Reputation at Stake! A Trust Layer over Decentralized Ledger for Multiparty Computation and Reputation-Fair Lottery

Mario Larangeira

Department of Mathematical and Computing Science
School of Computing
Tokyo Institute of Technology
mario@c.titech.ac.jp
IOHK
mario.larangeira@iohk.io

Abstract. This work leverages on the framework of Karakostas *et al.* (SCN'20) by extending it to the realm of reputation and trust. At the best of our knowledge, it is the first to introduce reputation and trust to proof of stake systems. Namely, we show that their delegation framework can be repurposed to construct a trust layer over a proof of stake consensus protocol in addition to its original stake delegation application. Furthermore, we show that such extension yields a concrete reputation system satisfying the positive results of (1) Asharov *et al.* (Asiacrypt'13), therefore allowing the secure execution of multiparty protocols such as GMW (STOC' 87) and Damgard and Ishai (Crypto'05), and (2) Kleinrock *et al.* (Indocrypt'20), therefore allowing the construction of Reputation-fair Lottery and therefore Proof of Reputation. More concretely, our devised layer is used to construct a concrete reputation system based on arbitrary stake distribution. In this layer groups of users can freely "assign their respective trust" to members of a set of trustees, *i.e.* participants that offered themselves as receivers of such assignment. Furthermore, our work offers the advantage of providing a clear stake based criteria, verifiable in the ledger, and, therefore, naturally resistant to sybil attack, that the set of trustees indeed yields an honest majority. This setting provides a better situation than a simple assumption of honest majority, since it involves stake in a decentralized ledger, and the public verifiability of the reputation score via verification of the stake distribution.

Keywords: Ranking, Reputation, Trust, Proof-of-Stake, MPC.

1 Introduction

Trust between individuals and reputation is ubiquitous to life in society. Human beings are instinctively designed to take it into account almost in all decisions that we do in our lives. Surprisingly, despite its fundamental role and numerous works in the literature, it seems there is still not a thorough solution that satisfactorily captures it in every detail. A closer look into a more practical situation may shed some light into the reason why that is the case.

Reputation in Everyday Life. As a practical effect of reputation, or lack thereof, George Arklof coined the term "Market for Lemons" [1] for the situation when only low-quality sellers are in the market, after the high-quality sellers left, due to the inability of the buyers to tell the difference between them, *i.e.* no reputation system available. More concretely, high-quality sellers cannot keep themselves in business, without buyers taking into account the reputation of the seller, instead of only the price tag, when deciding for a purchase. An even more concrete example, which may have crossed the reader's mind is Amazon ranking system [24] or Google Pagerank algorithm [23], which are designed to harvest feedback from the users pool and promote good sellers or relevant webpages to the end user. However, even these widely popular systems have limitations (as we discuss later, in particular on ranking based approach) and are prone to manipulation and fraud, requiring extra measures from its maintainers in order to provide a minimal functional system.

Technical Challenges. Although the straightforward practicality and deceivingly simplicity of the concept, reputation record has shown itself to be somewhat elusive to formal techniques intended to satisfactorily capture this concept, and properly embed it in a practical system free from manipulation and general enough to be used widely. It is natural to expect that such systems would eventually be securely incorporated into our lives via information systems that are increasingly permeating all aspects of our daily routine. For the moment, such general system does not seem to exist. The explanation seems to rely on the technical challenges in pursuing such solution, and they can be broken down in three main areas in the available literature:

- eliciting feedback: Systems based on ranking crucially depend on the feedback of the system users. Briefly, a reputation framework needs to continuously gather and process the "opinion" from the parties;
- aggregating/displaying: The feedback from the user has to be aggregated in order to establish a comprehensible and meaningful value;
- distribution: Each player of such system is subject to receive a "reputation" value, which others can consult or be used in decision making, in particular, it can be used in outlining general strategies.

We provide a more thorough discussion on each of the items on Appendix A. Such systems, as the early mentioned [23,24], also have a shared limitation regarding the distribution of the "amount of" reputation or trust. That is, they are contained in their own silos, *e.g.* company databases. That is, they do not belong to a general platform, making it hard comparison even when they are offering similar services of products.

It is straightforward to understand that reputation system are particularly needed when parties have a long term expectation of activities, *i.e.* a set of nodes is expected to keep interacting with each other in the long term. In the early mentioned main areas, if the problems are solved it paves the way to correctly capture feedback, process and distribute it, which is valuable in guiding future interactions. More importantly, they can fundamentally change the interactions by allowing parties to establish their own strategies for further interactions. For example, a reliable and available reputation system could be used to guide participants of a protocol to whom they would choose to interact in order to perform a joint protocol.

1.1 Related Work

An interesting overlap between reputation and multiparty computation was observed by Asharov *et al.* [2]. In particular, [2] points out that by observing the reputation of the protocol participants, in some cases, it is possible to guarantee an honest majority. It is important to remark, that in the positive cases, a fair distributed computation can be performed and therefore such scenario is highly desirable in numerous scenarios.

This early work adds to the rich and long literature of reputation and trust management [17], even in decentralized form [11,15]. In the set of decentralized reputation, Kleinrock *et al.* [22] proposed a new paradigm named Proof of Reputation, following roughly on the analogous idea of Proof of Stake (PoS) and Proof of Work (PoW). That is, proportionally to the stake, or computational work, a participant would be more frequently selected to issue the new block. The authors of [22] propose, that the participants, proportionally to their own reputation, would be chosen in a reputation-fair lottery: the more reputable user, the more frequent it would be selected to issue a new block.

The work in [22] remarks that their framework is *consensus agnostic*, it would work on any paradigm, either PoS and PoW. Moreover, it would even offer a *Nakamoto Fallback*, *i.e.* the protocol can recover itself by relying on the underpinning consensus mechanism. Moreover, [22] introduces a new definition for *reputation-fair lottery* which is suitable for Proof of Reputation. Roughly, it states that given a vector of reputation scores for a set of participants, even the ones with the lowest reputation can still have a chance of being selected by the lottery. A definition which differs from another work on Proof of Reputation by Biryukov *et al.* [5], which focuses on assuring a more even probability distribution for the lottery algorithm.

1.2 Our Work

The starting point of our work comes from the observation that the delegation of stake, as defined in [18], can be interpreted as *an assignment of trust.* In other words, when a stakeholder in the PoS environment delegates its stake to a *stake pool*, it is in fact, informing publicly that it trusts that entity. Hence, these especial participants, or *trustees* as we later denote them, can publicly claim a higher reputation score by accumulating more votes, or assignments, of trust via a similar delegation method adapted from [18].

PoS based Reputation Vector. We show that the framework proposed in [18] is general enough to be repurposed to a different context, *i.e.*, a trust layer. More concretely, when a participant wants to receive trust assignments from others, it would issue a certificate and publish it in the ledger $\mathcal{L}$. Similarly to what is already done with *stake pools* in [18]. The difference here is that such *registration certificates* contains a *context identification*, *i.e.* a string $ID_{context}$, which represents the context that the trust assigners trust that *trustee*. The string $ID_{context}$ is a unique identifier which can represent a particular service, product, company or a decentralized identity [26]. Therefore, by assigning trust, again, via delegation procedure similarly to the technique in [18], to the published certificate, a participant is publicly informing that it trusts the signer of that certificate with respect to that context. Needless to say, the trust assigning certificate should also contain the identifier $ID_{context}$. Hence it does not interfere with the critical PoS delegated consensus protocol, and also to allow multiple contexts.

In comparison to [22], our proposed reputation system is limited to PoS systems given the adaptation from [18]. On the other hand, [22] lacks the definition of a concrete reputation system (similarly with [2]), despite of presenting concrete results, including a Nakamoto style fallback in case of flawed reputation ranking. Here, we concretely present a distributed reputation protocol by allowing the participants of the PoS system to freely, and publicly, show their trust, therefore allowing reputation to be created. Typically, the reputation of a set of participants, say $\mathcal{T}$, is a vector of (real) values between $[0, 1]$ of size $|\mathcal{T}|$, where the i-th position indicates the likelihood of the i-th player being honest. Hence reputation 1 means that the player is assured to behave honestly, or, depending on the context, has a perfectly good reputation.

