Proof-of-Prestige: A Useful Work Reward System for Unverifiable Tasks

MICHAŁ KRÓL, City, University of London
ALBERTO SONNINO, MUSTAFA AL-BASSAM, and ARGYRIOS G. TASIOPOULOS, University College London
ETIENNE RIVIÈRE, UCLouvain
IOANNIS PSARAS, University College London

As cryptographic tokens and altcoins are increasingly being built to serve as utility tokens, the notion of useful work consensus protocols is becoming ever more important. With useful work consensus protocols, users get rewards after they have carried out some specific tasks useful for the network. While in some cases the proof of some utility or service can be provided, the majority of tasks are impossible to verify reliably. To deal with such cases, we design "Proof-of-Prestige" (PoP)—a reward system that can run directly on Proof-of-Stake (PoS) blockchains or as a smart contract on top of Proof-of-Work (PoW) blockchains. PoP introduces "prestige," which is a volatile resource that, in contrast to coins, regenerates over time. Prestige can be gained by performing useful work, spent when benefiting from services, and directly translates to users minting power. Our scheme allows us to reliably reward decentralized workers while keeping the system free for the end-users. PoP is resistant against Sybil and collusion attacks and can be used with a vast range of unverifiable tasks. We build a simulator to assess the cryptoeconomic behavior of the system and deploy a full prototype of a content dissemination platform rewarding its participants. We implement the blockchain component on both Ethereum (PoW) and Cosmos (PoS), provide a mobile application, and connect it with our scheme with a negligible memory footprint. Finally, we adapt a fair exchange protocol allowing us to atomically exchange files for rewards also in scenarios where not all the parties have Internet connectivity. Our evaluation shows that even for large Ethereum traces, PoP introduces sub-millisecond computational overhead for miners in Cosmos and less than 0.013$ smart contract invocation cost for users in Ethereum.

CCS Concepts: • **Security and privacy** → **Distributed systems security**; • **Networks** → *Network services*;

Additional Key Words and Phrases: Blockchain, security and privacy, network security

Michał Król and Etienne Rivière Riviere are supported by the Belgian *Fonds de la Recherche Scientifique* (FNRS) under Grant #F452819F, Alberto Sonnino is supported by the EU H2020 DECODE project under grant agreement number 732546, Mustafa Al-Bassam is supported by a scholarship from The Alan Turing Institute, and Ioannis Psaras is supported by the EPSRC INSP Early Career Fellowship under grant agreement number EP/M003787/1.

Authors' addresses: M. Król, City, University of London; email: michal.krol@city.ac.uk; A. Sonnino, Facebook Calibra; email: asonnino@fb.com; M. Al-Bassam, LazyLedger; email: mustafa@lazyledger.io; A. G. Tasiopoulos, University College London; email: a.tasiopoulos@ucl.ac.uk; E. Rivière, UCLouvain; Ioannis Psaras; email: etienne.riviere@uclouvain.be; I. Psaras, University College London; email: i.psaras@ucl.ac.uk.

1533-5399/2021/06-ART44 $15.00
https://doi.org/10.1145/3419483

ACM Reference format:
Michał Król, Alberto Sonnino, Mustafa Al-Bassam, Argyrios G. Tasiopoulos, Etienne Rivière, and Ioannis Psaras. 2021. Proof-of-Prestige: A Useful Work Reward System for Unverifiable Tasks. *ACM Trans. Internet Technol.* 21, 2, Article 44 (June 2021), 27 pages.
https://doi.org/10.1145/3419483

1 INTRODUCTION

Following the recent success of Bitcoin [65], a plethora of cryptocurrencies has experienced an increase in popularity [16]. There are over 1,900 cryptocurrencies one can invest in with the total market cap exceeding 260B USD.[1] Multiple cryptocurrencies (i.e., Bitcoin and Ethereum) are secured using Proof-of-Work (PoW) protocols, initially introduced by Bitcoin, binding users' mining power to a physical resource. In PoW, miners try to solve difficult cryptographic puzzles. Once a miner finds a solution to the puzzles, they gain the right to append a new block to the blockchain, discover (mine) new coins and collect fees from transactions included in the block. PoW incentivizes users to commit resources to secure the transactions and binds users' mining power to a physical resource (i.e., CPU power), thus preventing the generation of multiple fake identities by one party (Sybil attack).

