A Privacy-preserving Proof-of-Reputation

Marina DEHEZ-CLEMENTI
ISAE-SUPAERO
Université de Toulouse
Toulouse, France
0000-0003-0783-9848

Mourad RABAH
Laboratoire L3i
La Rochelle University
La Rochelle, France
0000-0001-8136-5949

Yacine GHAMRI-DOUDANE
Laboratoire L3i
La Rochelle University
La Rochelle, France
0000-0002-7986-2476

***Abstract*—The sharing of high-quality information is essential for improving the user experience in distributed systems such as vehicular networks or IoT-based monitoring systems. Crowd-sourced data collection systems rely heavily on a secure and efficient consensus algorithm. However, consensus algorithms are often considered a bottleneck for scalability. With the emergence of blockchains, proof-of-work (PoW) has become the most popular and secure consensus algorithm for open and asynchronous distributed networks. However, PoW has been criticized for its energy consumption and waste of resources. To address this issue, alternative consensus protocols such as proof-of-stake (PoS) and proof-of-authority (PoA) have been proposed. Reputation-based consensus algorithms have recently gained attention but suffer from limitations in terms of privacy and predictability. In this paper, we propose a privacy-preserving proof-of-reputation (PoR) mechanism that combines ring signatures and dlog-based zero-knowledge proof systems. The proposed PoR algorithm aims to build unpredictable blockchain consensus algorithms that are privacy-friendly for permissioned blockchains. We describe the main algorithms used in PoR, demonstrate their security, and evaluate their performance through an experimental study. We conclude by discussing how these algorithms can be scaled to permissionless blockchain settings.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Reputation is traditionally defined as a numerical value, or a score, given to an entity and which evaluates its trustworthiness in a system *w.r.t.* other members of a same environment. In P2P networks, reputation is mostly used to facilitate interactions as each participant can individually decide whether to trust another based on this score [1], [2].

This metric has been reused in the context of blockchains since 2018 for the design of an alternative consensus algorithm family named *Proof-of-Reputation* (PoR). The advantages of using reputation scores in permissioned blockchains are multiple: firstly, since all the identities are known, it reduces the impact of the consensus on the system's efficiency and energy consumption; secondly, since they have the same properties as electronic money (numerical variables, approved by all participants), they are easy to compute and take decisions upon. The scaling to permissionless blockchains can then be managed by a system of locking/unlocking deposits as presented in [3].

In the context of Blockchains, Gramoli defines the consensus problem as the process by which *"the non-faulty or correct processes of a distributed system agree on one block of transactions at a given index of a chain of block"* [4]. The consensus problem must comply with three essential security properties: ***agreement***: no two correct processes decided different blocks; ***validity***: the decided block was proposed by one process; ***termination***: eventually, correct processes will decide on one block. Most of the existing reputation-based consensus algorithms comply with these requirements.

In 2020, Boneh *et al.* introduce one additional security property called *unpredictability*. Applied to blockchain consensus, the property should prevent an attacker from identifying and targeting the next leader (the node elected and authorized to produce a new block of transactions) via a DoS attack in order to make it unavailable, or by bribing them [5], [6]. Current reputation-based consensus algorithms are built upon proofs of reputation that are strongly attached to the identity of the proving peers, as such the election of the next block producer is therefore deterministic. Thus, these PoR-based consensus algorithms do not respect the *unpredictability* property.

In this paper, we propose the construction of a **privacy-preserving proof-of-reputation** mechanism for unpredictable reputation-based consensus algorithms in permissioned blockchain systems. To this end, we combine powerful cryptographic primitives such as ring signatures and traditional zero knowledge proofs for discrete-logarithm (dlog) based cryptosystems with the distributed nature of blockchain systems. In Section II, we analyse the exiting literature on reputation-based consensus algorithms. Then, in Section III, we give the cryptographic background for the proposed scheme. In Section IV, we describe the main algorithms of our PoR, and demonstrate their security in Section V. In Section VI, we detail the implementation and analyze its performance (execution time and memory occupation). Finally, we show their usefulness to design unpredictable privacy-friendly consensus algorithms for permissioned blockchains, and discuss the scaling to permissionless settings in the Section VIII.

II. Related work

In [7], Gai *et al.* design the first reputation-based consensus algorithm for blockchain systems as an alternative to the traditional Proof-of-Work. The algorithm aims to address the known limitations of reputation management systems (namely the presence of a single point of failure and the lack of global

ISBN 978-3-903176-57-7

reputation evidence) by providing a distributed ledger of reputation. Although it is specific for permissioned blockchains, it facilitates the efficient and secure sharing of unmodified reputation scores. The implementation gives promising results in terms of scalability, block production rate and transaction throughput. Nonetheless, the proposition necessitates to query another blockchain, dedicated to reputation management, which complicates its use. In addition, the rewriting of the chain of reputation is technically possible and only cognitively disregarded as the attack would be "too costly to be carried out".

Zhuang *et al.* [8] extend on previous work by proposing a new block structure which enable them to embed reputation scores inside data blocks (thus, transactions can be verified along with the reputation score of their respective issuer). The novelty of the algorithm lies in the leader election method and block validation: indeed, the next *miner* is selected as the node with the highest reputation score among the set of transacting nodes (whose transactions are to be inserted in the next block). Then, the leader submits the new block to a consortium of nodes made of the top 20% with the highest reputation scores. While this selection criteria reduces the communication complexity caused by the block submission and verification process, one can question the justifying Pareto argument as it weakens the distribution of the network, thus facilitating the 51% attacks.

With [9], Cai *et al.* add the dynamicity property. Indeed, up until there, reputation-based consensus algorithms only considered fixed-size networks and focused on developing the reputation attribution function (calculation and update), along with the mechanics of leader election and block validation. To deal with dynamicity, authors divide their network into four categories of nodes, notably the monitoring node. This division enables the authors to define: firstly, a dynamic protocol for adding new node to the network via a join/issue-like interactive protocol with this monitoring node. The communication overhead of the joining process and the latency induced by consensus termination and block validation are thus reduced. Secondly, the update of reputation score is automatic and fast as it is handled by one entity in the network. While the presence of the monitoring node drastically improves the performance of the algorithm, it is also the main identified flaw in this work, as it centralizes the reputation management authority.

