma_n^{(a_1)}, \ldots, \sigma_n^{(a_k)}), t, Rate, ([\![y]\!]^{(1)}, \ldots, [\![y]\!]^{(k)}), (Sig_1, \ldots, Sig_k)>$$

where $Sig_i = Sign([\![y]\!]^{(i)}, sk_i)$ for $1 \leq i \leq k$.

3) PHASE 3: REPUTATION AGGREGATION ON-CHAIN (PERFORMED BY MINERS)

Protocol 9: Phase 3: The miner that gained the right to form the current block performs the following actions:
1) Verify the Transaction:
   a) Verifies the transaction signatures are valid;
   b) Recovers signers' addresses from their signatures;
   c) Checks the source parties are in $\sigma_l^{(t)}[\,]$;
   d) Verify that the nonces in the transaction match the ones in each party account.
2) Include the transaction in the current mined block;

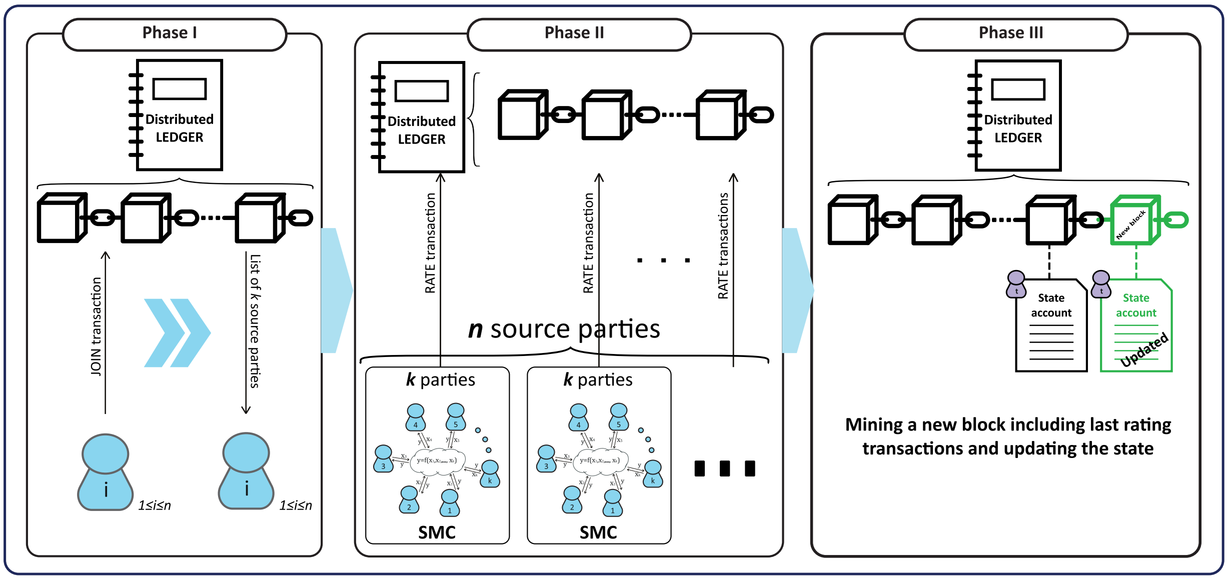

FIGURE 4. System overview diagram.

3) Reconstructs the reputation score from the received results $\mathcal{R}_t = \sum_{a_j \in U} [\![y]\!]^{(j)}$;
4) Update the reputation score and nonces in the account state of t and $\{a_i\}_{1 \le i \le k}$ respectively:
   a) $\sigma_n^{(a_i)} = \sigma_n^{(a_i)} + 1$ for all $a_i \in U$;
   b) $\sigma_r^{(t)} = \dfrac{\sigma_r^{(t)} \times \sigma_w^{(t)} + \mathcal{R}_t \times k}{\sigma_w^{(t)} + k}$;
   c) $\sigma_w^{(t)} = \sigma_w^{(t)} + k$.
5) Remove the participating parties from the source parties list $\sigma_l^{(t)}[\,]$.

VII. SECURITY ANALYSIS

A. CORRECTNESS

To demonstrate the protocol's correctness, we check that it precisely computes the desired reputation aggregation function *Rep*. More specifically, we examine the function equivalent arithmetic circuit gate by gate. Also, because addition and multiplication form a complete basis for arithmetic circuits, checking that their respective gates compute the proper values is sufficient.

For simplicity reason, we distinguish between operations that involve two, one, or no private inputs, which leads us to four cases:

- *Addition with one private input:* Given a private inputs $[\![a]\!]$ and a public one δ, the sum was defined as:

$$[\![a]\!] + \delta = (a^{(1)} + \delta,\ a^{(2)},\ \dots,\ a^{(k)};\ m(a)^{(1)} + \delta \cdot \kappa^{(1)},\ \dots,\ m(a)^{(k)} + \delta \cdot \kappa^{(k)};\ \kappa^{(1)},\ \dots,\ \kappa^{(k)})$$

We verify that,

$$\begin{aligned} a^{(1)} + \delta + a^{(2)} + \dots + a^{(k)} &= \sum_{i=1}^{n} a^{(i)} + \delta \\ &= a + \delta \end{aligned}$$

and

$$\begin{aligned} \sum_{i=1}^{k} m(a)^{(i)} + \delta \cdot \kappa^{(i)} &= \sum_{i=1}^{k} m(a)^{(i)} + \sum_{i=1}^{k} \delta \cdot \kappa^{(i)} \\ &= m(a) + \delta \cdot \kappa \\ &= \kappa \cdot (a + \delta) \\ &= m(a + \delta) \end{aligned}$$

thus $[\![a]\!] + \delta = [\![a + \delta]\!]$;