While PoW proved to be able to secure a global payment system, it has several drawbacks, the first one being an inefficient use of resources. Bitcoin's PoW consumes huge and growing amounts of electricity surpassing the electricity consumption of entire countries.[2] Proof-of-Stake (PoS) was recently proposed to deal with this problem [17, 29, 52]. With PoS, the power of each miner, and thus their probability of mining/minting a new block, is proportional to their stake—the number of coins they own. The winner can, therefore, be chosen using a variation of the weighted round-robin algorithm without wasting resources on solving cryptographic puzzles [29]. While being much more resource saving, PoS allows rich users to collect the majority of the fees and discover coins making them even richer, increasing inequality in the system.

A recent trend sees cryptocurrencies as an incentive method for users to perform useful work and create a shared economy environment. For instance, Filecoin [24, 56] rewards miners for renting storage capacity, while Golem [2] allows rental of processing power and to perform a client's computation-heavy tasks. The vision is to create a decentralised system where miners are incentivized to do useful work, secure the transactions, and automatically receive rewards when tasks are completed.

In the classic setup, useful work can be performed by a *"contributor"* for a *"beneficiary."* The beneficiary submits a task and a reward to the blockchain that is used to assure payment for service [9, 53]. When the contributor correctly completes the requested task, the payment is unlocked. However, currently, multiple cloud platforms do not expect their users to pay for the services (i.e., Facebook, Youtube). As a result, to attract more users, blockchain-based platforms must keep this feature as well. It means involving a third party in the system to whom we refer to as a *"motivator."* The motivator benefits from increasing the size and popularity of the network (i.e., by running advertisements) and rewards contributors' useful work, while keeping the platform free for the end-users. For instance, on Steem [8] authors (contributors) create content for readers (beneficiaries). Readers can then signal interesting content using a "vote-up" button. Steem (motivator), which is primarily interested in increased viewership to increase its income from advertisements, rewards the most successful authors with an automatic coin transfer.

[1]https://www.investing.com/crypto/currencies.
[2]https://digiconomist.net/bitcoin-energy-consumption.

Such a system presents multiple benefits. Beneficiaries do not have to pay for the content, and the system remains open for any contributor to join, perform useful work, and be paid according to their performance and contribution; the motivator, however, benefits from an open platform avoiding costly contracts and a contributor selection process. However, to reliably reward contributors, the network must be able to automatically verify their involvement. While completion of some tasks can be proven to a third party (e.g., file storage [56]), in many cases this is impossible. For instance, it is not possible to prove that a file has been successfully transferred between any two untrusted nodes. In such cases, the motivator relies only on beneficiary acknowledgments to reward contributors and can be thus susceptible to Sybil attacks: To maximize their reward, contributors can create multiple fake identities claiming usefulness of their work. Moreover, even with access restriction techniques or voting power bounded to the stake, users can collude and cross-acknowledge their potentially non-existing work.

We present Proof-of-Prestige (PoP), which aims to be a building block of cryptocurrencies based on useful work. Our system allows motivators to reliably reward untrusted, decentralized contributors without introducing monetary fees for the end-users. PoP builds on PoS, but instead of binding user minting power to stake, the probability of minting a new block is determined by users' prestige. One can generate *prestige* associated with each identity in the system from money or by performing useful work; *prestige* is much more volatile and renewable resource than virtual currencies. In PoP, beneficiaries pay for services by transferring prestige to contributors, while avoiding spending coins. A task is considered as completed when acknowledged by its beneficiary. PoP can thus be used in a wide variety of use-cases, securing the system against Colluding and Sybil attacks and avoiding artificial inflation of miners' power by completing dummy tasks. We also introduce the concept of domains the allow motivators to securely reward users for performing specific tasks in contrast to general rewards divided among all the users. PoP is not meant to be a new consensus protocol but rather a new system of assigning minting power to users that is compatible with multiple already existing PoS protocols.