Later on that year, De Oliveira *et al.* propose a dynamic reputation-based consensus algorithm which decentralized the role of the monitoring node [10]. Each node has a reputation score which is calculated based on the node's action and its age in the system: oldest nodes are privileged to become miners, they call this metric the *maturity* of the node. Access control is therefore handled by the miners: when a user wants to join it interacts with one or more of the consensus nodes which accept or not to let it in. Each miner is assigned a random group of judges to compute their reputation score. The scheme effectively reduces resources' consumption while maintaining a certain level of decentralization. Moreover, security arguments are given in favor of a resilience against well-known blockchain-related attacks (namely double spend, 51%, impersonation, selfish miner, eclipse and Denial of Service — DoS — attacks). Yet, the group of judges is publicly known and fixed in time, thus one only has to compromise half of one miner's group of judges to influence its reputation score and eventually exclude the targeted node.

All aforementioned proof-of-reputation systems only work in permissioned environments. Instead, Adbo *et al.* propose in [3] the first permissionless reputation-based consensus algorithm. The core idea is to replace the identity-based access control by a monetary deposit. The incentive is no longer the reputation but money: if a node behaves honestly, they get their deposit fund back, otherwise the money is destroyed. Authors' goal is to enable the mixing between their PoR and any other Proof-of-Anything (PoX).

However, none of the work above clearly discusses the *unpredictability* property. Indeed, in previous propositions, the election of the leader is often based on the highest reputation value; thus, it is easy to quickly *predict* which node will mine the next block and to target it (by a Denial of Service attack for instance).

III. Preliminaries

This section examines key concepts and theories that are fundamental to our work.

A. Cryptographic hash functions

A cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is a one-way deterministic function used to map input data of arbitrary size to fixed-size (here of n bits) output values, also called *hash values*. Cryptographic hash functions are a basic tool of modern cryptography and as such must comply with four properties: it should be efficient, and comply with the collision resistance, preimage resistance, and second preimage resistance requirements [11]. In this paper, we consider that hash functions are random oracles and, as such, we are working in the Random Oracle Model [12].

B. Ring Signatures

Let $\mathcal{P} = P_1, \ldots, P_n$ be a set of processes arranged in a distributed system. Let $R \subseteq \mathcal{P}$ be the ring. Each $P_i \in R$ is a possible signer, but only the one that produces the actual signature will be called the signer. A ring signature scheme $RS = (\mathsf{RGen}, \mathsf{RSign}, \mathsf{RVerify})$ consists of the following three Probabilistic Polynomial Time (PPT) algorithms [13]:

- $(rpk, rsk) \leftarrow \mathsf{RGen}(1^k)$: The *key generation algorithm* takes as input the security parameter k and outputs a ring private key rsk also known as the *signing key*, and the corresponding ring public key rpk, also referred to as the *verification key*.
- $\sigma \leftarrow \mathsf{RSign}(m, R, rsk)$: The *signing algorithm* is given a message $m \in \mathcal{M}$ to be signed (with $\mathcal{M}$ the message space), a ring R and the secret key of the signer rsk. It outputs the signature σ of message m.

- $1/0 \leftarrow \mathsf{RVerify}(R, m, \sigma)$: The (deterministic) *verification algorithm*, given a ring R, a message m and a signature σ, outputs either 1 for "accept" or 0 meaning "reject".

We adopt the same security framework as the one presented in [13] and recall the related definitions in Sub-section III-C.

C. Security model for Ring Signatures

In this sub-section, we recall the traditional security model for evaluating the security of ring signatures.

1) Correctness: We adopt the same security notion as presented in [13]. A ring signature scheme RS is secure in the Ideal Cipher Model (ICM) if and only if it is **correct**, *unforgeable under chosen-message attacks* (CMA) and *anonymous under full key exposure* of which we recall the definition in the next paragraphs. Let $n \equiv n(k)$ be a polynomial in k.

For the **correctness property**, we ask that Equation 1 holds for all $k \in \mathbb{N}$, $m \in \mathcal{M}$, $(rpk, rsk) \leftarrow \mathsf{RGen}(1^k)$, and R the ring such that $rpk \in R$:

$$\mathsf{RVerify}(R, m, \mathsf{RSig}(m, R, rsk)) = 1 \tag{1}$$

2) Unforgeability:

Definition 1. *(Unforgeability under CMA)*
Consider the experiment $Exp_{rsig}^{EUF-CMA}$, parametrized by the number of verification keys n, between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1) *$\mathcal{C}$ generates $(rpk_i, rsk_i) \leftarrow \mathsf{RGen}(1^k)$ for all $i \in [\![1, n]\!]$. Then, $\mathcal{C}$ gives the set of verification keys $rp \leftarrow (rpk_1, \ldots, rpk_n)$ to $\mathcal{A}$ and sets $S_{sig} \leftarrow \varnothing$.*
2) *$\mathcal{A}$ is allowed to make signing queries of the form (R, j, m), where R is a ring of public keys, $j \in [\![1, n]\!]$ an index such as $rpk_j \in R$, and $m \in \mathcal{M}$. When $\mathcal{C}$ receives (R, j, m), it computes $\sigma \leftarrow \mathsf{RSign}(m, R, rsk_j)$, sends the signature σ to A and appends (m, R, σ) to S_{sig}.*
3) *$\mathcal{A}$ outputs a tuple (R^*, m^*, σ^*).*

This game defines the following experiment:

$$Exp_{rsig}^{EUF-CMA}(\mathcal{A}) \equiv (\mathsf{RVerify}(R^*, m^*, \sigma^*) = 1) \wedge ((R^*, \cdot, m^*) \notin S_{sig})$$

Thus, a ring signature scheme RS is unforgeable under adaptative chosen-message attacks in the Ideal Cipher Model if for any adversary $\mathcal{A}$, the adversarial advantage in winning the security experiment $Exp_{rsig}^{EUF-CMA}$ in PPT is negligible, i.e. we have:

$$Adv_{rsig,\mathcal{A}}^{CMA}(k) \equiv Pr[Exp_{rsig}^{EUF-CMA}(\mathcal{A}) = 1] \leq negl(k) \tag{2}$$

where $negl(\cdot)$ represents the negligible function.