- *Addition of two private inputs:* Given two private inputs $[\![a]\!]$ and $[\![b]\!]$ the protocol defines the sum as follows:

$$[\![a]\!] + [\![b]\!] = (a^{(1)} + b^{(1)},\ \dots,\ a^{(k)} + b^{(k)};\ m(a)^{(1)} + m(b)^{(1)},\ \dots,\ m(a)^{(k)} + m(b)^{(k)};\ \kappa^{(1)},\ \dots,\ \kappa^{(k)}).$$

we verify that,

$$\begin{aligned} \sum_{i=1}^{k} a^{(i)} + b^{(i)} &= \sum_{i=1}^{k} a^{(i)} + \sum_{i=1}^{k} b^{(i)} \\ &= a + b \end{aligned}$$

and

$$\begin{aligned} \sum_{i=1}^{k} m(a)^{(i)} + m(b)^{(i)} &= \sum_{i=1}^{k} m(a)^{(i)} + \sum_{i=1}^{k} m(b)^{(i)} \\ &= m(a) + m(b) \\ &= \kappa \cdot a + \kappa \cdot b \\ &= \kappa \cdot (a + b) \\ &= m(a + b) \end{aligned}$$

thus $[\![a + b]\!] = [\![a]\!] + [\![b]\!]$.

- *Multiplication with one private input:* Given a private input $[\![a]\!]$ and a public one λ, their product is defined by

$$\lambda \cdot [\![a]\!] = (\lambda \cdot a^{(1)}, \ \dots, \ \lambda \cdot a^{(k)}; \ \lambda \cdot m(a)^{(1)}, \dots, \\ \lambda \cdot m(a)^{(k)}; \ \kappa^{(1)}, \ \dots, \ \kappa^{(k)})$$

We verify that,

$$\begin{aligned} \lambda \cdot a^{(1)} + \ \dots + \ \lambda \cdot a^{(k)} &= \lambda \cdot \sum_{i=1}^{n} a^{(i)} \\ &= \lambda \cdot a \end{aligned}$$

and

$$\begin{aligned} \lambda \cdot m(a)^{(1)} + \dots + \ \lambda \cdot m(a)^{(k)} &= \lambda \cdot \sum_{i=1}^{n} m(a)^{(i)} \\ &= \lambda \cdot m(a) \\ &= m(\lambda \cdot a) \end{aligned}$$

Thus $\lambda \cdot [\![a]\!] = [\![\lambda a]\!]$.

- *Multiplication of two private inputs:* To multiply two private inputs $[\![x]\!]$ and $[\![y]\!]$, given a triple $([\![a]\!], [\![b]\!], [\![c]\!])$ from the pre-processing phase, the product is defined by

$$[\![x]\!] \cdot [\![y]\!] = [\![c]\!] + \alpha [\![b]\!] + \beta [\![a]\!] + \alpha \cdot \beta.$$

where $\alpha = [\![x]\!] - [\![a]\!]$ and $\beta = [\![y]\!] - [\![b]\!]$.
It is clear from the three operations verified above that,

$$\alpha = [\![x]\!] - [\![a]\!] = [\![x - a]\!]$$

and

$$\beta = [\![y]\!] - [\![b]\!] = [\![y - b]\!]$$

thus,

$$\begin{aligned} [\![x]\!] \cdot [\![y]\!] &= [\![c]\!] + \alpha [\![b]\!] + \beta [\![a]\!] + \alpha \cdot \beta \\ &= [\![c]\!] + [\![\alpha \cdot b]\!] + [\![\beta \cdot a]\!] + \alpha \cdot \beta \\ &= [\![c + \alpha \cdot b + \beta \cdot a + \alpha \cdot \beta]\!] \\ &= [\![a \cdot b + (x - a) \cdot b + \beta \cdot a + \alpha \cdot \beta]\!] \\ &= [\![x \cdot b + \beta \cdot a + \alpha \cdot \beta]\!] \\ &= [\![x \cdot b + (y - b) \cdot a + (x - a) \cdot (y - b)]\!] \\ &= [\![x \cdot b + x \cdot (y - b)]\!] \\ &= x \cdot y. \end{aligned}$$

B. PRIVACY

Suppose we have an adversary $\mathcal{A}$, a coalition of dishonest parties $\mathcal{A} = \{a_{i_1}, \dots, a_{i_m}\}$ from the group of participating in the protocol $\{a_1, \dots, a_k\}$ and $\mathcal{A}$ is seeking to uncover the private input x_i of the honest party a_i. During the execution of the protocol till the end, every party $a_j \in \mathcal{A}$ gets from a_i the following information: $\{x_i^{(j)}, C_i^{(j)}, y^{(i)}, m(y)^{(i)} - \kappa^{(i)} y\}$ where $C_i^{(j)} = x_i \cdot \mathrm{Enc}_{pk_j}(\kappa^{(j)}) - \mathrm{Enc}_{pk_j}(e_i^{(j)})$ and $y = Rep(x_1, \dots, x_k)$. A question arises at this level. What can $a_j \in \mathcal{A}$ learn from this information either individually or as a coalition? It can of course decrypt $C_i^{(j)}$ to get $m(x_i)^{(j)} = x_i \cdot \kappa^{(j)} - e_i^{(j)}$. At this stage it is useful to highlight that $x_i^{(j)}$ and $e_i^{(j)}$ are uniform random numbers from $\mathbb{F}$ that a_i has generated independently from x_i. In addition $e_i^{(j)}$ are unknown but to $\{a_j\}_{j \neq i}$.

Therefore we can say that:

It is not feasible to reconstruct x_i from less than the k shares $\{x_i^{(j)}\}_{1 \leq j \leq k}$ which follows from the fact that the secret sharing scheme is an k out of k scheme.

The adversary cannot recover x_i from $m(x_i)^{(j)} = x_i \cdot \kappa^{(j)} - e_i^{(j)}$ despite the fact that he knows $\kappa^{(j)}$ because $e_i^{(j)}$ is unknown to him and it is blinding $x_i \cdot \kappa^{(j)}$. The probability for him to make the right guess is small $\simeq 1/p$.