PoP uses efficient composite signatures, allowing us to report multiple tasks at a time without increasing the amount of information submitted to the blockchain. Furthermore, we introduce a fair exchange protocol [32], making sure that workers[3] receive rewards if and only if the work was completed correctly (Section 7). The scheme eliminates the problem of free-riders and is adapted to work with PoP acknowledgments and allow reliably exchange content in a P2P manner even when both parties do not have Internet connectivity.

We build a discrete event simulator and prove mathematical properties of our scheme in large-scale experiments. Furthermore, we build, deploy, and evaluate a full content exchange platform using PoP to distribute rewards. We implement the blockchain module for Ethereum [76] (as a smart contract) and for Cosmos [26] (natively). The prototype includes a data dissemination application for Android phones that is connected to the both blockchain implementations. We prove that the mobile application requires only few kB of memory to store its private keys and is able to sign complex transactions within 40 ms. For the blockchain part, the experiments show less than 0.013$ cost of smart contract operations in Ethereum and sub-millisecond execution time overhead for Cosmos miners. The application and the platform are available for download and public use.

Outline. We first define our model and design goals in Section 2. We present an overview of the design of PoP in Section 3. We present its detailed construction in Section 4, followed by economics discussion in Section 5 and use-cases in Section 6. We address the problem of fair exchange in Section 7. Our evaluation is given in Section 8 for the simulations and in Section 9 for the prototype. We review related work in Section 10 and conclude in Section 11.

[3]For fair exchange we focus on a content dissemination scenario described in Section 6.

2 THREAT MODEL AND GOALS

We assume the following actors in our system:

- **Beneficiaries:** End-users of the system who wish to consume services.
- **Contributors:** Nodes that perform tasks for beneficiaries on behalf of motivators.
- **Motivators:** System operators or users that submit tasks to the network and reward contributors for their useful work.

Each user is represented by an identity and can act as contributor, beneficiary, and/or motivator. We assume presence of malicious attackers and that none of the actors can be trusted; i.e., they may attempt to steal funds, avoid making payments, create fake transfers, and create fake identities. At any given time, each party may drop, send, record, modify, and replay arbitrary messages in the protocol. In particular, users can create multiple fake identities and record interactions between them or collude with other users trying to maximize their rewards.

In PoP, users have two values associated with their accounts:

- **Coins** - similar to other cryptocurrencies (such as Bitcoin or Ether) or ERC20 tokens [1]. Coins can be directly sent to or received by other users via transactions, as is the case with all cryptocurrencies.
- **Prestige** - determines the probability of the user minting a new block. Prestige cannot be directly transferred between accounts but is exchanged as a reward for performing useful work.

Proof of Prestige assumes an underlying blockchain to facilitate *prestige* transfers without a trusted third party. We assume that the underlying blockchain is resistant to double-spending attacks, and guarantees liveness—that is, transactions submitted to the blockchain will be eventually processed, within bounded period of time. We assume fair exchange acknowledgment for service that is adapted to the task being performed.

Furthermore, Proof of Prestige requires a PoS consensus protocol that can assign custom weights to users, where each weight determines the probability of minting a new block (i.e., Algorand [40], Tendermint [54], Hashgraph [14], and Ouroboros [52]). As prestige is a volatile resource, it cannot be used with PoS protocols that require stakeholders to deposit and lock their coins for a long period of time (such as Casper FFG [20]), so their deposit can be reduced if they are caught misbehaving. This is because prestige cannot be locked, as by design it automatically increases or decreases. Finally, PoP can be implemented on top of a blockchain as a smart contract regardless of the underlying consensus protocol.