3) Anonymity:

Definition 2. *(Anonymity)*
Consider the following game, parametrized by the number of verification keys n, between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1) *$\mathcal{C}$ generates $(rpk_i, rsk_i) \leftarrow \mathsf{RGen}(1^k; r_i)$ for all $i \in [\![1, n]\!]$, where r_i is a randomness for generating the keypairs (rpk_i, rsk_i). Then, $\mathcal{C}$ gives the set of the public keys $R = rpk_{i i \in [\![1,n]\!]}$ and the random values $r_{i i \in [\![1,n]\!]}$ to $\mathcal{A}$.*
2) *$\mathcal{A}$ requests a challenge to $\mathcal{C}$ by sending a tuple (i_0, i_1, R^*, m^*), where i_0 and i_1 are indices such that $rpk_{i_0} \in R^*$ and $rpk_{i_1} \in R^*$. Then, $\mathcal{C}$ samples a challenge bit $b \xleftarrow{\$} \{0, 1\}$, computes $\sigma^* \leftarrow \mathsf{RSign}(m^*, R^*, rsk_{i_b})$, and gives σ^* to $\mathcal{A}$.*
3) *$\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$Adv_{rsig,\mathcal{A}}^{anon}(k) \equiv 2 \cdot |Pr[b = b'] - \frac{1}{2}| \tag{3}$$

We say that the ring signature scheme RS satisfies unconditional anonymity under full key exposure if for any computationally unbounded adversary $\mathcal{A}$, $Adv_{rsig,\mathcal{A}}^{anon}(k) \leq negl(k)$ holds.

D. Zero Knowledge Proofs for dlog-based cryptosystems

In this paper, we use a common non-interactive zero-knowledge (NIZK) proof technique for showing the equality of the two discrete logarithms to assert that A knows tsk the temporary secret key associated to the account they used ta during the execution of the application. It is defined by the (P, V) algorithms for respectively generating and verifying the zero knowledge proof of knowledge [14]; practically, we use the same algorithms as described in [15] but on different inputs. The security framework of ZKP for discrete logarithm (dlog)-based cryptosystems is briefly recalled in the next sub-section.

Algorithm 1: $P(x_1, y_1, x_2, y_2, \alpha)$

Comment: To show that $dlog_{x_1}(y_1) = dlog_{x_2}(y_2)$ holds without revealing the discrete logarithm α, a prover proceeds as follows:

Function *Main*:
- Compute $t_1 = x_1^w$ adding $t_2 = x_2^w$ for $w \in_R \mathbb{Z}_q$
- Compute $c = \mathsf{H}(x_1, y_1, x_2, y_2, t_1, t_2)$
- Compute $r = w - \alpha \cdot c(\mod q)$
- Output $\pi = \langle c, r \rangle$

Algorithm 2: $V(x_1, y_1, x_2, y_2, \pi)$

Comment: To check the correctness of a proof $\pi = \langle c, r \rangle$, showing that $dlog_{x_1}(y_1) = dlog_{x_2}(y_2)$ holds, a verifier proceeds as follows:

Function *Main*:
- Compute $t_1' = x_1^r \cdot y_1^c$ adding $t_2' = x_2^r \cdot y_2^c$
- Output VALID if $c = \mathsf{H}(x_1, y_1, x_2, y_2, t_1', t_2')$ hold ; output INVALID otherwise

E. Security model for ZKPs

Informally, a proof of knowledge allows a *prover* to convince (prove to) a *verifier* that he knows a solution of a hard-to-solve problem, s.t. the following properties hold, with L a language:

i. ***completeness***: an honest prover, knowing a solution, can successfully convince the verifier. More formally, if the prover runs its predetermined program P, then for every

constant $c > 0$ and large enough $x \in L$, the verifier accepts the common input x with probability at least $1 - |x|^{-c}$;

ii. ***soundness***: with overwhelming probability, a cheating prover, not knowing any solution, will fail to convince the verifier. In a more formal way, for every program P^*, run by the prover, for every constant $c > 0$ and a large enough $x \notin L$, the verifier rejects x with probability at least $1 - |x|^{-c}$;

iii. ***minimum-disclosure***: the verifier obtain no useful information about the solution the prover knows. [16].

Algorithms P and V from [15] have been proven to comply with the three aforementioned properties [14].

IV. Privacy-preserving Prove'n'Claim mechanism

In this section, we present our Prove'n'Claim scheme for reputation-based applications. It consists in four functions: proofGenerate(·), proofVerify(·), claimGenerate(·) and claimVerify(·). In the following paragraphs, we detail the system model, including considerations regarding the communication, connectivity and synchrony requirements. We then detail each algorithm and illustrate the execution flow in Figure 2. The algorithms are declined into five major phases: during the ① *proof generation* and the ② *proof verification* phases, A generates a proof that they own a threshold reputation score and B verifies this proof; upon acceptance, the ③ *application* starts; and finally, the ④ *claim generation* and ⑤ *claim verification* phases see A gaining or loosing reputation depending on their behavior during the application.

a) System model: We consider a set of N distinct entities (also indifferently referred to as *nodes*, *processes* or *peers*) $\mathcal{P} = (P_1, \ldots, P_N)$, with $N \in \mathbb{N}^*$ as shown on Figure 1. We designate A and B two distinct nodes in $\mathcal{P}$ and consider that A wants to prove to B that its reputation score rs_A is greater than a certain threshold score rs_{thr} without disclosing its identity to B. In the rest of this paper, we will demonstrate that the following algorithms enable A to produce such proof that we named *privacy-preserving proof of reputation*.

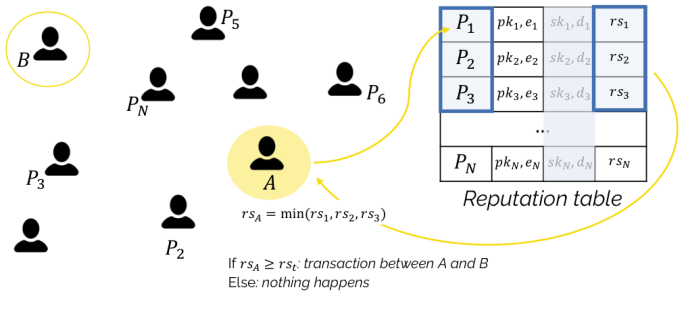

Fig. 1: Overview of the system model

b) Assumptions: We assume that $\mathcal{P}$ is a fixed group that defines a static distributed system. As such, all members are aware of the exact list of the N entities in $\mathcal{P}$, and every P_i is directly connected to every P_j with $j \neq i$ and $j \in [\![1, N]\!]$. To this end, we assume that a public key infrastructure is already deployed: each $P_i \in \mathcal{P}$ owns two pairs of public/private keys namely (e_i, d_i) which follow the RSA construction, and (pk_i, sk_i) a dlog-based keypair. We attach to each account a default reputation score $rs_i = 0.5$. We remark that the reputation function is out of the scope of this project. Since we focus on peer-to-peer interactions, we consider that communications are partially synchronous. We consider that privacy-preserving techniques are employed at the transport layer of the OSI model (e.g., dynamic IP addresses, traffic encryption) and focus on anonymizing the data transferred so as to ensure the functioning of the distributed application.