There is a case where only one party of the k parties is honest. Let us say it is a_1. In this case the adversary $\mathcal{A} = \{a_2, \dots, a_k\}$ can deduce x_1 just from his known ratings $\{x_2, \dots, x_k\}$ and the outcome of the computation $Rep(x_1, \dots, x_k)$.

Therefore we can state:

Theorem 2: The Protocol grants feedback privacy in the presence of at least two honest parties.

C. AUTHORIZABILITY

We enforce authorizability by requiring a valid receipt from the rater, proving that an interaction occurred between him and the ratee. We denoted it *Trace*. As mentioned above, it contains interaction-specific data, a timestamp, and signatures. The rater includes *Trace* in the JOIN transaction before he can join the list of source parties. As a result, a party that has not interacted with t cannot join its list or rate it. Furthermore, it cannot use the same receipt multiple times because it is registered on the blockchain the first time it does. An advantage of the blockchain property of preventing double-spending.

D. VERIFIABILITY

Verifiability is also a crucial property in a blockchain. It allows any user to get the full copy and follow from the genesis block to the last one, verifying that every transaction is executed correctly and reflected in accounts. Another fact is that each new block is invalid unless the network majority (consensus) verifies and accepts it. This possibility remains available, particularly for ratees and newcomers to the network.

E. AVAILIBILITY

The blockchain relies on a P2P network. If a peer fails, it does not affect others as the data is duplicated over the system, and all validators have complete copies of it. On the other hand, regular nodes maintain short copies comprising headers and the state that contains the comprehensive list of accounts and reputation scores.

F. CONSISTENCY

Every new block validator must broadcast it to the community for verification and acceptance. If the majority accepts it, then it is considered valid. As a result, every party has the latest copy of transactions and reputation information, observing a propagation time.

G. CONSERVATION

The property of conservation in the system is obtained straightforwardly from a blockchain characteristic. It relies on the property of immutability and the fact that all transactions on valid blocks cannot be altered or deleted, including rating transactions.

H. WHICH PARAMETER k FOR THE SYSTEM

Running secure multiparty computation simultaneously with all the n source parties is demanding, especially if n is a large number. Even more, in dynamic networks, where parties can enter and leave for diverse reasons. Besides, the task requires $O(n^2)$ message, which we reduce significantly by dividing the source parties into subgroups that do not exceed a fixed number of parties k with k a fixed parameter overall in the system.

From the discussion above in theorem 2, we know that privacy holds if there are at least two honest parties among the k parties in each subgroup. The probability to have, at least two honest parties participating in the subgroup is equal to $1-[(n-b)\binom{b}{k-1}+\binom{b}{k}]/\binom{n}{k}$ if the parties are selected in a uniform random way. k is the number of participating parties, n is the total number of source parties, and b is the total number of corrupt parties. Let $Pr(k)$ be this probability according to the value of k:

$$Pr(k)=1-\frac{(n-b)\binom{b}{k-1}+\binom{b}{k}}{\binom{n}{k}}$$

In the following, we take the number of corrupt parties b as a percentage of the total number of source parties n. Figure 5 shows how $Pr(k)$ behaves according to k values with b set to different percentages of n. In table 6 we vary b from 10% to 70% of the total number of source parties n. We set some target values for $Pr(k)$ to achieve and determine the thresholds of k that ensure the probability $Pr(k)$ is higher than the desired values:

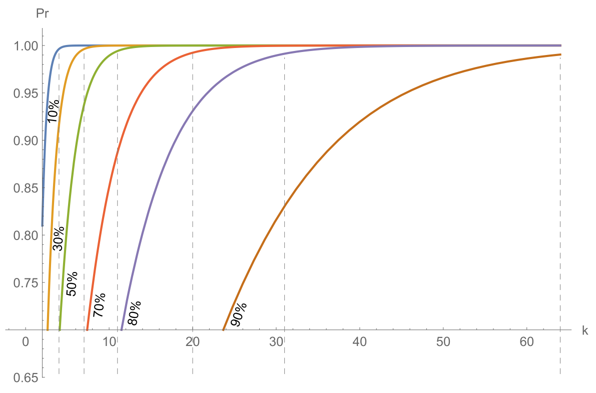

FIGURE 5. The probability $Pr(k)$ for different percentages of corrupt parties according to k.

VIII. PERFORMANCE EVALUATION

This section evaluates our system to validate its effectiveness in two ways. The first way estimates the system complexity component by component and provides its overall complexity (see Table 7).

TABLE 6. Minimal values of k for desired $Pr(k)$ according to b.

b	10%	20%	30%	40%	50%	60%	70%
$Pr(k)\geq$							
0.9	3	4	4	5	7	9	12
0.99	4	5	7	8	11	14	20
0.999	5	7	9	11	14	19	27
0.9999	6	8	11	14	18	24	34
0.99999	7	10	13	17	22	29	41
0.999999	8	11	15	19	25	34	48

The second way simulates the protocol and measures its performance. Several simulations were conducted to assess the system in different scenarios by varying parameters to demonstrate its strengths and weaknesses. All simulations were executed on an Intel(R) Core(TM) i7-8750H CPU laptop and repeated over 100 sessions. The results in this paper retain the average of the 100 sessions.

To simulate the blockchain, we used Ganache [82], an Ethereum simulator that enables application development on Ethereum (called Smart Contracts). Ganache includes all remote procedure call (RPC) functions and features and can be used programmatically via Node.js [83]. We implemented our on-chain logic using two Smart Contracts written in Solidity language [84], an object-oriented programming language used on various blockchain platforms, notably Ethereum. Then, we deployed the Smart Contracts to Ganache via Web3.js, the Ethereum JavaScript API [85], which enables interaction with Ethereum nodes using interprocess communication (IPC).

For the off-chain phase written in JavaScript, we used Web3.Utils.randomHex for generating randomness, ECDSA for authentication, and ECIES for encryption. We highlight that the JavaScript code does not rely on HTTP or Websockets. It is instead run on the Node.js runtime environment as a stand-alone application. The parameters and settings information for our simulation are shown in Table 8.