Proof of Prestige answers to the following design goals:

- *Open membership.* Any system user is able to participate in the system as a contributor or beneficiary without prior approval by some third party.
- *Creditless rewards.* A beneficiary can reward a contributor for completing a task without actual credit (i.e., virtual currency) but rather by increasing the chance of the contributor minting next blocks.
- *Contributor incentivization.* Contributors are economically incentivized to perform tasks.
- *Inflation control.* Given a predefined rule to determine the inflation rate, the rate of inflation of coins in the system cannot be changed by users.
- *Sybil attack resistance.* Creating multiple identities cannot increase a user's minting power without obtaining more coins.
- *Collusion attack resistance.* Users cannot increase their total amount of prestige by colluding with each others.

Table 1. Notations

Parameters			
U_i	user i	P_i	prestige of user i
C_i	coins of user i	d	decay
T_i	threshold value of user i	t	time slot t
f	negotiated fee (in prestige) for a service	f_{retain}	retained amount of prestige
x_{ij}	prestige transferred from user i to user j	v_k	verification key
sk	secret key	m	message
l	message descriptor	σ	signature
$\mathcal{D}_k$	domain k	k	encryption key
z	file encoding	h	Merkel tree hash
π	proof of misbehavior	r	Merkel Tree root hash over encoding z

3 OVERVIEW

We present an overview of operations in PoP (Figure 1). All the notations are listed in Table 1. Coins and prestige are related and can influence one another. The coins generate prestige over time, while prestige determines the probability of minting the next block, similarly to stake in PoS. *The probability of each user minting a block is thus proportional to their prestige.* Minting a new block, in turn, allows users to collect transaction fees, discover new coins, and increase their total number of coins. Prestige represents a much more volatile and renewable resource, while the number of coins does not change over time unless minted or transferred in transactions. *Prestige can be spent to benefit from services or gained when performing services for others.* A higher amount of prestige (or higher prestige growth) allows users to benefit from more services simultaneously without waiting for it to be reproduced by the possessed coins.

Motivators start by allocating rewards expressed in coins (step 1). The rewards can be allocated globally (for all the network participants) or for a specific task/domain (Section 5). Before performing a task, a worker and a beneficiary need to negotiate a fee, i.e., the amount of prestige that will be exchanged after the task is completed. We let users chose this value on their own so that it can be adapted to external factors (e.g., coins/fiat currency exchange rate) and satisfy both parties.

In the next step, the worker performs the task (step 2) and receives an acknowledgment from the beneficiary confirming the transfer of the negotiated amount of prestige (step 3). To prevent beneficiaries from walking away without paying, the exchange of task for acknowledgment should be atomic (Section 7).

Workers periodically update received acknowledgments to the blockchain (step 4), allowing the network to update prestige values of each user. The consensus protocol chooses the next miner to propose a new block taking into account users' prestige instead of their stake (step 5). The selected miner collects rewards allocated by the motivators as well as regular transaction fees and new coins (Section 5).

There is one note worth making regarding the difference of PoP to PoS as regards the ability of rich users to gain more from the system, while in PoS the number of coins (stake) a user has is the only resource that determines who mines the new block, in PoP useful work can increase a users' prestige and therefore, the probability of minting new coins, even if the user starts with a low amount of coins/prestige. Therefore, although someone who buys more coins can enter the system with higher prestige (and hence, a higher probability of minting new coins), this does not exclude users with fewer coins from building up prestige if they contribute to the system with useful work.

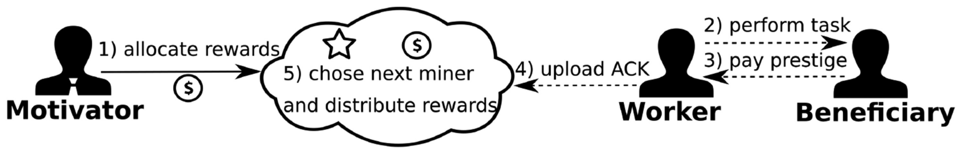

Fig. 1. PoP overview.