In our model, given an unique context identifier, again, $ID_{context}$, participants can issue, and publish in the $\mathcal{L}$, certificates that tie together $ID_{context}$ and their respective verification keys. In this way, they are perceived by all the users of the PoS system as members of the pool of potential receivers of delegated trust, *e.g.* the set $\mathcal{T}$, in that particular *context*. Given that these certificates and verification keys are public, they can be widely advertised by their issuers outside of the protocol, *i.e.* in the real world. In this setting, any user of the entire PoS system, therefore a *stakeholder* which is a member of the set $\mathcal{U}$, such that $\mathcal{T} \subset \mathcal{U}$, can issue trust votes (delegation certificates in the jargon of [18]) also containing the context identifier $ID_{context}$. The reputation scores naturally arise by taking into account the stake associated with each trust assignment combined. For a fixed context identification string $ID_{context}$, each score of the reputation vector is the percentage of stake that each participant, in $\mathcal{T}$, was assigned with respect to the combination of all stake of the set $\mathcal{U}$.

Stake as Trust Weight. Assume, for example, that all participants in $\mathcal{U}$ are assigning trust with the identifier $ID_{context}$ for multiple trustees. Furthermore, assume also that each of them assigns trust for every player in $\mathcal{T}$. In this scenario, every reputation value in the vector is 1, since they received trust votes from every player in $\mathcal{U}$. Note that the participants can cast votes for different members of the $\mathcal{T}$ set at the same time, *i.e.* same context. The reputation percentage, *i.e.* score, is taken by considering how much a single address of $\mathcal{T}$ has harvest for its own, with respect to the summation of all the stake in the set $\mathcal{U}$, *i.e.* the players voting for that context. More concretely, the overall stake being used to vote is the summation of all $s_1, \ldots, s_{|\mathcal{U}|}$, the corresponding shares of the participants in $\mathcal{U}$, and the resulting reputation score for the i-th member of the

$\mathcal{T}$ is $r_i = \frac{s_{\mathcal{U}_\mathcal{T}}}{s_1+\cdots+s_\mathcal{U}}$, where $s_{\mathcal{U}_\mathcal{T}}$ is the combination of the stake of all assigners to the i-th trustee. Needless to say, that although the system bases the "trust delegation" in the same fashion of [18], *i.e.* via certificates, this sort of purposely tailored delegation does not affect the stake delegation crucially important for the PoS consensus protocol. Given the context identification, it can be handled in isolation of the consensus.

This reputation system is compatible with the setting considered in [22]. More formally, each reputation value r_i has *correlation free*, and it is associated with a binary random variable H_i which tells if the i-th participant behaves honestly, thus $\Pr[H_i = 1] = r_i$. Differently from [22,2], our framework relies by design on the community of users interested in the context referred by the $ID_{context}$. Hence, ultimately, the reputation scores of our system reflect how the set of trust assigners, the set $\mathcal{U}$, perceives the members of the set $\mathcal{T}$. In other words, this perception can be subject to real world information.

Relying on the stake of the underlying PoS ledger when computing the reputation score has the "good" side effect to guarantee some accountability of the feedback, arguably increasing the quality of the resulting vector. Furthermore, it provides a more dynamic environment where the reputation can be verified with access to the ledger $\mathcal{L}$. On the other hand, our construction does not support negative feedback, *i.e.* -1 or "bad" as mentioned earlier in the technical challenge of modeling reputation. Our proposed system offers only a simple "trust association" which is already enough for numerous applications. Despite of the limited functionality, regarding inclusion, our model supports easy addition of new participants, both as trust assigners and trustees, since any newcomer could generate a new certificate containing the context identifier, and, therefore, turning itself in target of trust assignment.

Our proposed system addresses another drawback of handling reputation information. By relying on the underlying ledger also helps on the reputation distribution since all the trust assignments are public. In comparison to current and more common ranking reputation systems, our constructions is not deterred to a particular silo as a company platform or database. It enjoys the advantages of distributed systems as long as the consensus security assumptions are valid, typically honest majority of the stakes in the PoS paradigm. At the same time, by adopting specific context identifiers, groups of participants, for example, of the same market, can compare its own reputations by issuing certificates and publishing them in $\mathcal{L}$, thus competing by the trust delegation.

Trust is Transitive. As already mentioned, our framework allows participants to assign trust freely and publicly in multiple contexts. However, our work does not advocate a notion of right reputation. In our setting the correct score reputation is as correct as the public perceives it. Moreover, in our setting, we consider that *trust is transitive.* In other words, if a participant assigns trust to another, it is assumed that the trustee will behave honestly. Our basic assumptions are:

– Honest participants assign trust only to honest participants;
– The majority of the combined stake of the assigners is honest.

We advocate these two basic assumptions seem reasonable because while the latter can be assumed from the basic properties of the typical PoS ledgers [3,10,20,21], the former can be assumed since the trust assignments could be revoked (similar to the regular delegation in [18]). Therefore, via information of the real world, is expected that once, for a given context, the participants are not willing to associate their stake to a trustee any longer, they could revoke it. Leaving only honest trustees with assigned trust. Despite of being an optimistic view, this puts the reputation system in a better situation of contextualizing the reputation vector, in comparison to [2,22] which do not provide extra information on how the scores are obtained.

Concrete Benefits of Stake based Reputation. The concrete benefits of our novel design is that we can revisit the works in [2,22] in the light of a stake based reputation system, and derive an alternative criteria, this time based on stake distribution, in order to obtain honest majority of players while executing multiparty computation and fair proof-of-reputation protocol. This brings immediate advantages, because the participants can verify if indeed the conditions are satisfied via stake distribution in $\mathcal{L}$. We summarize our contributions:

– extension of the framework from [18], by providing a functionality $\mathcal{F}_{\mathcal{T}}$ (arguing also the existence of a protocol that realizes it under Universal Composability Framework [6]) to allow assignment of trust among participants for multiple contexts;
– introduce a concrete reputation scheme $\mathsf{Rep}_{\mathbb{S}}^{m}$ based on $\mathcal{F}_{\mathcal{T}}$ and the stake distribution $\mathbb{S}$ of a set of participants $\mathcal{T}$ with size m, in a PoS ledger $\mathcal{L}$;
– revisit the work of [2] in the light of $\mathsf{Rep}_{\mathbb{S}}^{m}$, and show that the early mentioned basic assumptions yield an honest majority on the set $\mathcal{T}$ except with negligible probability, and therefore allows the secure execution of protocols [13] and [9];
– revisit the work of [22] in the light of $\mathsf{Rep}_{\mathbb{S}}^{m}$, showing that $\mathcal{T}$ yields an honest majority except with negligible probability, therefore $\mathcal{T}$ can be used to build a reputation-fair lottery algorithm, and consequently a proof-of-reputation system over $\mathcal{L}$.

A drawback of our work is that, in fact, the guarantees it provides in terms of honest majority relies heavily on the perception of the public. In other words, it relies on how much users assign their trust and that can mislead the fact that the trustee may not be *faithfully* honest. We argue that this is intrinsic to all reputation/trust systems. Furthermore, our framework allows a dynamic setting which once an assigned trustee is identified of being dishonest, users can revoke their trust assignment in a publicly verifiable fashion via the PoS global ledger.

2 Basic Definitions

The trust in our model is closely tied to the stake, which will translate into honest majority of users, therefore we need to review previous results for reputation vectors and basic lemmas as, for example, the Hoeffding Inequality used in [2]. Due to page limitation, we leave the review of the proof of stake ledger $\mathcal{L}$, as it is given by the Kachina framework [19], and the delegation framework $\mathcal{F}_{\text{CoreWallet}}$, as given by [18], to the Appendices B and C respectively.

Security with Reputation Vector. For completeness we briefly review the key definitions for secure computation with reputation vector introduced in [2] which suits better our purposes. These definitions rely heavily on the standard definition available on the literature [6]. Let us start by reviewing the running time of family of functionality and protocol. For a complete description we refer the reader to [2]. Moreover, in the next definitions, let PPT mean probabilistic polynomial-time with respect to the security parameter $\lambda \in \mathbb{N}$.

Definition 1. *Let $\mathcal{F} = \{f^m\}_{m \in \mathbb{N}}$ be an infinite family of functionalities, where f^m is an m-ary functionality. We say that $\mathcal{F}$ is a PPT family of functionalities if there exists a polynomial $p(\cdot)$ and a Turing machine M that on input λ and m outputs a circuit $C_{\lambda,m}$ in time at most $p(\lambda + m)$ such that for every input $x_1, \ldots, x_m$ it holds that $C_{\lambda,m}(x_1, \ldots, x_m) = f^{(m)}(1^\lambda, x_1, \ldots, x_m)$.*

Let the family of protocol π be defined analogously. That is, it is said to be **polynomial time** if the running of time of all parties are upper bounded by a polynomial on $\lambda + m$.

The next definition introduces the extra vector parameter $\mathbf{x} \in (\{0,1\}^*)^{m(\lambda)}$ corresponding to the information regarding the reputation of the m participants defined as the function $m : \mathbb{N} \to \mathbb{N}$, for a varying number of participants.