Firstly, by simulating the system, we verify that the produced reputation scores match the ratings' weighted average. This way, we demonstrate that the protocol is sound and outputs accurate results. Then, we study the effect of varying parameters like subgroups cardinality k, shares bitlength, and Paillier modulo on performance.

Experiment 1: We know from subsection VII-H that the minimal value of parameter k required to maintain targeted privacy is proportional to the ratio of corrupt parties in the system. Hence, to study the impact of increasing malicious parties' ratio, it is sufficient to raise k and monitor the execution time. The experiment's results illustrated in Fig. 6, 7, 8, and 9 show a running time quasi-linear in k.

Experiment 2: The second parameter studied was the shares bitlength, which depends solely on the finite field $\mathbb{F}_p$ prime number p. The shares are primarily generated randomly in $\mathbb{F}_p$, and they have the same bitlength as the parameter p regardless of the rating domain. The experiment consists of changing shares' bitlength by changing p accordingly and

TABLE 7. System complexity estimation.

	Communication	*Encryptions*	*Random Generation*	*Hashing*	*Additions*	*Multiplications*	*Decryptions*
$\Pi_{[\![.]\!]}$.Initialize	(k-1)n	(3k-3)n	n	(2k-2)n	(2k-2)n	(2k-2)n	0
$\Pi_{[\![.]\!]}$.Input	(k-1)n	(k-1)n	(2k-2)n	0	(3k-3)n	(k-1)n	(k-1)n
$\Pi_{[\![.]\!]}$.Open	(k-1)n	0	0	0	(k-1)n	0	0
$\Pi_{[\![.]\!]}$.Check	(k-1)n	0	n	kn	(k+1)n	n	0
Total	(4k-4)n	(4k-4)n	2kn	(3k-2)n	(7k-5)n	(3k-2)n	(k-1)n
Complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

TABLE 8. Simulation parameters.

Parameters	*Values*
Reputation function	weighted average
Total number of nodes	500
Subgroups cardinality	3-100
Rating range	0-100
Shares bitlength	64, 128, 256 bits
ECDSA security	256 bits
ECIES security	256 bits

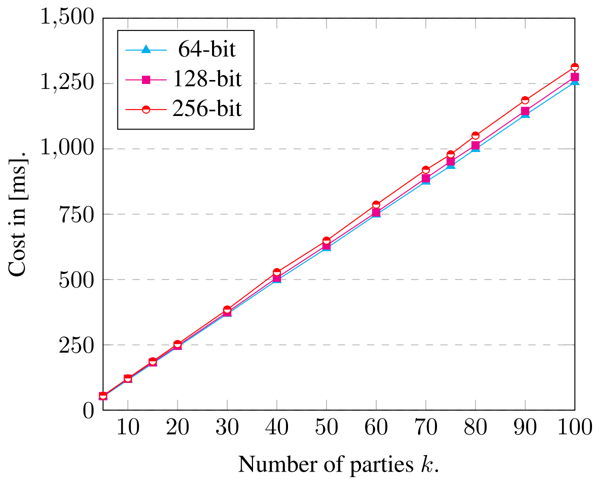

FIGURE 6. $\Pi_{[\![.]\!]}$.Input cost/user according to shares bit length and k.

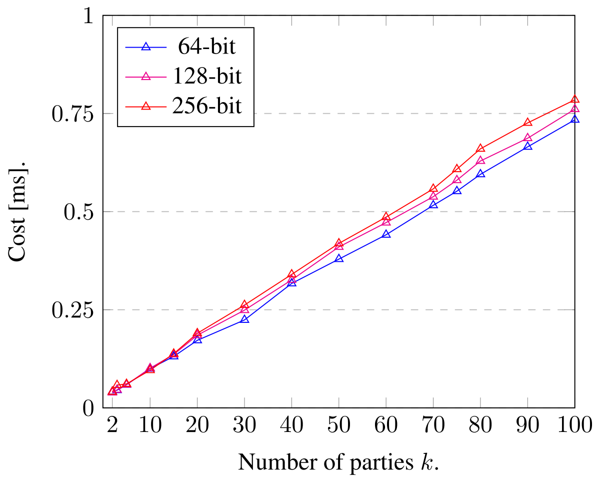

FIGURE 7. $[\![.]\!]$.Compute cost/user according to shares bit length and k.

TABLE 9. Computation cost/user according to Paillier modulo bit length.

Paillier Public Modulo Bit Length	*2048*	*3072*	*4096*
Π_{NIZKPoK}	19ms	61ms	139ms
$\Pi_{[\![.]\!]}$.Initialize	26ms	82ms	187ms
$\Pi_{[\![.]\!]}$.Check	0.021ms	0.021ms	0.021ms

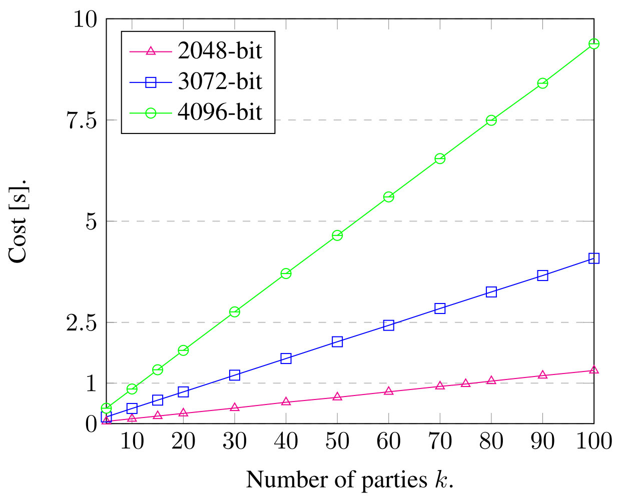

FIGURE 8. $\Pi_{[\![.]\!]}$.Input cost/user according to Paillier key length and k.