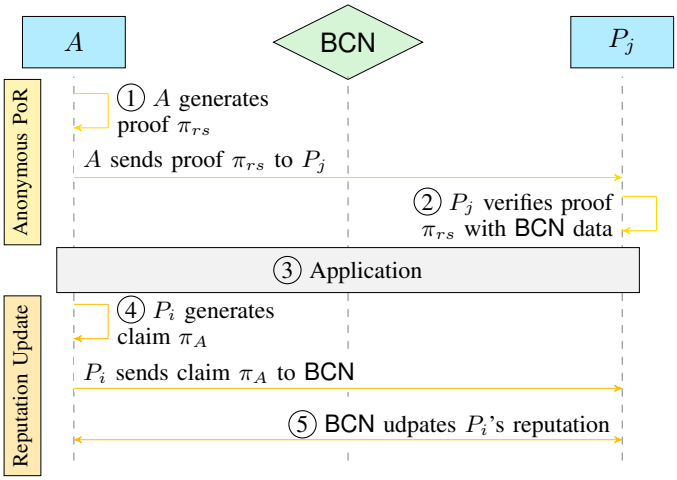

Fig. 2: Sequence diagram for the execution of Prove'n'Claim, the proposed Proof of Reputation (with BCN: Blockchain Network, PoR: proof of Reputation).

A. Proving reputation score with proofGen(·) — Step ①

When A wants to prove their reputation score rs_A is greater than a certain threshold score rs_{thr} without revealing their identity, they use the proofGen algorithm. It takes as input A's temporary address $ta \leftarrow$ accGen(1^k), a ring $\mathcal{R}$ which is a subset of size r of $\mathcal{P}$ and, an index s and the corresponding secret RSA key d_s corresponding to A (i.e. $A \equiv P_s$). As assumed, each P_i in the ring comes with their one-way trapdoor permutations $g_i(\cdot)$, and a pair of public/private RSA keys (e_i, d_i). The function starts by deriving ta to get the corresponding temporary dlog-based keypairs (tpk, tsk). Then, it computes a symmetric key as the hash value of the public temporary address ta with the H(·) hash function, and randomly selects a glue value v of size b which defines the Ring Equation. Then, for all members of the ring, it randomly selects $(r-1)$ x_i and computes their images y_i after applying the corresponding one-way trapdoor permutations $g_i(\cdot)$. Next, it solves the Ring Equation, i.e. it finds y_s such as $C_{k,v}(y_1, \ldots, y_s, \ldots, y_r) == v$ holds, and loops until it finds a correct value. By using the trapdoor d_s, it then inverts y_s as x_s. The algorithm finally returns the proof of reputation as π which contains the ring $\mathcal{R}$, the glue value

v, and the values $(x_1, \ldots, x_r)$. It also joins a signature of the proof to attach it to the entity that generated it. Lastly, the couple $\langle \pi, \sigma \rangle$ is sent from A to B.

Algorithm 3: proofGen(1^k,$params$) — *Step* ①

Input: $params$ contains an address ta, a ring $\mathcal{R} = \{P_1, \ldots, P_r\} \subseteq \mathcal{P}$ with corresponding one-way trapdoor permutations $g_i(\cdot)$, an index $s \in [\![1, r]\!]$, a secret d_s
Output: the proof $\pi_p = (\mathcal{R}; v; x_1, \ldots, x_r)$, a signature σ_p

Function *Main*:
- $(tpk, tsk) \leftarrow$ derive(ta)
- $k \leftarrow \mathsf{H}(ta)$ `// Symmetric key`
- $v \xleftarrow{\$} \{0,1\}^b$ `// The glue`
- **for** $i \in [\![1, r]\!]$ **do**
  - $x_i \xleftarrow{\$} \{0,1\}^b$; $y_i \leftarrow g_i(x_i)$
- ring$= (C_{k,v}(y_1, ..., y_r) == v)$
- **while** *!ring* **do**
  - iterate y_s `// Solving the ring equation`
- $x_s \leftarrow g_s^{-1}(y_s)$ `// Invert` y_s `with` d_s
- $\pi_p \leftarrow (P_1, \ldots, P_r; v; x_1, \ldots, x_r)$; $\sigma_p \leftarrow \mathsf{Sig}(\pi_p, tsk)$ `// Outputs`
- **return** $\langle \pi_p, \sigma_p \rangle$

B. Verifying proof of reputation with proofVerify(·) — Step ②

When B receives $\langle \pi, \sigma \rangle$ from A, they will use the proofVerify algorithm. The function takes as input the couple and returns a decision under the form of a boolean acc (for "accept") if the proof is correct, or rej (for "reject") if the proof is incorrect. To decide, the function first determines whether the signature is valid. If not, it does not process the remaining checks. Then, it evaluates the ring equation and asserts its validity. If it is correct, the algorithm returns the acc$(= True)$ boolean value.

Algorithm 4: proofVerify($\langle \pi_p$,$\sigma_p \rangle$) — *Step* ②

Input: the proof $\pi_p = (\mathcal{R}; v; x_1, \ldots, x_r)$, a signature σ_p
Output: a boolean decision acc/rej

Function *Main*:
- $res \leftarrow$rej
- **if** ($Verify(\sigma_p, tpk) == True$) **then**
  - **for** $i \in [\![1, r]\!]$ **do**
    - $y_i \leftarrow g_i(x_i)$ `// Retrieve` y_i `values`
  - $k \leftarrow \mathsf{H}(ta)$ `// Retrieve symmetric key`
  - **if** $(C_{k,v}(y_1, ..., y_r) == v)$ **then**
    - $res \leftarrow$acc `// Check ring equation`
- **return** res

These two algorithms enable A to prove to B that they know a secret key associated to a user P_j in the ring $\mathcal{R}$ without disclosing their identity. As such, A proves in a privacy-preserving manner (as will be shown in Section V) that their account has a reputation score rs_A greater than a threshold value $rs_{thr} = min_{i=1..r}(rs_i)$ where rs_i is the reputation score of P_i.

C. Claiming reputation with claimGen(·) — Step ④

The two first algorithms, namely proofGen and proofVerify, are used before executing a (set of) transaction(s) in a reputation-based application. A gives to B a privacy-preserving proof of (minimum) reputation such that B is not able to tell which P_i in the ring $\mathcal{R}$ actually generates the proof, yet they can trust A and the transferred data based on this score rs_A.