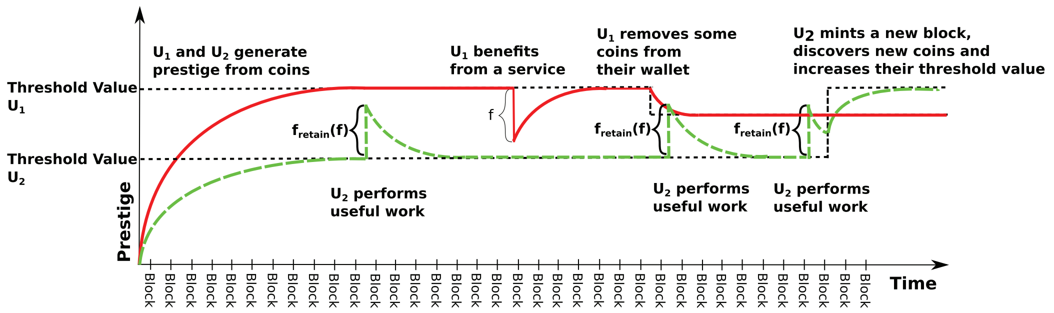

Fig. 2. Example of evolution of the prestige and threshold value with time (expressed in blocks).

4 PROOF OF PRESTIGE

When user U_i joins the system (i.e., creates their identity), he or she starts with no prestige $P_i = 0$. With each new block (mined by any user in the system) the amount of prestige of every user in the system is increased by the number of coins in his or her wallet C_i, so that richer users generate prestige at a higher rate. At the same time, to avoid prestige increasing indefinitely, we introduce a decay parameter d. The decay determines what percentage of current prestige is lost with each block. Specifically, the prestige of a user evolves over time according to a non-homogeneous, autonomous, affine, first-order Discrete Dynamical System (DDS) with prestige increment on time-slot t:

$$\delta P_i^t = C_i^t - dP_i^{t-1}, \tag{1}$$

where $t \geq 1$ and $0 < d < 1$ is a tunable system parameter. We can see that prestige on time-slot t can be written as:

$$P_i^t = \sum_{j=1}^{t} \delta P_i^j = P_i^{t-1} + \delta P_i^t = C_i + (1-d)P_i^{t-1}. \tag{2}$$

The fixed point(s) of Equation (2) DDS can be easily derived by applying simple linearisation techniques. In detail, consider that $P_i^t = g(P_i^{t-1})$; then for a candidate fixed point T_i we have that $T_i = g(T_i)$:

$$T_i = \frac{C_i}{d}. \tag{3}$$

In fact, $|g'(T_i)| < 1$, i.e., $g'(\cdot) = 1 - d < 1$, and therefore T_i is an attractor fixed point indicating the convergence of prestige DSS to T_i. The amount of prestige that a user can generate from coins is thus limited to T_i—called the *Threshold Value*—where increasing decay evens up prestige generated from coins (Figure 2). The threshold value depends only on the amount of coins in user's wallet C_i and the decay parameter d, and it therefore increases by acquiring more coins.

To increase their prestige above their Threshold Value, users have to perform useful work, confirmed by a beneficiary for whom the task was completed. When performing useful work, users instantly get more prestige (see Prestige spikes in Figure 2) and therefore increase their chances