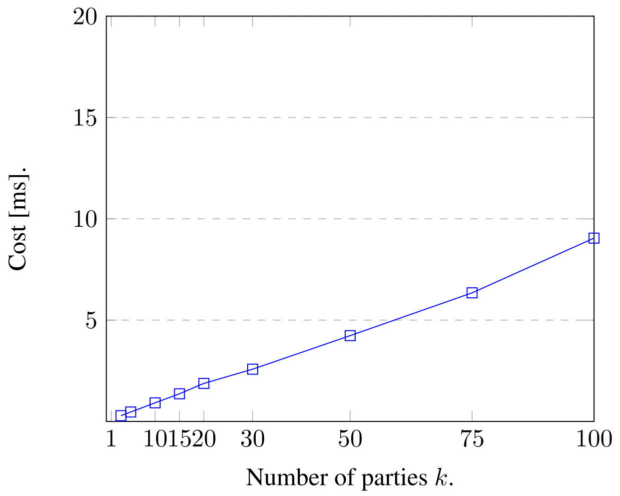

FIGURE 9. Phase 3: Rate transaction cost for the miner according to k.

checking any effect on the system performance/ execution time. Namely, we set p's bitlength to 64, 128, and 256 bits, as shown in Figures 6 and 7. The results reveal a minimal effect on performance.

Experiment 3: Similarly, we look into the security of the underlying SHE scheme and its impact on performance. Specifically, we target different levels of security by setting the Paillier modulo to 2048, 3072, and 4096 bits. Then we evaluate the differences in performance between the three values. The results specific to the affected phases are illustrated in Figure 8 and Table 9. They show how consequent the impact of changing the security parameter on performance. Although, the results remain reasonable even with 4096-bit security (e.g., under 2.5s for k=25).

IX. CONCLUSION

So far, we have realized a new dynamic, decentralized, and privacy-preserving reputation system. The proposed approach uses blockchain to store and update reputation information in a distributed manner and SMC to ensure feedback privacy. We got a system with better accessibility that preserves reputation information from loss and delivers a complete version of it. Moreover, it is secure in the malicious adversarial model, even with a dishonest majority, which is strong security. Using such techniques, we got a reliable system that is not affected by parties leaving the network.

In future works, we would like to address the issue of privacy alongside other challenges that reputation systems encounter. These challenges include self-promotion, slandering, whitewashing, and oscillation. To this end, we consider exploring advanced cryptographic tools such as data obfuscation, functional encryption, and homomorphic secret-sharing.

Another field of interest for prospective works is decentralized reputation systems for location-sensitive networks such as MANETs and VANETs. In such networks, feedback is relayed by immediate neighbors that are present in the same area as the sender party, and naturally, they know its location. The situation being a clear violation of the sender's privacy, the challenge is to create a protocol that allows parties to provide feedback while remaining anonymous.

Besides, for decentralized systems relying on the blockchain, the issue of blockchain scalability requires special attention. Indeed, blockchain has limitations on the number of processed transactions, and we need to apply additional techniques such as Sharding or Nested Blockchain to overcome the problem and produce a compelling solution.

APPENDIX
PRE-PROCESSING PHASE

Protocol 10: Π_{Triple}: Protocol for random triple generation

- **Multiply:**
    1) Each party P_i samples $a^{(i)}, b^{(i)}, \hat{b}^{(i)} \xleftarrow{\$} \mathbb{F}$.
    2) Every unordered pair (P_i, P_j) executes the following:
        a) P_i uses Π_{NIZKPoK} to send P_j the encryption $\text{Enc}_{\text{pk}_i}(a^{(i)})$.
        b) P_j computes $C^{(ij)}{=}b^{(j)}.\,\text{Enc}_{\text{pk}_i}(a^{(i)}){-}\text{Enc}_{\text{pk}_i}(e^{(ij)})$ for random $e^{(ij)} \xleftarrow{\$} \mathbb{F}$ and sends it to P_i.
        c) P_i decrypts $d^{(ij)}{=}\text{Dec}_{\text{sk}_{ij}}(C^{(ij)})$.
        d) Repeat the last two steps with $\hat{b}^{(i)}$ to get $\hat{e}^{(ij)}$ and $\hat{d}^{(ij)}$.
    3) Each party P_i computes $c^{(i)}{=}a^{(i)}.\,b^{(i)} + \sum_{j\neq i}(e^{(ij)} + d^{(ij)})$ and $\hat{c}^{(i)}$ similarly.
- **Authenticate:**
    1) Each Party P_i calls $\Pi_{[\![\cdot]\!]}$.Input on $(a^{(i)}, b^{(i)}, \hat{b}^{(i)}, c^{(i)}, \hat{c}^{(i)})$ so that every party P_j gets its MAC shares $m(a^{(i)})^{(j)}$, $m(b^{(i)})^{(j)}$, $m(\hat{b}^{(i)})^{(j)}$, $m(c^{(i)})^{(j)}$, $m(\hat{c}^{(i)})^{(j)}$ respectively.
    2) Each party P_i sums its MAC shares of elements in $\{a^{(j)}\}_{j=1\ldots n}$ to get a MAC share in a: $m(a)^{(i)}{=}\sum_{j=1}^{n} m(a^{(j)})^{(i)}$. Similarly with $[\![b]\!]$, $[\![\hat{b}]\!]$, $[\![c]\!]$, and $[\![\hat{c}]\!]$.
- **Sacrifice:** The parties do the following:
    1) Call $r \leftarrow \Pi_{\text{Rand}}$.
    2) Set and store $[\![\rho]\!]{=}r \cdot [\![b]\!] - [\![\hat{b}]\!]$.
    3) Open $\rho \leftarrow \Pi_{[\![\cdot]\!]}$.Open $([\![\rho]\!])$.
    4) Call $\Pi_{[\![\cdot]\!]}$.Open$(\cdot)$ on $[\![\tau]\!] \leftarrow r \cdot [\![c]\!] - [\![\hat{c}]\!] - \rho \cdot [\![a]\!]$. If $\tau \neq 0$ then abort; else continue.
    5) Call $\Pi_{[\![\cdot]\!]}$.Check an all opened values. If any check fails, then abort; otherwise, continue the protocol.
- **Output:** $([\![a]\!], [\![b]\!], [\![c]\!])$ as a vector of valid triples.