The claimGen algorithm has been designed to enable an entity P_s in the system to claim its reputation score after a (set of) transaction(s), in other words to update its reputation score. The algorithm works as follows. It takes as inputs the generator g (defined by the underlying dlog-based cryptosystem), P_s's temporary public key tpk that was used for the r-anonymous transactions, P_s's known public key pk_s, a symmetric key k made of both sk_s and tsk by $k = pk_s^{tsk} = tpk^{sk_s} = g^{sk_s \cdot tsk}$, and the tuple proof/signature $\langle \pi, \sigma \rangle$. The algorithm computes the transient t_1 and t_2 values, uses the $\mathsf{H}(\cdot)$ hash function on public inputs along with transient markers t_1, t_2. The proof π shows that P_i knows the secret key associated to the formerly r-anonymous ta address. The algorithm attaches a signature to it in order to preserve its integrity and authenticity.

Algorithm 5: claimGen(g,tpk,pk_s,k,tsk) — *Step* ④

Input: the generator g, the public key of the proving entity pk_s, the temporary public key used by P_s to remaain anonymous tpk, a symmetric key $k = pk_s^{tsk} = tpk^{sk_s} = g^{sk_s \cdot tsk}$, and tsk the secret key corresponding to tpk
Output: the claim π_c, and a signature σ_c

Function *Main*:
- $w \xleftarrow{\$} \mathbb{Z}_q$; $t_1 \leftarrow g^w$; $t_2 \leftarrow pk_s^w$
- $c \leftarrow \mathsf{H}(g, tpk, pk_s, k, t_1, t_2)$
- $r \leftarrow w - tsk \cdot c \mod q$
- $\pi_c \leftarrow \langle c, r \rangle$; $\sigma \leftarrow \mathsf{Sig}(\pi_c, sk_s)$
- **return** $\langle \pi_c, \sigma_c \rangle$

D. Verifying claim with claimVerify(·) — Step ⑤

The claim is generated with the claimGen algorithm and automatically verified by the claimVerify function. The function checks the validity of both the signature and the proof, and upon validation, updates P_i's reputation score rs_i and broadcasts the result.

Algorithm 6: claimVerify(g,tpk,pk_s,k,$\langle \pi_c, \sigma_c \rangle$) — *Step* ⑤

Input: the generator g, the public key of the proving entity pk_s, the temporary public key used by P_s to remaain anonymous tpk, a symmetric key $k = pk_s^{tsk}$, and the proof $\langle \pi_c, \sigma_c \rangle$
Output: a boolean value valid/invalid

Function *Main*:
- $res \leftarrow$ invalid
- **if** ($Verify(\sigma_c, pk_s) == True$) **then**
  - $\langle r, c \rangle \leftarrow$ parse(π_c)
  - $t_1' \leftarrow g^r \cdot tpk^c$; $t_2' \leftarrow pk_s^r \cdot k^c$
  - $\tilde{c} \leftarrow \mathsf{H}(g, tpk, pk_s, k, t_1', t_2')$
  - **if** $c == \tilde{c}$ **then**
    - $res \leftarrow$valid `// i.e.` $t_1' == t_1$ `and` $t_2' == t_2$
- **return** res

V. Security Analysis

In this section, we demonstrate that the proposed Prove'n'Claim scheme for reputation-based applications com-

ply with the three following security properties: *correctness*, $k-$*anonymity*, and *reputation unforgeability*. We first define these properties in the context of the current study, then we detail the corresponding proofs.

A. Security properties

Definition 3. (Correctness)
The correctness property is twofold: first, it asks that the proof verification algorithm proofVerify *successfully verifies the proof legitimately generated by the proof generation algorithm* proofGen*; then, it requires that the claim verification algorithm* claimVerify *successfully verifies the claim legitimately generated by the claim generation algorithm* claimGen*. This is summarized by Equation 4:*

$$\begin{aligned}&(\textit{proofVerify}(\textit{proofGen}(1^k, params)) = 1)\\ &\quad \wedge (\textit{claimVerify}(pp, \textit{claimGen}(pp, tsk)) = 1)\end{aligned} \tag{4}$$

with $pp = g, tpk, pk_s, k$ the public parameters.

Theorem 1. ***(Correctness of Prove'n'Claim)***
Under the assumption that a secure $k-$anonymous existentially unforgeable under chosen message attacks ring signature scheme and a sound zero-knowledge proofs technique for dlog-based cryptosystems exist, the Prove'n'Claim *system is correct.*

Argument. *This condition boils down to:*

$$(\textit{VSig}(\textit{RSig}(m, R, tsk)) = 1) \wedge (\textit{V}(\textit{P}(pp, tsk)) = 1) \tag{5}$$

Yet, since both the underlying ring signature scheme and ZKP are correct, the condition is verified. □

Definition 4. *(Reputation Unforgeability) The reputation unforgeability property of the* Prove'n'Claim *algorithm is twofold: first, it ensures that no adversary $\mathcal{A}$ can convince a honest user of having a reputation score rs greater than the reputation score attached to their public identity; second, it sets that no adversary $\mathcal{A}$ can claim transactions that they did not actually generated (i.e. they cannot convince a honest user that they know the secret key tsk associated to the transacting account ta without actually knowing tsk). This is characterized by a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ that we thoroughly describe below.*

The game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, on which we elaborate our discussion on the reputation unforgeability property of the proposed Prove'n'Claim mechanism, goes as follows:

- $\mathcal{C}$ generates the group $\mathcal{P} = \{P_1, \ldots, P_n\}$: $(pk_i, sk_i, e_i, d_i) \xleftarrow{\$} \mathsf{accountGen}(1^k)$. In addition, $\mathcal{C}$ generates n temporary accounts $ta_j \xleftarrow{\$} \mathsf{accountGen}(1^k)$ for $i = 1..n$, meaning there is exactly one ta_j per P_i. Then, $\mathcal{C}$ gives the set of public keys $\{(pk_i, e_i)\}_{i=1..n}$ and the set of temporary accounts' public keys $\{(tpk_i)\}_{i=1..n}$ to $\mathcal{A}$. Finally, it sets $S_{proofs} \leftarrow \varnothing$ and $S_{claims} \leftarrow \varnothing$.
- $\mathcal{A}$ is allowed to make proof queries of the form (R, j), where R is a ring of public keys, $j \in [\![1, n]\!]$ an index such as $P_j \in R$. In this context, the message to sign $m \in \mathcal{M}$ is equal to tsk_j. When $\mathcal{C}$ receives (R, j), it computes $\pi_{P,j} \leftarrow \mathsf{proofGen}(1^k, R, g_s, s, d_s)$. It sends back the proof $\pi_{P,j}$ to A and appends (R, j) to S_{proofs}. In parallel, $\mathcal{A}$ is allowed to make claim queries of the form (i, tpk_i). Upon reception of the query, $\mathcal{C}$ computes $\pi_{C,i} \leftarrow \mathsf{claimGen}(g, tpk_i, pk_i, k_{ii}, tsk_i)$. It sends back $\pi_{C,i}$ to $\mathcal{A}$ and adds i to S_{claims}.
- $\mathcal{A}$ outputs (1) $o_1 \leftarrow \langle \pi_{P,k_1}, \sigma_{P,k_1} \rangle$ and (2) $o_2 \leftarrow langle \pi_{C,k_2}, \sigma_{C,k_2} \rangle$ with k_1 and k_2 two integer values such as (1) $P_{k_1} \in \pi_{P,k_1}.\mathcal{R}$ and $(\pi_{P,k_1}.\mathcal{R}, k_1) \notin S_{proofs}$, and (2) $k_2 \notin S_{claims}$.