Definition 2. *Let $m : \mathbb{N} \to \mathbb{N}$ be a function. We say that protocol π $t(\cdot)$-securely computes the functionality $\mathcal{F} = \{f^{m(\lambda)}\}_{\lambda \in \mathbb{N}}$ with respect to $m(\cdot)$, if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$, such that for every PPT distinguisher D, there exist a negligible function $\mu(\cdot)$ such that for every $\lambda \in \mathbb{N}$, every $I \subseteq [m(\lambda)]$ with $|I| \leq t(m(\lambda))$, every reputation vector $\mathsf{x} \in (\{0,1\}^*)^{m(\lambda)}$ and $z \in \{0,1\}^*$, it holds that $\big| \Pr\left[D(\mathsf{IDEAL}_{\mathcal{F},\mathcal{S}(z),I}(\lambda, m, \mathsf{x})) = 1\right] - \Pr\left[D(\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda, m, \mathsf{x})) = 1\right] \big| \leq \mu(\lambda)$.*

Lastly, we review the definition with respect to a reputation system. Assume Rep is the reputation system which provides $\mathsf{r}^m = (r_1^m, \ldots, r_m^m)$. Furthermore, we denote by $I \leftarrow \mathsf{r}^m$ the subset $I \subseteq [m]$ of parties chosen probabilistically where every $i \in I$ with probability $1 - r_i^m$, and the probabilistic choice of I is given to the distinguisher.

Definition 3. *Let m, Rep, $\mathcal{F}$ and π be as earlier mentioned. We say π securely computes $\mathcal{F}$ with respect to $(m(\cdot), \mathsf{Rep})$, if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$, such that for every PPT distinguisher D, there exist a negligible function $\mu(\cdot)$ such that for every $\lambda \in \mathbb{N}$, every reputation vector $\mathsf{x} \in (\{0,1\}^*)^{m(\lambda)}$ and $z \in \{0,1\}^*$, it holds $\big| \Pr_{I \leftarrow \mathsf{r}^{m(\lambda)}}\left[D(\mathsf{IDEAL}_{\mathcal{F},\mathcal{S}(z),I}(\lambda, m, \mathsf{x})) = 1\right] - \Pr_{I \leftarrow \mathsf{r}^{m(\lambda)}}\left[D(\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda, m, \mathsf{x})) = 1\right] \big| \leq \mu(\lambda)$.*

Feasibility Reputation. Similarly to [2], we focus on the relation between honest majority of players performing a protocol, despite of stating security assumptions in terms of stake. Namely, a major difference from our work is that while we focus on the honest stake, whereas [2] focus on majority in terms of number of participants running the protocol. Although it is not immediately clear to establish the relation between stakes and number of players, for the moment we review Hoeffding Inequality [16], rather than the Chernoff bound, in order to relate to the summation on the individual reputation scores. Later, in Section 3, we formally clarify the relation between stakes and reputation by introducing a concrete reputation system. Concretely, given a family of reputations $\mathsf{Rep} = \{\mathsf{r}^{m(\lambda)}\}_{\lambda\in\mathbb{N}}$ for a number of participants $m = m(\lambda)$ being assigned trust from regular stake holders, Hoeffding Inequality allows to state the average of the reputation should be greater than $1/2 + \omega\left(\sqrt{\frac{\log m}{m}}\right)$, or, equivalently, that the number of honest parties is greater than $m/2 + \omega\left(\sqrt{m \cdot \log m}\right)$.

Lemma 1 (The Hoeffding Inequality). *Let $X_1, \ldots, X_m$ be independent random variables, each ranging over the (real) interval $[0, 1]$, and let $\mu = \frac{1}{m}$. Then let $E[\sum_{i=1}^{m} X_i]$ denote the expected value of the mean of these variables. Then $\Pr\left[\left|\frac{\sum_{i=1}^{m} X_i}{m} - \mu \geq k\right|\right] \leq 2 \cdot e^{-2k^2 \cdot m}$ for every $k > 0$.*

Likewise [2] and, as already outlined earlier, the random variables we consider only have the values 0 and 1, therefore we rely on a simpler version of the inequality. The following claim is proven in [2].

Claim. Let $m : \mathbb{N} \to \mathbb{N}$ be such that $O(\log m(\lambda)) = O(\log \lambda)$, let $\mathsf{Rep} = \{\mathsf{r}^{m(\lambda)}\}_{\lambda\in\mathbb{N}}$ be a family of reputation vectors, and let $m = m(\lambda)$. If it holds that $\sum_{i=1}^{m} r_i^m > \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m})$, then there exist a negligible function $\mu(\lambda)$ such that for every λ $\Pr_{I \leftarrow \mathsf{r}^m}\left[|I| \geq \left\lfloor \frac{m}{2} \right\rfloor\right] < \mu(\lambda)$.

3 Trust Layer over Proof of Stake Ledger

The basic setting we consider has two sets of participants, the regular ones $\mathcal{U}$ and the *trustees* $\mathcal{T}$, with varying sizes of, respectively, $n(\lambda)$ and $m(\lambda)$ for the trust *assigners* and the receivers of trust. Here we introduce a framework to deal with the assignment of trust, via the delegation of stake (as briefly reviewed in Appendix C). That is, each participant $u \in \mathcal{U}$ controls an address α, such that it contains an amount of stake $\mathbb{S}(\alpha)$ as it is registered in a decentralized ledger $\mathcal{L}$. Our proposed reputation system $\mathsf{Rep}_{\mathbb{S}}^m$ takes into account the stake of all the trust assigners, say $\mathbb{S}(\alpha_{u_1}), \ldots, \mathbb{S}(\alpha_{u_n})$, to generate the reputation vector as they access $\mathcal{F}_{\mathcal{T}}$, our Trust Assignment Functionality. Before we define the Stake Reputation System $\mathsf{Rep}_{\mathbb{S}}^m$, we introduce $\mathcal{F}_{\mathcal{T}}$.

Trust Assignment Framework. From now we adapt the delegation and addressing framework of [18], and repurpose it in order to capture the "trust assignment" feature. In particular, we extend the functionality $\mathcal{F}_{\text{CoreWallet}}$ into the Trust Assignment Functionality $\mathcal{F}_{\mathcal{T}}$ by adding two new interfaces *Assign Trust* and *Verify Trust* which handle "trust transactions" *ttx*. We remark that $\mathcal{F}_{\mathcal{T}}$ contains the interfaces of $\mathcal{F}_{\text{CoreWallet}}$, however we left it out of the next definition for simplicity of the description. Furthermore, each regular participants u have verification keys vks_u which is used in the trust assignment.

For simplicity, let $\mathbb{P} = \mathcal{U} \cup \mathcal{T}$, the next functionality keeps a list $\mathbb{P}$ of both regular participants and trustees. Moreover for each participant, it also keeps a list S of tuples (ttx, σ, f), respectively, a trust transaction string, a signature and the result of the signature verification. Lastly, for each participant P, it keeps lists K_P which contains the corresponding verification key vks.

Functionality $\mathcal{F}_{\mathcal{T}}$

Assign Trust: Upon receiving $(\textsc{Trust}, sid, ttx)$ from P, such that $ttx = (\mathsf{vks}, m)$ for a metadata string m, forward the message to $\mathcal{S}$. Upon receiving the message $(\textsc{Trusted}, sid, ttx, \sigma)$ from $\mathcal{S}$, if $\forall (ttx', \sigma', b') \in S : \sigma' \neq \sigma$, $(ttx, \sigma, 0) \notin S$, and $\mathsf{vks} \in K_P$, add $(ttx, \sigma, 1)$ to S and return $(\textsc{Trusted}, sid, ttx, \sigma)$ to P.
Verify Trust: Upon receiving $(\textsc{VerifyTrust}, sid, ttx, \sigma)$ from $P \in \mathbb{P}$, forward to $\mathcal{S}$ and wait for $(\textsc{VerifiedTrust}, sid, ttx, \sigma, \phi)$, with $ttx = (\mathsf{vks}, m)$. Then find P_{s}, such that $\mathsf{vks} \in K_{P_{\mathsf{s}}}$, and:

- if $(ttx, \sigma, 1) \in S$, set $f = 1$
- else if P_{s} is not corrupted and $(ttx, \sigma, 1) \notin S$, set $f = 0$ and insert $(ttx, \sigma, 0)$ to S
- else if exists an entry $(ttx, \sigma, f') \in S$, set $f = f'$
- else set $f = \phi$ and insert (ttx, σ, ϕ) to S.