REFERENCES

[1] F. Hendrikx, K. Bubendorfer, and R. Chard, "Reputation systems: A survey and taxonomy," *J. Parallel Distrib. Comput.*, vol. 75, pp. 184–197, Jan. 2015.
[2] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proc. 10th Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, Oct. 2001, pp. 310–317.
[3] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. 12th Int. Conf. World Wide Web (WWW)*, New York, NY, USA, 2003, pp. 640–651.
[4] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," in *Proc. 2nd ACM Workshop Secur. Ad Hoc Sensor Netw.*, New York, NY, USA: ACM Press, Oct. 2004, pp. 66–77.
[5] E. Pavlov, J. S. Rosenschein, and Z. Topol, "Supporting privacy in decentralized additive reputation systems," in *Trust Management* (Lecture Notes in Computer Science), C. Jensen, S. Poslad, and T. Dimitrakos, Eds. Berlin, Germany: Springer, 2004, pp. 108–119.
[6] O. Hasan, L. Brunie, and E. Bertino, "Preserving privacy of feedback providers in decentralized reputation systems," *Comput. Secur.*, vol. 31, no. 7, pp. 816–826, Oct. 2012.
[7] T. Dimitriou and A. Michalas, "Multi-party trust computation in decentralized environments," in *Proc. 5th Int. Conf. New Technol., Mobility Secur. (NTMS)*, May 2012, pp. 1–5.
[8] O. Hasan, L. Brunie, E. Bertino, and N. Shang, "A decentralized privacy preserving reputation protocol for the malicious adversarial model," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 6, pp. 949–962, Jun. 2013.
[9] M. R. Clark, K. Stewart, and K. M. Hopkinson, "Dynamic, privacy-preserving decentralized reputation systems," *IEEE Trans. Mobile Comput.*, vol. 16, no. 9, pp. 2506–2517, Sep. 2017.
[10] O. Goldreich, *Foundations of Cryptology*, vol. 1. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[11] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proc. 20th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 1988, pp. 103–112.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin, 2008, pp. 1–9, vol. 4, no. 2. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[13] M. Najafi, L. Khoukhi, and M. Lemercier, "Decentralized reputation model based on Bayes' theorem in vehicular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2021, pp. 1–6.

[14] C. Kugblenu and P. Vuorimaa, "Decentralized reputation system on a permissioned blockchain for e-commerce reviews," in *Proc. 17th Int. Conf. Inf. Technol. New Generations (ITNG)*, in (Advances in Intelligent Systems and Computing), S. Latifi, Ed. Cham, Switzerland: Springer, 2020, pp. 177–182.

[15] M. Li, L. Zhu, Z. Zhang, C. Lal, M. Conti, and M. Alazab, "Anonymous and verifiable reputation system for E-commerce platforms based on blockchain," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4434–4449, Dec. 2021.

[16] S. Qi, Y. Li, W. Wei, Q. Li, K. Qiao, and Y. Qi, "Truth: A blockchain-aided secure reputation system with genuine feedbacks," *IEEE Trans. Eng. Manag.*, early access, Feb. 24, 2022, doi: 10.1109/TEM.2021.3128930.

[17] M. Debe, K. Salah, M. H. Ur Rehman, and D. Svetinovic, "IoT public fog nodes reputation system: A decentralized solution using Ethereum blockchain," *IEEE Access*, vol. 7, pp. 178082–178093, 2019.

[18] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.

[19] S. A. M. Mirhosseini, A. Fanian, and T. A. Gulliver, "A trust and reputation system for IoT exploiting distributed ledger technology," 2021, *arXiv:2111.13500*.

[20] P. Weerapanpisit, S. Trilles, J. Huerta, and M. Painho, "A decentralized location-based reputation management system in the IoT using blockchain," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 15100–15115, Jan. 2022.

[21] M. A. Azad, S. Bag, and F. Hao, "PrivBox: Verifiable decentralized reputation system for online marketplaces," *Future Gener. Comput. Syst.*, vol. 89, pp. 44–57, Dec. 2018.

[22] S. Bag, M. A. Azad, and F. Hao, "A privacy-aware decentralized and personalized reputation system," *Comput. Secur.*, vol. 77, pp. 514–530, Aug. 2018.

[23] M. A. Azad, S. Bag, F. Hao, and A. Shalaginov, "Decentralized self-enforcing trust management system for social Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2690–2703, Apr. 2020.

[24] S. Lee and S.-H. Seo, "Design of a two layered blockchain-based reputation system in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 1209–1223, Feb. 2022.

[25] I. S. Ochôa, V. R. Q. Leithardt, L. Calbusch, J. F. De Paz Santana, W. D. Parreira, L. O. Seman, and C. A. Zeferino, "Performance and security evaluation on a blockchain architecture for license plate recognition systems," *Appl. Sci.*, vol. 11, no. 3, p. 1255, Jan. 2021.

[26] Y. I. Alzoubi, A. Al-Ahmad, and H. Kahtan, "Blockchain technology as a fog computing security and privacy solution: An overview," *Comput. Commun.*, vol. 182, pp. 129–152, Jan. 2022.

[27] D. G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, 2014, pp. 1–32, vol. 151, no. 2014.

[28] V. G. Martínez, L. H. Encinas, and C. S. Ávila, "A survey of the elliptic curve integrated encryption scheme," *J. Comput. Sci. Eng.*, vol. 2, no. 2, pp. 7–13, 2010.

[29] A. M. Antonopoulos, *Mastering Bitcoin: Programming the Open Blockchain*. Sebastopol, CA, USA: O'Reilly Media, Jun. 2017.