We say that the Prove'n'Claim mechanism respects the reputation unforgeability property $\iff$

$$\begin{aligned}&\Pr[\mathsf{proofVerify}(o_1) == 1] \ < \mathrm{negl}(k)\\ &\quad \wedge \Pr[\mathsf{claimVerify}(o_2) == 1] \ < \mathrm{negl}(k)\end{aligned} \tag{6}$$

with $\mathrm{negl}(k)$ the negligible function in k the security parameter.

Theorem 2. ***(Reputation unforgeability of Prove'n'Claim)***
Under the assumption that a secure $k-$anonymous existentially unforgeable under chosen message attacks ring signature scheme and a sound zero-knowledge proofs technique for dlog-based cryptosystems exist, the Prove'n'Claim *system respects the Reputation Unforgeability property.*

Argument. *The first half of the condition unfolds from the existential unforgeability of the chosen ring signature scheme; while the second half is given by the soundness of the selected zero knowledge proofs technique.* □

Definition 5. ($k-$anonymity)
The $k-$anonymity property of the Prove'n'Claim *algorithm is characterized by the $k-$anonymity of the communicating node during phases ① to ③. This is characterized by a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ that we thoroughly describe below.*

The game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, on which we elaborate our discussion on the $k-$anonymity property of the proposed Prove'n'Claim mechanism, goes as follows:

1) $\mathcal{C}$ generates the group $\mathcal{P} = \{P_1, \ldots, P_n\}$: $(pk_i, sk_i, e_i, d_i) \xleftarrow{\$} \mathsf{accountGen}(1^k)$. In addition, $\mathcal{C}$ generates n temporary accounts $ta_j \xleftarrow{\$} \mathsf{accountGen}(1^k)$ for $i = 1..n$, meaning there is exactly one ta_j per P_i. Then, $\mathcal{C}$ computes the n proofs of reputation, one for each ta_j (i.e. P_i) as $\langle \pi_j, \sigma_j \rangle \leftarrow \mathsf{proofGen}(1^k, params)$ with $params = (\mathcal{P}, g_j(\cdot), j, d_j)$ as defined in Section IV. Finally, $\mathcal{C}$ gives the set of public keys $\{(pk_i, e_i)\}_{i=1..n}$ and the set of proofs of reputation $\{\langle \pi_j, \sigma_j \rangle\}_{j=1..n}$ to $\mathcal{A}$.
2) $\mathcal{A}$ requests a challenge to $\mathcal{C}$ by sending a tuple (i_0, i_1, R^*), where i_0 and i_1 are indices such that $P_{i_0} \in$

R^* and $P_{i_1} \in R^*$. Then, $\mathcal{C}$ samples a challenge bit $b \xleftarrow{\$} \{0,1\}$, computes $\langle \pi^*, \sigma^* \rangle \leftarrow$ proofGen$(1^k, params)$, and gives the resulting proof to $\mathcal{A}$.

3) $\mathcal{A}$ outputs a bit $b^{'} \in \{0,1\}$.

We say that the Prove'n'Claim mechanism respects the $k-$anonymity property $\iff$

$$|\Pr[b == b'] - \frac{1}{2}| < \text{negl}(k) \tag{7}$$

with negl(k) the negligible function in k the security parameter.

Theorem 3. *($k-$**anonymity of Prove'n'Claim**)*
Under the assumption that a secure $k-$anonymous existentially unforgeable under chosen message attacks ring signature scheme exists, the Prove'n'Claim system is $k-$anonymous.

Proof. The demonstration directly unfolds from the construction of Prove'n'Claim which leverages the ring signature scheme defined in Section III. □

VI. Simulations

To highlight the feasibility and practicability of our approach, we simulate the execution of our four main functions proofGen$(\cdot)$, proofVerify$(\cdot)$, claimGen$(\cdot)$, claimVerify$(\cdot)$ so as to analyze their behaviour when the ring size increases. The testing bench is available at https://github.com/mdehezcl/PNC.git.

The simulation bench is implemented in Python 3.10 and testing is performed on a MacBook pro 15 with a Core i7 processor (2.5 GHz - 1 To HDD + SSD - RAM 16Go). For all the experiments below, the network is composed of 100 distinct identities and we measured the time to execute the proposed functions (proofGen, proofVerify, claimGen and claimVerify) as well as the size of the output objects (PoR and claims) while varying the size r of the ring from 5 to 100.

A. Execution time to generate/verify the PoR

Figure 3 provides the measured execution time for the generation (blue) and verification (red) of our anonymous proof-of-reputation. We first observe that the generation of the proof is longer than its verification. Then, the polynomial approximation shows that both plots evolve linearly according to the size of the ring. We can approximate the execution time per ring size for proofGen (resp. proofVerify) with the function $0.015x + 2.803$ (resp. $0.015x + 2.097$). For a ring of 100 peers (hence providing $100-$anonymity), the generation of the proof is performed in less than 5 milliseconds and its verification in about 3.5 milliseconds.

B. Execution time to generate/verify the Claim

Figure 4 provides the measured execution time for the generation (blue) and verification (red) of our claims of reputation. We first observe that, unlike previously, the generation of the claim is faster than its verification by a factor 2.3. Then, thanks to the approximation polynomials (respectively $0.001x + 2.681$ and $0.002x + 6.329$), we see that the execution time of claimGen and claimVerify is constant and independent of the size of the ring.