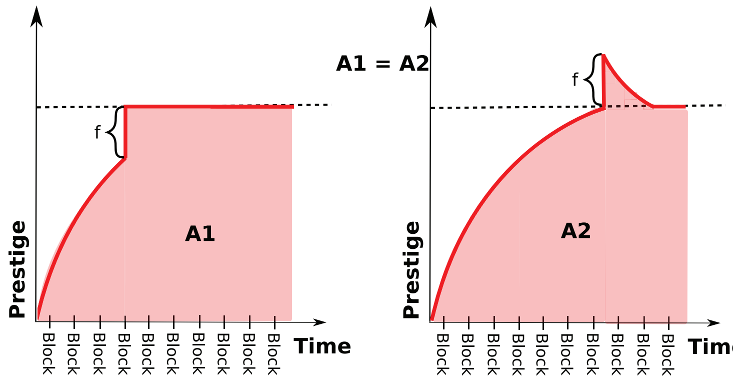

Fig. 3. Time of submitting a proof of useful work does not influence the probability of minting a new block within a specified time frame (area under the curve).

of minting the new block. In turn, minting a new block results in extra coins and thus in a higher *Threshold Value* (see User U_2 in Figure 2).

Note, that if the prestige coming from useful work is adequately smaller than the current, at time-slot t, prestige of user i, P_i^t, it has small difference if it is included immediately to the prestige of the user or at different time interval(s). In fact, given a time interval $[1, t_2]$ where a user fails to mint a coin, it can be shown, since the system converges to a Threshold Value, that the volume of prestige with and without including the prestige coming from useful work converges asymptotically as we increase t_2. This means that users can submit their proofs of useful work later without significantly lowering their probability of minting new coins (Figure 3). Such a solution allows us to adapt the solution for constraint devices (Section 9).

In PoP, users do not pay with coins when benefiting from/receiving a service but with prestige that, even if depleted, will be slowly replenished (as long as the user has some coins). However, when user prestige is higher than its threshold value, the decay exceeds prestige gained from coins and the user loses prestige until they reach their threshold value again. The reduction in prestige is another desired property, as it incentivizes users to keep on contributing to the system by performing useful work. The network acts thus as a closed-loop control system correcting users current prestige to their threshold values (Figure 2).

Our system needs to be secure and resistant against Sybil and collusion attacks (Section 2). When having a fixed amount of coins, we claim that users cannot gain more prestige by creating Sybil identities:

THEOREM 1. *Each user has no prestige gain incentives to divide their coins into multiple identities.*

We present all the proofs in Appendix A.

4.1 Mining Overview

Users can act both as contributors and as beneficiaries and exchange services for prestige. The total change of prestige for useful work δx_i^t of U_i at block t is thus given by prestige spent to benefit from services (being a beneficiary) and gained by performing useful work (being a contributor),

$$\delta x_i^t = x_{gained}^t - x_{spent}^t. \tag{4}$$

The total amount of prestige that a user is spending is the sum of prestige spent on each service that U_i benefited from $x_{spent}^i = \sum_{j=0}^m x_{ij}$, where the prestige fee transferred between each pair of nodes $x_{ij} = f$ can be predefined or negotiated between the beneficiary and the contributor.

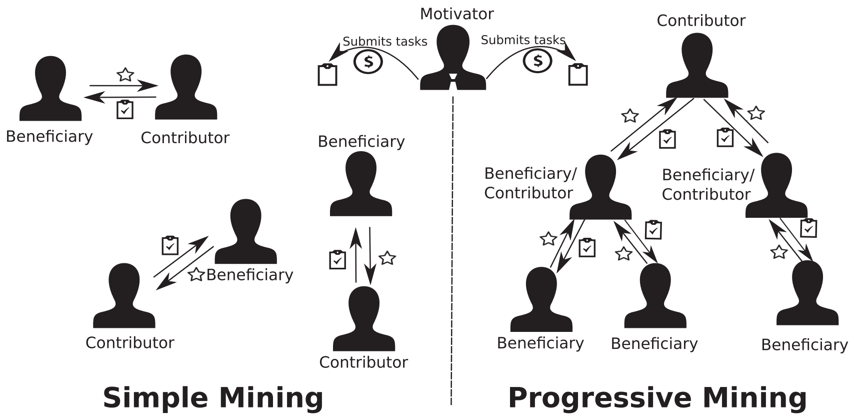

Fig. 4. Simple and Progressive Mining.