[30] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[31] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), E. Oswald and M. Fischlin, Eds. Berlin, Germany: Springer, 2015, pp. 281–310.

[32] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Dec. 2016.

[33] M. Peck. *The Bitcoin Arms Race Is On!—IEEE Spectrum*. [Online]. Available: https://spectrum.ieee.org/computing/networks/the-bitcoin-arms-race-is-on (May 2013).

[34] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," Self-Published Paper, Aug. 2012, pp. 1–6, vol. 19, no. 1.

[35] P. Vasin, "Blackcoin's proof-of-stake protocol v2," 2014, vol. 71, pp. 1–2. [Online]. Available: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf

[36] (2016). *Whitepaper:Nxt—Nxt Wiki*. [Online]. Available: https://nxtwiki.org/wiki/Whitepaper:Nxt

[37] K. Davarpanah, D. Kaufman, and O. Pubellier, "NeuCoin: The first secure, cost-efficient and decentralized cryptocurrency," 2015, pp. 1–38, *arXiv:1503.07768*.

[38] G. M. Nikolopoulos and M. Fischlin, "Information-theoretically secure data origin authentication with quantum and classical resources," *Cryptography*, vol. 4, no. 4, p. 31, Nov. 2020.

[39] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias, "Semi-homomorphic encryption and multiparty computation," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), K. G. Paterson, Ed. Berlin, Germany: Springer, 2011, pp. 169–188.

[40] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), J. Stern, Ed. Berlin, Germany: Springer, 1999, pp. 223–238.

[41] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, p. 13, 2014.

[42] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, p. 34, 2009.

[43] R. Bendlin and I. Damgård, "Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems," in *Theory Cryptography* (Lecture Notes in Computer Science), D. Micciancio, Ed. Berlin, Germany: Springer, 2010, pp. 201–218.

[44] V. Lyubashevsky, "Public-key cryptographic primitives provably as secure as subset sum," *IACR Cryptol. ePrint Arch.*, vol. 2009, p. 576, Feb. 2009.

[45] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge, U.K.: Cambridge Univ. Press, Jan. 2007.

[46] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), A. M. Odlyzko, Ed. Berlin, Germany: Springer, 1987, pp. 186–194.

[47] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), R. Safavi-Naini and R. Canetti, Eds. Berlin, Germany: Springer, 2012, pp. 643–662.

[48] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure MPC for dishonest majority—Or: Breaking the SPDZ limits," in *Computer Security—ESORICS* (Lecture Notes in Computer Science), J. Crampton, S. Jajodia, and K. Mayes, Eds. Berlin, Germany: Springer, 2013, pp. 1–18.

[49] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making SPDZ great again," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), J. B. Nielsen and V. Rijmen, Eds. Cham, Switzerland: Springer, 2018, pp. 158–189.

[50] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1575–1590.

[51] P. Rindal and M. Rosulek, "Faster malicious 2-party secure computation with online/offline dual execution," in *Proc. 25th USENIX Conf. Secur. Symp.*, Berkeley, CA, USA: USENIX Association, Aug. 2016, pp. 297–314.

[52] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 21–37.

[53] S. Gueron, Y. Lindell, A. Nof, and B. Pinkas, "Fast garbling of circuits under standard assumptions," *J. Cryptol.*, vol. 31, no. 3, pp. 798–844, Jul. 2018.

[54] M. Keller, E. Orsini, and P. Scholl, "MASCOT: Faster malicious arithmetic secure computation with oblivious transfer," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 830–842.

[55] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), J. Feigenbaum, Ed. Berlin, Germany: Springer, 1992, pp. 420–432.

[56] M. Kinateder and K. Rothermel, "Architecture and algorithms for a distributed reputation system," in *Proc. 1st Int. Conf. Trust Manage.*, Berlin, Germany: Springer-Verlag, May 2003, pp. 1–16.

[57] J. Sabater and C. Sierra, "Review on computational trust and reputation models," *Artif. Intell. Rev.*, vol. 24, no. 1, pp. 33–60, Sep. 2005.

[58] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Comput. Surveys*, vol. 42, no. 1, pp. 1–31, Dec. 2009.