Finally, return $(\textsc{VerifiedTrusted}, sid, ttx, \sigma, f)$ to P.

Fig. 1. The trust interface which extends the delegation framework of [18].

Trustee Registration Certificate. We approach it, as follows: any trustee interested in receiving "trust" from regular participants generates a *Registration Certificate* Σ_{reg} and publishes it in the ledger $\mathcal{L}$ in similar fashion as outlined in Section C for stake pools. This certificate introduces the address whose keys are controlled by the trustee, say $\alpha_{\mathcal{T}_i}$, and it contains the metadata which is instantiated with an arbitrary string $ID_{context}$ of the choice of $\mathcal{T}_i$, that is $\mathcal{T}_i$ sets $m \leftarrow ID_{context}$. This string works as a unique identifier of the context which $\mathcal{T}_i$ is expected to be assigned trust. In other words, $\mathcal{T}_i$ publishes the transaction $tx = (\alpha_{\mathcal{T}_i}, \Sigma_{reg})$, for $\Sigma_{reg} = ((\mathsf{vks}_{\mathcal{T}_i}, m), \sigma)$ where $\mathsf{vks}_{\mathcal{T}_i}$ is verification key for $\mathcal{T}_i$. Other participants can issue similar certificates, and they also would be potential receivers of "trust" of regular participants.

Publicly Assigning Trust. The assignment is done via the issuing of "trust transactions" ttx, and then publishes a corresponding certificate in the ledger. A trust assigning certificate, is similar to the delegation certificate from Section C, it is a tuple $d = (\mathsf{vks}_{\mathcal{U}_j}, \langle \mathsf{vks}_{\mathcal{T}_i}, m \rangle)$ with $m \leftarrow ID_{context}$, for a pre-existing Trustee Registration Certificate with metadata $m = ID_{context}$ to identify in which context $\mathcal{U}_j$ trusts $\mathcal{T}_i$. The first element is the verification key of the receiver of the trust assignment $\mathsf{vks}_{\mathcal{U}_j}$ which assigns trust, while the second is the verification key $\mathsf{vks}_{\mathcal{T}_i}$ of the receiver of the rights, *i.e.* the delegate, while the third element is the certificate's metadata which contains the identification of the context $ID_{context}$. In order to sign the delegation certificate, the $\mathcal{U}_j$ accesses $\mathcal{F}_{\mathcal{T}}$ via the *Assign Trust* interface and then publishes $\Sigma = (d, \sigma)$ on the ledger. The trust assignment is publicly verifiable via the *Verify Trust* interface. Therefore we have the following definition.

Definition 4 (Trust Assignment). *It is said that the participant u assigns trust to $\mathcal{T}$ for a context ID when there is a Registration Certificate Σ_{reg} =*

$((\mathsf{vks}_{\mathcal{T}}, ID), \sigma_{\mathcal{T}})$ published in the ledger $\mathcal{L}$, and u publishes the certificate $\Sigma_{trust} = (d_u, \sigma_u)$ such that $d_u = (\mathsf{vks}_u, \langle \mathsf{vks}_{\mathcal{T}}, ID \rangle)$.

Protocol and Security. As early mentioned, the functionality $\mathcal{F}_{\mathcal{T}}$ is an extension of $\mathcal{F}_{\text{CoreWallet}}$ from [18]. Likewise, $\mathcal{F}_{\text{CoreWallet}}$, which has a corresponding protocol $\pi_{\text{CoreWallet}}$, we claim, without giving a concrete construction, that there is a corresponding protocol $\pi_{\mathcal{T}}$ which UC realizes $\mathcal{F}_{\mathcal{T}}$, given

– the signature scheme $\Sigma = \langle \mathsf{KeyGen}, \mathsf{Verify}, \mathsf{Sign} \rangle$ is Existential Unforgeability under Chosen Message Attack security (EUF-CMA);
– internal building blocks described in [18] and its security properties, *i.e.* RTagGen, HKeyGen, and GenAddr functions.

Presenting $\pi_{\mathcal{T}}$ and fully proving its security here would be tedious for the reader since it contains the basic technique of $\pi_{\text{CoreWallet}}$, thus we present the following theorem (without proof), assuming our claim that there is a protocol $\pi_{\mathcal{T}}$. Our next theorem states that $\pi_{\mathcal{T}}$ realizes $\mathcal{F}_{\mathcal{T}}$.

Theorem 1. *Let the protocol $\pi_{\mathcal{T}}$ be parameterized by a signature scheme $\Sigma = \langle \mathsf{KeyGen}, \mathsf{Verify}, \mathsf{Sign} \rangle$ and the* RTagGen*,* HKeyGen*, and* GenAddr *be functions. Then $\pi_{\mathcal{T}}$ securely realizes the ideal functionality $\mathcal{F}_{\mathcal{T}}$ if and only if Σ is EUF-CMA,* GenAddr *is collision resistant and attribute non-malleable,* RTagGen *is collision resistant, and* HKeyGen *is hierarchical for Σ.*

A proof for the earlier theorem derives very closely to the main theorem in [18] given that our adapted $\mathcal{F}_{\mathcal{T}}$ contains only minor changes to the original functionality definition. Therefore, as mentioned earlier, we skip a detailed proof.

Concrete Stake Based Reputation System. In order to concrete instantiate a stake based reputation system $\mathsf{Rep}^m_{\mathbb{S}}$, assume there are two sets $\mathcal{U}$ and $\mathcal{T}$, respectively the sets of regular participants and trustees, then assume there are n regular participants $u_1, \ldots, u_n$ who have respectively the following shares $\mathbb{S}(\alpha_1), \ldots, \mathbb{S}(\alpha_n)$ with respect to the ledger $\mathcal{L}$, for their respective addresses. Moreover, we assume that for a particular context (to be formally defined later), there are m *trustees* $\mathcal{T}_1, \ldots, \mathcal{T}_m$ which are target of the trust from the regular participants. Each single regular participant can publicly assign its own trust to multiple trustees.

Definition 5 (Stake based Reputation System). *Let $\mathbb{S}$ be the combination, the sum $\mathbb{S}(u_1) + \cdots + \mathbb{S}(u_n)$, of all the shares of participants. The reputation system $\mathsf{Rep}^m_{\mathbb{S}} = (r^m_1, \ldots, r^m_m)$ such that $r^m_i = \frac{\mathbb{S}(\mathcal{T}_i)}{\mathbb{S}}$, and $\mathbb{S}(\mathcal{T}_i)$ is the combination of the stake assigned to $\mathcal{T}_i$.*

Note that the earlier definition is handy because stakes are used to quantify the amount of trust assigned. Thus $\mathsf{Rep}^m_{\mathbb{S}}$ provides the percentile of how much stake a trustee receives as trust assigned with respect to all stake assigned for that context. Thus, once a participant u_i assigns trust to several trustees, its stake is taken into account only once in the total combination s, however it is taken multiple times on each value $\mathcal{T}_i$. We stress that with access to the functionalities $\mathcal{L}$ and $\mathcal{F}_{\mathcal{T}}$, respectively to verify the stakes of each address and the validity of the assignment of trust, any participant can compute the reputation vector, with respect to a context identification string $ID_{context}$.

Feasibility of Honest Majority with Respect to Stakes. Next, we take a closer look on $\mathsf{Rep}_{\mathbb{S}}^{m}$ and the guarantees it offers in order to provide a honest majority. Our approach is similar to the one in [2, Claim 3.2], which relates to the reputation values and the number of participants. In the terminology of our work, it is equivalent to the reputation values and the number of honest trustees. However, given our concrete reputation system $\mathsf{Rep}_{\mathbb{S}}^{m}$, we provide a lemma that relates the honest stake of the regular participants and the number of honest trustees. In the following, we slightly abuse the notation by denoting $\mathbb{S}(\mathcal{U})$ as the combination, *i.e.* sum, of stakes in the set $\mathcal{U}$.