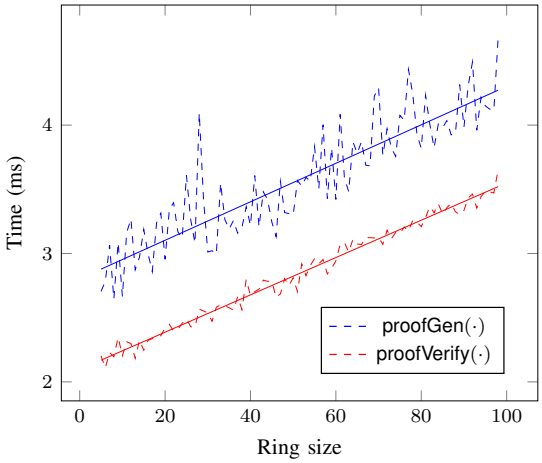

Fig. 3: Execution time (in ms) to generate/verify the proof according to the size of the ring.

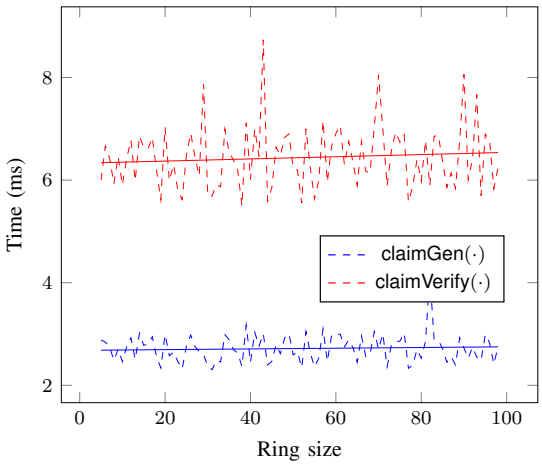

Fig. 4: Execution time (in ms) to generate/verify the claim according to the size of the ring.

C. Size of the PoR/Claim

Figure 5 provides the measured execution size of the proofs of reputation (blue) and the claims (red) according to the size r of the ring. We observe that the PoR size grows linearly in the ring size and weight $5 \cdot 10^4$ bytes, thus about 49 Kilobytes. On the contrary, we see that the claim size is not correlated to the size of the ring and has a median size equal to 39 Kilobytes approximately.

D. From a transactional perspective

TABLE I: Parameters

Hypothesis	Value
Gas per 256-bit-long word	20 000 gas [17]
Gas cost	16 - 19 gwei[1]
Equivalent USD	0.00000161 USD[2]

In Table I, we recall essential parameters to discuss the applicability of Prove'n'Claim in the context of blockchains

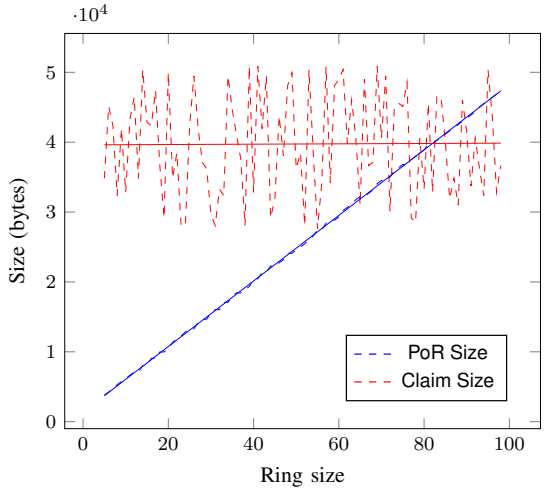

Fig. 5: Size (in bytes) of the proof of reputation and the claim according to the size of the ring.

TABLE II: Analysis

Object	Size (KB)	Gas cost ($\cdot 10^3$)	Gwei equivalent ($\cdot 10^3$)	Final USD cost [18]
Maximum PoR size	49	31 360	501 760 - 595 840	807 - 959
Average Claim size	39	24 960	399 360 - 474 240	642 - 763

transactions. The table lists the amount of gas spent per 256-bit-long words, the cost of one unit of gas in Gwei ([1]from etherscan.io/gastracker), and the USD equivalent value ([2]from cryps.info). Then, in Table II, we draw a short analysis of the cost of each object, namely the anonymous PoR and the claim objects. The final USD cost acts as a deterrent in the sense that it reminds us the non-feasability to store all the data inside a smart contract. Yet, there are two areas of improvements: the first consists in determining which kind of data should absolutely be stored inside the smart contract such as to provide security and robustness to any overlaid application. The second axis consists in optimizing the proposed algorithms by adding signature aggregation and thus, proof aggregation.

VII. Applications

We envision two main ways to apply the presented Prove'n'Claim mechanism. In the first case, the scheme can be used per se to improve privacy in reputation-based crowdsourcing systems. The second case is the design of unpredictable and privacy-preserving reputation based consensus algorithms, example which opens a greater field of research and necessitates further investigations.

A. Improving privacy in reputation-based crowdsourcing systems

The presented Prove'n'Claim mechanism can notably be used per se in all applications that leverage reputation as a determining metric to take decisions on. As quoted in introduction, crowdsourced data collection systems are an example of such applications. This data collection mechanism is widely use in distributed systems and consists in retrieving data from individual devices or processes to get a global picture of the state of a decentralized system. It is particularly prevalent in industrial settings to improve the workload repartition and overall efficiency of an indutrial system [19] or to reduce the CO2 emission of smart building [20], in combinaison with IoT-based distributed systems to implementation indoor localization systems [21] usable in dangerous settings, or even in Vehicular Ad-hoc NETworks to improve disaster management mechanisms [22]. Yet, these systems traditionally leverage reputation as a metric of data confidence. And reputation-based mechanisms yield several limitations including the lack of anonymity since reputation is often attached to an identity. This implies the vulnerability to coercion as all actors are identifiable and prone to corruption and facilitates targeted attacks such as Denial of Services. Using the presented Prove'n'Claim mechanism solves these limitations in the following way:

- Participants are anonymous when they issue transactions or when they query a service;
- Decision making on collected data is still very quick as based on a reputation score attached to their anonymous identity;
- Good behaviors of users are easily traceable and rewardable as they can claim reputation points based of their behavioral history.

B. Design of an unpredictable and privacy-preserving reputation based consensus algorithm

The presented Prove'n'Claim mechanism can also be used to design unpredictable and privacy-preserving reputation based consensus algorithms for permissioned blockchain settings. It the following paragraphs, we decline the assumptions that can be taken and the future issues we plan to tackle to this end.

a) Realistic assumptions:

- All nodes in the P2P network are users and miners *i.e.* they can generate and broadcast transactions, and participate to the consensus protocol that validates the block made with these transactions. This requirement is easily reachable as long as the nodes have an Internet access.
- The initial state of the system consists in N identified miners, each one initialized with the cryptographic parameters in accordance with the Prove'n'Claim requirements. This requirement can be achieved during the deployement of the blockchain or on top on it.
- In order to be authorized to participate in the system, each blockchain node $P_i \in \mathcal{P}$ must sign and commit to their RSA keypair. Thus, we assume that all N initial nodes did so in an initialization transaction. This is alike registrations to an application.

b) Future work: In order to design a consensus algorithm, one must define several aspects: the access control to

the network/application, the miner group selection, the leader election, the reputation update and miner revocation/expulsion as precised on Figure 6. The Prove'n'Claim mechanism would be particularly useful to define the rules of transaction validation, leader selection and block production.