[59] W. Conner, A. Iyengar, T. Mikalsen, I. Rouvellou, and K. Nahrstedt, "A trust management framework for service-oriented environments," in *Proc. 18th Int. Conf. World Wide Web*, New York, NY, USA, Apr. 2009, pp. 891–900.
[60] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, Mar. 2007.
[61] Y. Yao, S. Ruohomaa, and F. Xu, "Addressing common vulnerabilities of reputation systems for electronic commerce," *J. Theor. Appl. Electron. Commerce Res.*, vol. 7, no. 1, pp. 1–20, Apr. 2012.
[62] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, Jan. 2000, p. 9.
[63] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system," in *The Economics of the Internet and E-Commerce*, M. R. Baye Ed. Amsterdam, The Netherlands: Elsevier Science, 2002, pp. 127–157.
[64] T. Dimitriou, G. Karame, and I. Christou, "SuperTrust: A secure and efficient framework for handling trust in super peer networks," in *Distributed Computing and Networking* (Lecture Notes in Computer Science), S. Rao, M. Chatterjee, P. Jayanti, C. S. R. Murthy, and S. K. Saha, Eds. Berlin, Germany: Springer, 2008, pp. 350–362.
[65] M. Srivatsa and L. Liu, "Securing decentralized reputation management using TrustGuard," *J. Parallel Distrib. Comput.*, vol. 66, no. 9, pp. 1217–1232, 2006.
[66] Z. Liang and W. Shi, "PET: A PErsonalized Trust model with reputation and risk evaluation for P2P resource sharing," in *Proc. 38th Annu. Hawaii Int. Conf. Syst. Sci.*, 2005, p. 201.
[67] L. Xiong and L. Liu, "PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 7, pp. 843–857, Jul. 2004.
[68] A. Jøsang and R. Ismail, "The beta reputation system," in *Proc. 15th Bled Conf. Electron. Commerce*, Jan. 2002, pp. 2502–2511.
[69] J. Patel, W. T. L. Teacy, N. R. Jennings, and M. Luck, "A probabilistic trust model for handling inaccurate reputation sources," in *Trust Management* (Lecture Notes in Computer Science), P. Herrmann, V. Issarny, and S. Shiu, Eds. Berlin, Germany: Springer, 2005, pp. 193–209.
[70] S. Ruohomaa, L. Kutvonen, and E. Koutrouli, "Reputation management survey," in *Proc. 2nd Int. Conf. Availability, Rel. Secur. (ARES)*, Apr. 2007, pp. 103–111.
[71] E. Koutrouli and A. Tsalgatidou, "Reputation-based trust systems for P2P applications: Design issues and comparison framework," in *Trust and Privacy in Digital Business* (Lecture Notes in Computer Science), S. Fischer-Hübner, S. Furnell, and C. Lambrinoudakis, Eds. Berlin, Germany: Springer, 2006, pp. 152–161.
[72] A. Jøsang, X. Luo, and X. Chen, "Continuous ratings in discrete Bayesian reputation systems," in *Trust Management II* (IFIP The International Federation for Information Processing), Y. Karabulut, J. Mitchell, P. Herrmann, and C. D. Jensen, Eds. Boston, MA, USA: Springer, 2008, pp. 151–166.
[73] S. Buchegger and J.-Y. Le Boudec, "A robust reputation system for P2P and mobile ad-hoc networks," in *Proc. P2P Mobile Ad-Hoc Netw., 2nd Workshop Econ. Peer-to-Peer Syst.*, Jun. 2004, pp. 1–6.
[74] S. Song, K. Hwang, R. Zhou, and Y.-K. Kwok, "Trusted P2P transactions with fuzzy reputation aggregation," *IEEE Internet Comput.*, vol. 9, no. 6, pp. 24–34, Nov. 2005.
[75] R. Aringhieri, E. Damiani, S. De Capitani Di Vimercati, S. Paraboschi, and P. Samarati, "Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 57, no. 4, pp. 528–537, 2006.
[76] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," in *Proc. 1st Int. Joint Conf. Auto. Agents Multiagent Syst.*, New York, NY, USA: Association for Computing Machinery, Jul. 2002, pp. 475–482.
[77] R. Nithyanand and K. Raman, "Fuzzy privacy preserving peer-to-peer reputation management," Cryptol. ePrint Arch., 2009, pp. 1–10. [Online]. Available: https://eprint.iacr.org/2009/442
[78] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative peer groups in Nice," in *Proc. IEEE INFOCOM. 22nd Annu. Joint Conf. IEEE Comput. Commun. Societies*, Mar. 2003, pp. 1272–1282.
[79] R. Zhou and K. Hwang, "PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 460–473, Apr. 2007.
[80] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing," in *Proc. 3rd Conf. Networked Syst. Design Implement.*, vol. 3, Berkeley, CA, USA: USENIX Association, May 2006, p. 1.
[81] A. Whitby, A. Jøsang, and J. Indulska, "Filtering out unfair ratings in Bayesian reputation systems," in *Proc. 7th Int. Workshop Trust Agent Soc.*, 2004, pp. 106–117.
[82] (2022). *Ganache|Overview—Truffle Suite*. Accessed: Nov. 4, 2022. [Online]. Available: https://trufflesuite.com/docs/ganache/
[83] *Node.js*. Accessed: Nov. 4, 2022. [Online]. Available: https://nodejs.org/en/
[84] *Solidity—Solidity 0.8.17 Documentation*. Accessed: Nov. 4, 2022. [Online]. Available: https://docs.soliditylang.org/en/v0.8.17/
[85] *Web3.js—Ethereum JavaScript API—Web3.js 1.8.0 Documentation*. [Online]. Available: https://web3js.readthedocs.io/en/v1.8.0/

KHALID MRABET received the M.S. degree in mathematics, computer science, and their applications from the Faculty of Science, Mohamed First University, Oujda, Morocco, in 2009. He is currently pursuing the Ph.D. degree with the ENSIAS College of Engineering, Mohammed V University, Rabat. His main research interests include decentralized networks, reputation and trust, cryptography, privacy, and security.

FAISSAL EL BOUANANI (Senior Member, IEEE) was born in Nador, Morocco, in 1974. He received the M.S. and Ph.D. degrees in network and communication engineering from Mohammed V University–Souissi, Rabat, Morocco, in 2004 and 2009, respectively. He was a Faculty Member at the University of Moulay Ismail, Meknes, from 1997 to 2009. In 2009, he joined the National High School of IT/ENSIAS College of Engineering, Mohammed V University, Rabat, where he is currently an Associate Professor. He advised many Ph.D. and master's students at Mohammed V University and the University of Moulay Ismail. So far, his research efforts have culminated in more than 90 papers in a wide variety of IEEE/ACM international conferences and journals. His current research interests include performance analysis and the design of wireless communication systems. He has been involved as a TPC member in various conferences and IEEE journals. His Ph.D. Thesis was awarded the best one by Mohammed V University–Souissi, in 2010. He served as the TPC Chair for the ICSDE conferences, the General Co-Chair for ACOSIS 2016 and CommNet 2018 conferences, and the General Chair for the 2019/2020/2021 CommNet conferences. He is an Associate Editor of IEEE Access and *Frontiers in Communications and Networks* journal. He also serves as a Lead Guest Editor for the Physical Layer Security Special Issue-*Frontiers in Communications and Networks* journal.

HUSSAIN BEN-AZZA received the Ph.D. degree in mathematics and computer science from Claude Bernard University Lyon 1, France, in 1995. He is currently a Professor with the ENSAM National High School of Arts and Trades, Meknes, Morocco, attached to the Department of Industrial and Production Engineering, Moulay Ismail University. His research interests include coding theory, cryptography, and wireless communications, but also applications of optimization techniques to industrial engineering.