Lemma 2. *Let $m : \mathbb{N} \to \mathbb{N}$ be such that $O(\log m(\lambda)) = O(\log \lambda)$, let $\mathsf{Rep}_{\mathbb{S}}^{m} = \{\mathsf{r}^{m(\lambda)}\}_{\lambda \in \mathbb{N}}$ be a family of reputation vectors, $m = m(\lambda)$, and m trustees $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and n regular participants $u_1, \ldots, u_m$. If the two following conditions hold*

- *given that all honest regular participants $\mathcal{U}_h \subseteq \mathcal{U}$ assign trust* only *to honest trustees $\mathcal{T}_i$;*
- *for the subset of honest regular participants $\mathcal{U}_h$, it holds that*

$$\frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} > \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m}),$$

then there exist a negligible function $\mu(\lambda)$ such that for every λ

$$\Pr_{I \leftarrow \mathsf{r}^m}\left[|I| \geq \left\lfloor \frac{m}{2} \right\rfloor\right] < \mu(\lambda).$$

Proof. Since all honest regular participants assign trust only to honest trustees as given by hypothesis, and the reputation system $\mathsf{Rep}_{\mathbb{S}}^{m}$ is defined as $r_i^m = \frac{\mathbb{S}(\mathcal{T}_i)}{\mathbb{S}}$ then

$$\sum_{i=1}^{m} r_i^m = |\mathcal{U}_h| \cdot \frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} + (m - |\mathcal{U}_h|) \cdot \frac{\mathbb{S} - \mathbb{S}(\mathcal{U}_h)}{\mathbb{S}}.$$

Given the security assumption $\frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} \geq \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m})$, we infer that

$$\sum_{i=1}^{m} r_i^m \geq \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m}).$$

The claim stated in Section 2 gives the proof.

4 Secure MPC and Reputation-Fair Lottery

From now we argue that our Lemma 2 translates the main positive results from [2] to the setting of PoS by considering the stakes of each participant, *i.e.* secure computation and subset with honest majority. Later, we also explore the consequences of the previous lemma with respect to [22].

4.1 Revisiting the Positive Results of [2]

Concretely, a set of parties willing to run a secure protocol that requires honest majority, like, for example, GMW [13], would need access to a ledger $\mathcal{L}$ and $\mathcal{F}_{\mathcal{T}}$, in order to compute the functionality $\mathcal{F}$. In order to present the next general theorem, likewise [2], we need to review a known fact.

Fact. *[2] Let $\mathcal{F} = \{f^m\}_{m\in\mathbb{N}}$ be a functionality and let π denote the GMW protocol for $\mathcal{F}$. Then, for every polynomial $m(\cdot) : \mathbb{N} \to \mathbb{N}$, the protocol $\Pi(m, \lambda)$ $\frac{m(\lambda)}{2}$- securely computes $\mathcal{F}$ with respect to $m(n)$.*

The earlier fact is required by the following theorem.

Theorem 2. *Given the two sets of regular participants $\mathcal{U}$ and trustees $\mathcal{T}$ with access to a ledger $\mathcal{L}$ and a trust assignment functionality $\mathcal{F}_{\mathcal{T}}$, and $|\mathcal{T}| = m$. Let $\mathcal{F} = \{f^m\}_{m\in\mathbb{N}}$ be a functionality, and assume $\pi = \{\pi(m, \lambda)\}$ be the GMW protocol as stated in the earlier Fact. Moreover assume $m(\cdot)$ is a function such that $O(\log m(\lambda)) = O(\log \lambda)$, let $m = m(\lambda)$ and $\mathsf{Rep}^m_{\mathbb{S}}$ be as given by Definition 5. If the two following conditions hold*

- *given that all honest regular participants $\mathcal{U}_h \subseteq \mathcal{U}$ assign trust* only *to honest trustees $\mathcal{T}_i$;*
- *for the subset of honest regular participants $\mathcal{U}_h$, it holds that*

$$\frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} > \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m}),$$

then π securely computes $\mathcal{F}$ with respect to $(m(\cdot), \mathsf{Rep}^m_{\mathbb{S}})$.

As expected the proof of the theorem is immediate given the similarity with the equivalent result in [2]. The crucial observation for the proof is that our Lemma 2 guarantees honest majority, albeit considering the assignment of stake, with negligible probability in λ. Hence the ideal and real execution, as stated by Definition 3, are indistinguishable.

Subset Honest Majority. For completeness, we also state the result regarding finding a *subset honest majority* which is motivated by [9], which states that in order to achieve secure computation with complete fairness, it suffices to have a subset of participants that with negligible probability contains an honest majority. In order to explore this subset, we slightly depart from our initial model by assuming a subset $T \subseteq \mathcal{U}$ which, in the setting of [9], would perform the computation. We emphasize that it is equivalent to say, given our framework with $\mathcal{L}$, $\mathcal{F}_{\mathcal{T}}$ and a arbitrary context given by a string $ID_{context}$, that the subset means basically that the regular participants $\mathcal{U}$ are issuing Registration Certificates and publishing them in $\mathcal{L}$ in order to receive trust within members of the $\mathcal{U}$.

Thus, from [9], as long as T contains an honest majority except with negligible probability, there is a protocol for $\mathcal{U}$ and a family of reputation vectors. In terms of stake, we can state

Lemma 3. *Let $n(\cdot)$, $m(\cdot)$, $\mathcal{F}$, $\mathcal{U}$ and $\mathsf{Rep}^n_{\mathbb{S}}$ be defined as before. If there exists a negligible function $\mu(\cdot)$, such that*

- *for every λ there exists a subset $T_\lambda \subset \mathcal{U}$, with $|T_\lambda| = m$, for which*

$$\Pr_{I \leftarrow \mathsf{r}^{n(\lambda)}} \left[|T_\lambda \cap I| \leq \frac{|T_\lambda|}{2} \right] \leq \mu(\lambda),$$

- *given that all honest regular participants $\mathcal{U}_h \subseteq \mathcal{U}$, with $|\mathcal{U}| = n$, assign trust* only *to honest participants within $\mathcal{U}$;*
- *for the subset of honest regular participants $\mathcal{U}_h$, it holds that*

$$\frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} > \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m});$$

then there exists a (non-uniform) protocol π that securely computes *$\mathcal{F}$ with respect to $(m(\cdot), \mathsf{Rep}^n_{\mathbb{S}})$.*

The proof for this lemma is similar to the one from Theorem 2, therefore we skip it. Moreover, likewise [2] and as highlighted there, the subset T_λ may differ across the values of λ, which results in the claim that the protocol π is non-uniform.

Reputation Vector Criteria for* $\mathsf{Rep}^n_{\mathbb{S}}$*. The earlier Lemma 3 shows that given a subset of parties T it is possible to compute a functionality $\mathcal{F}$ with respect to $\mathsf{Rep}^n_{\mathbb{S}}$. However it says nothing regarding when $\mathsf{Rep}^n_{\mathbb{S}}$ gives a subset with honest majority except with negligible probability. Here, once again due to the similarity of the proofs, in particular with the Lemma 2, we will skip them for the next lemmas.

Lemma 4. *Let $m(\cdot)$, $n(\cdot)$ and $\mathsf{Rep}^n_{\mathbb{S}}$ be defined as earlier. For every λ and subset $T_\lambda \subseteq \mathcal{U}$. If there is a series of subsets $\{T_\lambda\}_{\lambda \in \mathbb{N}}$, with $|T_\lambda| = m$, and*

- *all honest regular participants $\mathcal{U}_h \subseteq \mathcal{U}$, with $|\mathcal{U}| = n$, assign trust* only *to honest participants within $\mathcal{U}$, and $\Delta_{T_\lambda} \overset{def}{=} \frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} - \frac{|T_\lambda|}{2}$, such that $\frac{(\Delta_{T_\lambda})^2}{|T_\lambda|} = \omega(\log \lambda)$;*
- *for the subset of honest regular participants $\mathcal{U}_h$, it holds that*

$$\frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} > \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m}),$$

then there exists a negligible function $\mu(\cdot)$ such that for every λ,

$$\Pr_{I \leftarrow \mathsf{r}^{n(\lambda)}} \left[|T_\lambda \cap I| \leq \frac{m}{2} \right] \leq \mu(\lambda).$$

Given the Lemmas 3 and 4, which, respectively, give us the existence of a suitable subset T, and a criteria for finding such subset, thus we can conclude, analogously to Theorem 2, but now for subsets, the following

Theorem 3. *Given the set of regular participants $\mathcal{U}$ with access to a ledger $\mathcal{L}$ and a trust assignment functionality $\mathcal{F}_{\mathcal{T}}$. Let $\mathcal{F} = \{f^m\}_{m \in \mathbb{N}}$ be a functionality, $m(\cdot)$, $n(\cdot)$ and $\mathsf{Rep}^n_{\mathbb{S}}$ be defined as earlier. Now assume that the following conditions hold*