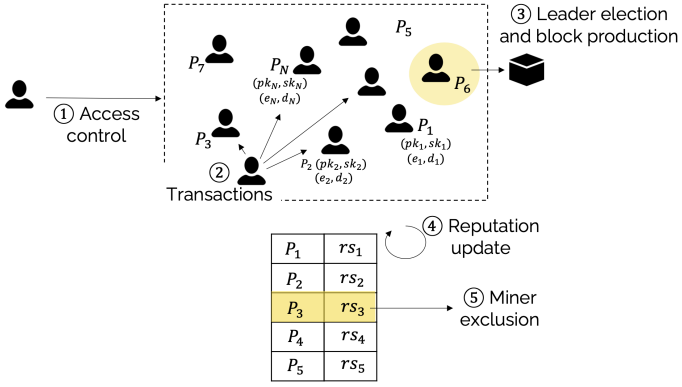

Fig. 6: Overview of the system model for the consensus

VIII. Conclusion

The Prove'n'Claim algorithm proposed in this paper enables blockchain users to generate privacy-preserving proofs of reputation. we have demonstrated that the construction is theoretically secure, correct, unforgeable, and $k-$anonymous based on the security model described.

While we have briefly discussed the performance, both in terms of computational cost and communication overhead, of the proposed algorithms, further discussion on the linearity of the execution time could be included in future work to validate the approximations for any ring size. In addition, a comparison with state-of-the-art approaches, for instance [10], could be made to further evaluate the proposed algorithm. However, this was not the focus of this paper, and we have instead focused on demonstrating the security and feasibility of our approach. Finally, there may be threats to the validity of the simulation results due to the assumptions made in the simulation, such as the network model, node behavior, and attacker model. The performance results were obtained using simulations and may differ from actual implementation results. Nevertheless, we have demonstrated the effectiveness of the proposed algorithm in achieving privacy-preserving proofs of reputation, which is a promising step towards designing unpredictable consensus algorithms.

In future work, we aim to further explore the application of Prove'n'Claim in the design of unpredictable consensus algorithms for permissioned and permissionless blockchains. Additionally, we plan to investigate the scalability and assess the practical feasibility of our approach in larger networks.

Acknowledgment

We would like to express our gratitude to B4IoT (Région Nouvelle-Aquitaine), FEDER MISMAR, and ISAE-SUPAERO for their support in conducting this research.

References

[1] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '03. New York, NY, USA: ACM, 2003, p. 144–152.

[2] A. Selcuk, E. Uzun, and M. Pariente, "A reputation-based trust management system for p2p networks," in *IEEE International Symposium on Cluster Computing and the Grid, 2004.*, 2004, pp. 251–258.

[3] J. Bou Abdo, R. El Sibai, and J. Demerjian, "Permissionless proof-of-reputation-x: A hybrid reputation-based consensus algorithm for permissionless blockchains," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4148, 2021.

[4] V. Gramoli, "From blockchain consensus back to byzantine consensus," *Future Generation Computer Systems*, vol. 107, pp. 760–769, 2020.

[5] A. Gouget, J. Patarin, and A. Toulemonde, "Unpredictability properties in algorand consensus protocol," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021, pp. 1–3.

[6] J. Bonneau, E. W. Felten, S. Goldfeder, J. A. Kroll, and A. Narayanan, "Why buy when you can rent? bribery attacks on bitcoin consensus," 2016.

[7] F. Gai, B. Wang, W. Deng, and W. Peng, "Proof of reputation: A reputation-based consensus protocol for peer-to-peer network," in *International Conference on Database Systems for Advanced Applications*. Springer, 2018, pp. 666–681.

[8] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, "Proof of reputation: A reputation-based consensus protocol for blockchain based systems," in *Proceedings of the 2019 International Electronics Communication Conference*. New York, NY, USA: ACM, 2019, p. 131–138.

[9] W. Cai, W. Jiang, K. Xie, Y. Zhu, Y. Liu, and T. Shen, "Dynamic reputation–based consensus mechanism: Real-time transactions for energy blockchain," *International Journal of Distributed Sensor Networks*, vol. 16, no. 3, p. 1550147720907335, 2020.

[10] M. T. de Oliveira, L. H. Reis, D. S. Medeiros, R. C. Carrano, S. D. Olabarriaga, and D. M. Mattos, "Blockchain reputation-based consensus: A scalable and resilient mechanism for distributed mistrusting applications," *Computer Networks*, vol. 179, p. 107367, 2020.

[11] I. B. Damgård, "A design principle for hash functions," in *Proceedings on Advances in Cryptology*, ser. CRYPTO '89. Berlin, Heidelberg: Springer-Verlag, 1989, p. 416–427.

[12] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 1993, p. 62–73.

[13] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 552–565.

[14] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," *Technical Report/ETH Zurich, Department of Computer Science*, vol. 260, 1997.

[15] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, "Ethdkg: Distributed key generation with ethereum smart contracts," *Cryptology ePrint Archive*, 2019.

[16] D. Miyahara, T. Sasaki, T. Mizuki, and H. Sone, "Card-based physical zero-knowledge proof for kakuro," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E102A, no. 9, pp. 1072–1078, 2019.

[17] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, pp. 1–32, 2014.

[18] "Eth gas station," last accessed: 4 August 2022.

[19] V. Pilloni, "How data will transform industrial processes: Crowdsensing, crowdsourcing and big data as pillars of industry 4.0," *Future Internet*, vol. 10, no. 3, p. 24, 2018.

[20] M. Attia, N. Haidar, S. M. Senouci, and E.-H. Aglzim, "Towards an efficient energy management to reduce co 2 emissions and billing cost in smart buildings," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2018, pp. 1–6.

[21] B. Lashkari, J. Rezazadeh, R. Farahbakhsh, and K. Sandrasegaran, "Crowdsourcing and sensing for indoor localization in iot: A review," *IEEE Sensors Journal*, vol. 19, no. 7, pp. 2408–2434, 2019.

[22] M.-A. Lèbre, F. Le Mouël, and E. Ménard, "Efficient vehicular crowdsourcing models in vanet for disaster management," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–5.