- *there is a series of subsets $\{T_\lambda\}_{\lambda \in \mathbb{N}}$, with $|T_\lambda| = m$, such that $\Delta_{T_\lambda} \overset{def}{=} \frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} - \frac{m}{2}$ and $\frac{(\Delta_{T_\lambda})^2}{m} = \omega(\log \lambda)$;*
- *all honest regular participants $\mathcal{U}_h \subseteq \mathcal{U}$ assign trust* only *to honest participants within $\mathcal{U}$;*
- *for the subset of honest regular participants $\mathcal{U}_h$, it holds that*

$$\frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} > \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m}),$$

then there is (non-uniform) protocol π which securely computes *$\mathcal{F}$ with respect to $(m(\cdot), \mathsf{Rep}^n_{\mathbb{S}})$.*

Finding a Conservative Subset** T **via Stake. The generation of the subset T_λ or an equally valid subset given the non-uniformity of the protocol, is as straightforward as presented in [2]. The idea is to sort out the participants $\mathcal{U}$ in decreasing order of reputation, and then selecting the highest reputable members of the set. What differs from [2] is that, here, we rely primely on the stakes of each member of the set as they are published on $\mathcal{L}$. Concretely, let $u_{i_1}, \ldots, u_{i_n}$ be the members of $\mathcal{U}$ sorted in decreasing order of reputation, as they were provided by $\mathsf{Rep}^n_{\mathbb{S}}$, and each respective stake $\mathbb{S}(u_{i_1}), \ldots, \mathbb{S}(u_{i_n})$ in $\mathcal{L}$. Then

1. for every $j = 1, \ldots, n$, compute $\Delta_j = \sum_{k=1}^{j} \frac{\mathbb{S}(u_{i_k})}{\mathbb{S}} - \lfloor \frac{j}{2} \rfloor$;
2. if j^* is the index such that $\frac{(\Delta_{j^*})^2}{j^*}$ is maximum over all indexes j, then output the set $T = \{i_1, \ldots, i_{j^*}\}$.

Roughly the above routine shows a conservative approach. That is, it selects *always* the highest reputation among the reputation vector. Without a change in the reputation vector, the output of the selected T set remains the same.

4.2 Revisiting Reputation-Fair Lottery [22]

Given the early description for finding a subset T with the guarantees it contains an honest majority with high probability. However, as already mentioned, and also pointed out in [22], the earlier method for finding T is not suitable because members with low reputation score, would not be selected. An alternative method was introduced by [22] and it is not based on the conservative approach from [2], but on partition of the candidate set, which in our case is $\mathcal{U}$. Each partition, or *tier*, $\mathcal{P}_i$ would aggregate users $u_j \in \mathcal{U}$ with similar reputation score, such that for a number of w tiers, $\mathcal{U} = \bigcup_{i=1}^{w} \mathcal{P}_i$. The procedure associates, for each set $\mathcal{P}_i$, a fixed number $\ell_i \in \mathbb{N}$ representing the number of participants to be picked in a random fashion on each sampling.

Given this setting, roughly, the procedure would progress in rounds. In the first round it randomly picks the set $\widehat{\mathcal{P}_1} \xleftarrow{\$} \mathcal{P}_1$ with $|\widehat{\mathcal{P}_1}| = \ell_1$, in the second round it randomly picks the set $\widehat{\mathcal{P}_2} \xleftarrow{\$} \mathcal{P}_1 \bigcup \mathcal{P}_2$, with $|\widehat{\mathcal{P}_2}| = \ell_2$, until $i = n$. It is not hard to understand the difference from the previous method, since this approach would even provide chances for participants with lower reputation score, given that the tiers are sorted in decrease order, *i.e.* members of $\mathcal{P}_n$ have the lowest reputation scores.

The authors in [22] formally showed that, with access to a reputation system Rep that presents the so called *feasibility* property, *i.e.* guarantee of an honest majority under right conditions, their introduced algorithm for lottery, say $\mathsf{L}^{\mathsf{Rep}}$, which is based on partitioning the set $\mathcal{U}$ is *fair* for suitable definition of *reputation fairness* they provide. Given that the analogous *feasibility* property in our work is the Lemma 4, we have the following corollary.

Corollary 1. *Let $n(\cdot)$, $\mathcal{U}$ and $\mathsf{Rep}^n_{\mathbb{S}}$, with $|\mathcal{U}| = n$, be defined as before. For every λ, there is a series of subsets $\{T_\lambda\}_{\lambda \in \mathbb{N}}$ with $|T_\lambda| = m$. If the following hold*

– *all honest regular participants $\mathcal{U}_h \subseteq \mathcal{U}$, assign trust* only *to honest participants within $\mathcal{U}$, and $\Delta_{T_\lambda} \overset{def}{=} \frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} - \frac{m}{2}$, such that $\frac{(\Delta_{T_\lambda})^2}{m} = \omega(\log \lambda)$;*

– *for the subset of honest regular participants $\mathcal{U}_h$, it holds that*

$$\frac{\mathbb{S}(\mathcal{U}_h)}{\mathbb{S}} > \left\lfloor \frac{m}{2} \right\rfloor + \omega(\sqrt{m \cdot \log m}),$$

then there is a lottery algorithm $\mathsf{L}^{(\cdot)}$, such that $\mathsf{L}^{\mathsf{Rep}^n_{\mathbb{S}}}$ is reputation-fair.

The proof of the corollary is immediate as one should notice that given the initial conditions, which are the same of Lemma 4, the set $\mathcal{U}$ yields a subset with honest majority which is the only requirement for the reputation system from the Theorem 1 in [22]. Thus, $\mathsf{L}^{\mathsf{Rep}^n_{\mathbb{S}}}$ is reputation-fair as $\mathsf{L}^{(\cdot)}$ is the lottery algorithm given in [22].

5 Final Remarks

We have extended the Core Wallet Functionality from [18]. In addition to its regular stake delegation use, our proposed $\mathcal{F}_{\mathcal{T}}$ allows a participant of the PoS consensus protocol to "assign trust", without harming the ledger consensus protocol. The immediate consequence is that we could proposed in this work the creation of a reputation system based on stake of the "trust assigners". This concrete design of a reputation system allowed us to revisit the works of [2] and [22] which deals with the performing of relevant MPC protocols and the construction of proof of reputation protocol resistant to sybil attacks given the stake distribution underpinning the system, *i.e.* the PoS consensus protocol.

Our work is relevant because given existing global PoS ledger in place, groups of users can gather and build reputation around a context of their choice. Furthermore, each honest participant, based on which of the other participants it trusts, can individually verify whether a certain group of players, say $\mathcal{T}$, who jointly received the trust assignments from a community of other players, yields an honest majority. This verification can be done just by simple checking of the ledger and verifying the stake distribution with respect to the trust assignments. This is an enhancement in comparison to [2,22]. Although these works do deal with reputation, they do not provide insights on how the reputation score is computed nor how it can be verified. Let alone to integrate them in concrete, and deployed, PoS ledgers.

A drawback of the work is that the guarantees of honest majority are based on the public perception of honesty of the trustees, which can be misleading. However, given that the trust assignment of our construction is very dynamic and publicly verifiable, once a misbehavior trustee is identified, the trust assignment can be easily revoked by the users, which promotes accountability of actions by the trustees.

The introduction of the stake, from a PoS system, into the area of reputation systems may suggest analysis based on rational adversaries or game theory. Such approach was also suggested in [2]. We leave this next step for future work.

References

1. G. Akerlof. *The Market for "Lemons": Quality Uncertainty and the Market Mechanism*, pages 175–188. Macmillan Education UK, London, 1995.
2. Gilad Asharov, Yehuda Lindell, and Hila Zarosim. Fair and efficient secure multiparty computation with reputation systems. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 201–220. Springer, Heidelberg, December 2013.
3. Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 913–930, New York, NY, USA, 2018. ACM.
4. Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 324–356. Springer, Heidelberg, August 2017.
5. Alex Biryukov, Daniel Feher, and Dmitry Khovratovich. Guru: Universal reputation module for distributed consensus protocols. Cryptology ePrint Archive, Report 2017/671, 2017. `http://eprint.iacr.org/2017/671`.
6. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.
7. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007.
8. EOS Community. Eos.io technical white paper v2, 2018. `https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md`.
9. Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 378–394. Springer, Heidelberg, August 2005.
10. Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Cryptology ePrint Archive, Report 2017/573, 2017. `http://eprint.iacr.org/2017/573`.
11. Tassos Dimitriou. Decentralized reputation. Cryptology ePrint Archive, Report 2020/761, 2020. `https://eprint.iacr.org/2020/761`.
12. Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. Cryptology ePrint Archive, Report 2014/765, 2014. `http://eprint.iacr.org/2014/765`.
13. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
14. LM Goodman. Tezos—a self-amending crypto-ledger white paper, 2014.
15. Andreas Gutscher. A trust model for an open, decentralized reputation system. In *IFIP International Conference on Trust Management*, pages 285–300. Springer, 2007.
16. Wassily Hoeffding. *Probability Inequalities for sums of Bounded Random Variables*, pages 409–426. Springer New York, New York, NY, 1994.
17. Audun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

18. Dimitris Karakostas, Aggelos Kiayias, and Mario Larangeira. Account management in proof of stake ledgers. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 3–23. Springer, Heidelberg, September 2020.
19. T. Kerber, A. Kiayias, and M. Kohlweiss. Kachina - foundations of private smart contracts. In *2021 2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pages 47–62, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.
20. Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019.
21. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, August 2017.
22. Leonard Kleinrock, Rafail Ostrovsky, and Vassilis Zikas. Proof-of-reputation blockchain with nakamoto fallback. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 16–38. Springer, Heidelberg, December 2020.
23. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
24. Lakshmi Ramachandran. Behavior-based popularity ranking on amazon video. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, page 564–565, New York, NY, USA, 2020. Association for Computing Machinery.
25. Steem. Steem whitepaper, 2018. `https://steem.com/steem-whitepaper.pdf`.
26. Peter Wuille. Decentralized identifiers (dids) v1.0. `https://en.bitcoin.it/wiki/BIP_0032`, 2021. [Online; accessed 23-February-2021].

A Technical Challenges for Reputation Systems

The major technical challenges for reputation systems can be divided in three major areas.

– eliciting feedback: Systems based on ranking crucially depend on the feedback of the system users. Briefly, a reputation framework needs to continuously gather and process the "opinion" from the parties;
– aggregating/displaying: The feedback from the user has to be aggregated in order to establish a comprehensible and meaningful value;
– distribution: Each player of such system is subject to receive a "reputation" value, which others can consult or be used in decision making, in particular, it can be used in outlining general strategies.

Eliciting Feedback. In general, participants are entitled to decide in providing feedback or not. Therefore without proper incentive we may be led to the situation that we would have a highly biased view of the reputation of a particular target, since only parties willing to spread bad reputation, for some particular reason, would bother to interact with the system in order to feed with its opinion. The quality of the feedback is also subject to problems. Ideally, the desirable feedback are the honest ones, reporting bad or good behavior alike, however it is notoriously hard to obtain or verify. Furthermore, bad report, if public, can even cause retaliation from the feedback target in future interactions. One approach to incentivize “good feedback”, or at least “valid feedback”, may be to somehow tie together some sort of stake to the participant providing such feedback.

Aggregating and Displaying. A highly popular technique to process feedback is simply associate a score with the target node, with respect to some service/interaction. In extreme cases, the nodes have only two options “good” or “bad”, *i.e.* 1 or -1. In this binary feedback scenario, reputation aggregators would apply a simple technique named *netfeedback*. Basically it is the simple sum of the feedbacks in order to provide a straightforward “good” or “bad” status , which, later, can be used to assemble a *ranking*, for the target nodes. This approach is widely deployed in major Internet websites, however it has clear setbacks, since it does not take into account the type of interaction which led to the reported score. More concretely, the target node which interacted with several other feedback providers but with respect with difference services/products,

Reputation Distribution. Once a reputation is established, as, for example, a simple score, a major challenge is to bind it with the parties and allow public availability, including compatibility with other systems. In an environment where users/identities can easily be created, participants with bad reputation can create new identities effectively erasing past history, and bad reputation, *i.e. whitewashing.* The newly created node would be free of the bad effects of its past actions. Here, once again some sort of relation with a stake, have the potential to minimize such erasure. A large amount of work was dedicated to minimizing such effects in game theory. A common strategy is to penalize the newcomer until it has enough history within the system. Previous work had shown that given the risk of *whitewashing*, the system as whole has to carry the burden in the form of penalties for every newcomer.

B The Proof of Stake Ledger

The main novelty of our work is that we base our reputation system on the use of stake of each participants have in a proof of stake $\mathcal{L}$. For our purposes, it suffices that we assume that the consensus protocol performs in a predefined number of rounds which compose a *slot*, besides each block is associated with a single slot. In our reputation system, *stakeholders*, *i.e.* participants of the PoS system who have stake in it, can publicly *assign trust* to other participants via a mechanism similar to the delegation mechanism of [18], which will be clarified in Section 3. In a nutshell, each participant, as a regular user of the PoS system, have access to a wallet functionality, which in our case, is an extension of the one in [18], *i.e.* it has extra features to assign trust.

For completeness, we now describe the model for our protocol. As a regular participant of a PoS system, the parties are assumed to have synchronous communication where messages sent by honest players are delivered by the beginning of the following round. Besides we also assume the presence of a rushing adversary [6], which can actively corrupt parties. Regarding communication capabilities, we assume all the players have access to a diffusion (multicast) channel which can be built by standard flooding/gossip protocols.

B.1 Transactions, Blocks, and the Global Ledger

Our protocol relies on the Kachina framework [19]. In particular we rely on its formulation of ledger due to its simplicity.

The Simple Ledger Functionality. Our trust model assumes each user interacts with a ledger functionality in a hybrid execution, and a secure ledger is described as follows.

Definition 6. *A secure distributed ledger [12] satisfies the following properties:*

- ***Persistence**: A transaction which is part of a block at least k blocks away from the ledger's head, i.e. a block which is part of the chain which results from removing the last k blocks of the current chain, is stable, i.e. every honest party reports it in the same position in the ledger.*
- ***Liveness**: A transaction which is provided continuously as input to the parties is stable after u rounds.*

One option is Bitcoin's formalization from [4]. However, this functionality is local, while we would prefer a global functionality, following the Global UC Framework [7], hence we will use the $\overline{\mathcal{G}}_{simpleLedger}$ functionality from Kachina [19], which we denote in this work by $\mathcal{L}$. The functionality is available in Figure 2, where $\prec$ defines the prefix operation, *i.e.* $\Omega \prec \Omega'$ means the state Ω is included in Ω', and, for readability and consistency purposes, we rename *transaction* (τ) to *block* (b).

The functionality $\mathcal{L}$ is generic enough to abstract transactions and blocks, focusing on the ledger's properties. In general, that is enough for our purposes, however, for the sake of completeness and formality, we need to define, more concretely, the entries in the ledger in order to better formulate a real-world blockchain.

Typically a transaction is the tuple $tx = (\Theta, \alpha_s, \alpha_r, v, f)$, where

1. $\alpha_s, \alpha_r \in \{0,1\}^*$ are the sender's and receiver's addresses respectively, for the tradable asset set Θ,
2. $v \in \mathbb{R}$ is the value transferred from α_s to α_r, and
3. $f \in \mathbb{R}$ is the fees of the transaction.

A block consists of an ordered list of transactions. In order to organize transaction in blocks, we assume a function blockify which, given a set of transactions and a chain, returns a block which can extend the chain.

Global Ledger Functionality $\mathcal{L}$

The functionality keeps a state Ω and a mapping M of parties to states, both initially empty.

- When receiving a message (SUBMIT, b) from a party P, query $\mathcal{A}$ with (BLOCK, b).
- When receiving a message READ from a party P, return $M(P)$; if P is $\mathcal{A}$, it returns Ω.
- When receiving a message $(\mathsf{EXTEND}, \Omega')$ from $\mathcal{A}$, set $\Omega \leftarrow \Omega || \Omega'$.
- When receiving a message $(\mathsf{ADVANCE}, P, \Omega')$ from $\mathcal{A}$, if $M(P) \prec \Omega' \prec \Omega$ then set $M(P) \leftarrow \Omega'$.

Fig. 2. The Simple Global Ledger Ideal Functionality.

C The Stake Delegation Framework

In a nutshell [18] introduces a mechanism for issuing address strings which contain attributes. The immediate purpose of the framework is to capture the property of delegated PoS system, where users can delegate the right of its own stakes to special consensus protocol participants named *stake pools*. The proposed framework was presented as a universally composable functionality capable of generating addresses.

More concretely, their framework defines a core-wallet functionality $\mathcal{F}_{\text{CoreWallet}}$, representing the key management capabilities of a concrete wallet. That is, every PoS has an internal core which manages the private information with respect to that wallet, and therefore all its addresses, forming a *account*. Intuitively, the functionality $\mathcal{F}_{\text{CoreWallet}}$ offers all the actions that a PoS wallet needs to perform the basic actions such as *staking*, *recover*, *initialization* and etc, including the creation of addresses α, while keeping an internal central registry of transactions, *i.e.* $\mathcal{T}$. For completeness the full description of $\mathcal{F}_{\text{CoreWallet}}$ is given by Figures 3 and 4.

Their framework introduces a predicative M for malleability of addresses and describes three types of addresses, *i.e.* *base*, *pointer* and *exile*, according to a list of attributes δ (which can contain cryptographic keys). The intuition of the predicative M is that it outputs correctly if the address is rightfully generated, *i.e.*, it is not a result of malicious change in a previously generated address string. Briefly, it can be set to output 1 for correctly generated addresses or 0 otherwise, casting $\mathcal{F}_{\text{CoreWallet}}$ a functionality that does not accept *malleable addresses*. Other definitions for M are possible which sets $\mathcal{F}_{\text{CoreWallet}}$ to operate with different malleability tolerant modes. We refer the reader to [18] for a complete discussion on the topic.

Functionality $\mathcal{F}^{M}_{\text{CoreWallet}}$ (First Part)

Initialization: Upon receiving $(\textsc{Init}, sid)$ from $P \in \mathbb{P}$, forward it to $\mathcal{S}$ and wait for $(\textsc{InitOk}, sid)$. Then initialize the empty lists L_P of addresses and attribute lists and K_P of staking keys, and send $(\textsc{InitOk}, sid)$ to P.
Wallet Recovery: Upon receiving $(\textsc{RecoverWallet}, sid, i)$ from $P \in \mathbb{P}$, for the first i elements in L_P return $(\textsc{Tag}, sid, \delta_2)$.
Address Recovery: Upon receiving $(\textsc{RecoverAddr}, sid, \alpha, i)$ from P, if (α, l) is one of the first i elements of L_P or $M(L_P, \text{“recover”}, \alpha) = 1$, return $(\textsc{RecoveredAddr}, sid, \alpha)$.
Address Generation: Upon receiving $(\textsc{GenerateAddress}, sid, aux)$ from $P \in \mathbb{P}$, forward it to $\mathcal{S}$. Upon receiving $(\textsc{Address}, sid, \alpha, l_\alpha)$ from $\mathcal{S}$, parse l_α as $(\delta_1, \ldots, \delta_g)$ and $\forall P' \in \mathbb{P}$ check if $\forall(\alpha', (\delta'_1, \ldots, \delta'_g)) \in L_{P'}$ it holds that $\alpha \neq \alpha'$, $\delta'_2 \neq \delta_2$, and $\forall j \in [i, \ldots, g] : \delta'_j \neq \delta_j$, *i.e.* the address, recovery tag, and private attributes are unique. If so, then:

- if $aux = (\text{“base”})$, check that $\forall(\alpha', (\delta'_1, \ldots, \delta'_g)) \in L_P : \delta'_1 \neq \delta_1$,
- else if $aux = (\text{“pointer”}, \mathsf{vks})$, check that $\delta_1 = \mathsf{vks}$,
- else if $aux = (\text{“exile”})$, check that $\delta_1 = \perp$.

If the checks hold or P is corrupted, then insert (α, l_α) to L_P and return $(\textsc{Address}, sid, \alpha)$ to P. If $aux = (\text{“base”})$ also insert δ_1 to K_P and return $(\textsc{StakingKey}, sid, \delta_1)$ to P.
Issue Transaction: Upon receiving $(\textsc{Pay}, sid, \Theta, \alpha_{\mathsf{s}}, \alpha_{\mathsf{r}}, m)$ from $P \in \mathbb{P}$, if $\exists l_\alpha : (\alpha_{\mathsf{s}}, l_\alpha) \in L_P$ forward it to $\mathcal{S}$. Upon receiving $(\textsc{Transaction}, sid, tx, \sigma)$ from $\mathcal{S}$, such that $tx = (\Theta, \alpha_{\mathsf{s}}, \alpha_{\mathsf{r}}, m)$, check if $\forall(tx', \sigma', b') \in \mathcal{T} : \sigma' \neq \sigma$, $(tx, \sigma, 0) \notin \mathcal{T}$, and $M(L_P, \text{“issue”}, \alpha_r) = 1$. If all checks hold, then insert $(tx, \sigma, 1)$ to $\mathcal{T}$ and return $(\textsc{Transaction}, sid, tx, \sigma)$.
Verify Transaction: Upon receiving $(\textsc{VerifyPay}, sid, tx, \sigma)$ from $P \in \mathbb{P}$, with $tx = (\Theta, \alpha_{\mathsf{s}}, \alpha_{\mathsf{r}}, m)$ for a metadata string m, forward it to $\mathcal{S}$ and wait for a reply message $(\textsc{VerifiedPay}, sid, tx, \sigma, \phi)$. Then:

- if $M(L_P, \text{“verify”}, \alpha_s) = 0$, set $f = 0$
- else if $(tx, \sigma, 1) \in \mathcal{T}$, set $f = 1$
- else, if P is not corrupted and $(tx, \sigma, 1) \notin \mathcal{T}$, set $f = 0$ and insert $(tx, \sigma, 0)$ to $\mathcal{T}$
- else, if $(\Theta, \alpha_s, \alpha_r, m, \sigma, b) \in \mathcal{T}$, set $f = b$
- else, set $f = \phi$.

Finally, send $(\textsc{VerifiedPay}, sid, tx, \sigma, f)$ to P.

Fig. 3. The first part of the full Core Wallet Functionality from [18].

Functionality $\mathcal{F}^{M}_{\text{CoreWallet}}$ (Second Part)

Issue Staking: Upon receiving $(\textsc{Stake}, sid, stx)$ from P, such that $stx = (\mathsf{vks}, m)$ for a metadata string m, forward the message to $\mathcal{S}$. Upon receiving $(\textsc{Staked}, sid, stx, \sigma)$ from $\mathcal{S}$, if $\forall(stx', \sigma', b') \in S : \sigma' \neq \sigma$, $(stx, \sigma, 0) \notin S$, and $\mathsf{vks} \in K_P$, add $(stx, \sigma, 1)$ to S and return $(\textsc{Staked}, sid, stx, \sigma)$ to P.
Verify Staking: Upon receiving $(\textsc{VerifyStake}, sid, stx, \sigma)$ from $P \in \mathbb{P}$, forward it to $\mathcal{S}$ and wait for $(\textsc{VerifiedStake}, sid, stx, \sigma, \phi)$, with $stx = (\mathsf{vks}, m)$. Then find P_{s}, such that $\mathsf{vks} \in K_{P_{\mathsf{s}}}$, and:

- if $(stx, \sigma, 1) \in S$, set $f = 1$
- else if P_{s} is not corrupted and $(stx, \sigma, 1) \notin S$, set $f = 0$ and insert $(stx, \sigma, 0)$ to S
- else if exists an entry $(stx, \sigma, f') \in S$, set $f = f'$
- else set $f = \phi$ and insert (stx, σ, ϕ) to S.

Finally, return $(\textsc{VerifiedStake}, sid, stx, \sigma, f)$ to P.

Fig. 4. The second part of the full Core Wallet Functionality from [18].

Stake Pool Registration and Delegation. The stake pools are identified by a registered staking key generated by accessing its internal $\mathcal{F}_{\text{CoreWallet}}$, to compute a new staking key $(\mathsf{vks}, \mathsf{sks})$ pair, respectively verification and secret keys, in order to issue a registration certificate (vks, m), where m is the pool's metadata. By accessing its internal functionality $\mathcal{F}_{\text{CoreWallet}}$, it receives back $((\mathsf{vks}, m), \sigma)$, where σ is the signature corresponding to sks of the tuple (vks, m). Next, it publishes in the ledger the registration certificate $\Sigma_{reg} = ((\mathsf{vks}, m), \sigma)$ via a regular transaction $tx = (\alpha_{reg}, \Sigma_{reg})$, where α_{reg} is the special address of the pool. The stake delegation is also achieved with certificates via a process similar to the staking pool registration described. A delegation certificate is a tuple $d = (\mathsf{vks}_s, \langle \mathsf{vks}_d, m \rangle)$. The first element is the staking key vks_s which assigns of the rights of the stake to someone else, *i.e.* the owner, while the second is the staking key vks_d of the receiver of the rights, *i.e.* the delegate, while the third element is the certificate's metadata. In order to sign the delegation certificate, the participant accesses its internal $\mathcal{F}_{\text{CoreWallet}}$ and then publishes $\Sigma = (d, \sigma)$ on the ledger.

For our purposes, this is a fair description of the delegation methods. However the framework proposes different types of procedure, for a complete description, we refer the reader to the work on [18].

Our trust platform, described in the body of the work, is based on similar ideas for delegation. In particular, we adapt the early mentioned $\mathcal{F}_{\text{CoreWallet}}$, by adding interfaces for special sort of delegation, *i.e.* the "trust assignment" (details on Section 3).