Analogically, the amount of prestige gained by a contributor is the sum of prestige gained on each service that U_i performed but is modified by a retain function $x^i_{gained} = \sum_{j=0}^{m} f_{retain}(x_{ji})$. The retain function defines what percentage of received prestige will be kept by a contributor. We define different patterns of useful work (represented by simple and progressive mining) with different retain functions explained in detail in the following sections. When a task is completed by a contributor, the beneficiary recognizes it by generating a signed *acknowledgment* and sends it to the contributor (Section 4.4). The contributor can then upload the acknowledgment to the blockchain to register the completed task and get the corresponding prestige.

When performing useful work during block k and as long as the amount of prestige transferred by the beneficiary U_i is greater than or equal to the prestige retained by the contributor U_j, so that $x^k_{ij} \geq f_{retain}(x_{ji})^k$, then the aggregate amount of prestige possessed by U_i *and* U_j does not increase. This means that *users cannot increase their minting power by cross-acknowledging their work and the system is resistant against both the Sybil and collusion attacks.*

Theorem 2. *There are no prestige gains produced by prestige transfers between users.*

The result of Theorem 2 can be easily extended for the general case of N users and all time-slots. We present all the proofs in Appendix A.

4.2 Simple Mining

We define *Simple Mining* to reward services performed uniquely between one contributor and one beneficiary (i.e., renting out contributor's CPU power to perform beneficiary's computations). In *Simple Mining*, when an acknowledgment of service performed by contributor U_i for beneficiary U_j is submitted to the blockchain, the contributor retains the whole amount of transferred prestige f so that $f = x^t_{ij} = -x^t_{ji}$ (Figure 4). Therefore, the simple mining retain function is defined as

$$f_{retain}(x_{ij}) = x_{ij}. \tag{5}$$

In *Simple Mining*, the retained value is equal to the transferred value. Therefore, Theorem 2 applies and we conclude that *Simple Mining* is resistant to Sybil and collusion attacks.

4.3 Progressive Mining

For cases where benefiting from a service allows the beneficiary to perform useful work for others (e.g., seeding in a content distribution system), we introduce the concept of *Progressive Mining*. In

Progressive Mining, contributors are rewarded by their own useful work but also for work performed by their beneficiaries. That is, if U_i performs a service for U_j, then U_i will receive some prestige for each service performed by U_j and other nodes that U_j provided the service to (Figure 4). The scheme can be seen as a Directed Acyclic Graph (DAG) with users as nodes and edges representing useful work performed for subsequent users. In this case, Prestige is flowing from the leaves toward the root.

This type of rewarding scheme is useful in scenarios such as file propagation, where receiving a file allows it to be distributed to other users. At the same time, we want to protect the system from distribution manipulations that would change the amount of prestige received by legitimate parties.

In particular, when U_j performs a task for U_i, the transferred prestige value x_{ji} is not directly added to U_j's account. Instead, U_j must share earned prestige with its DAG predecessors and can retain only a part of earned prestige:

$$f_{retain}(x_{ji}) = \frac{x_{ji} * P_i}{P_i + P_b^i}, \tag{6}$$

where P_b^i is branch power of U_i expressed as the sum of the prestige values of the predecessors of U_i multiplied by a branch power parameter b:

$$P_b^i = sum\_prestige(predecessors(U_i)) * b. \tag{7}$$

Such a branch power function incentivizes users to attach to shorter branches, automatically balancing the DAG. U_i will thus keep only a part of x_{ji}, while the remaining part will be transferred upstream toward the root. If $P_b^i = 0$, then no prestige is sent upstream (which is the case for the DAG root), and users with no base prestige cannot retain any prestige flowing upstream, which protects the scheme from DAG manipulations using Sybil identities. With increasing P_b^i more prestige is pulled upstream toward the root of the distribution tree.

The whole scheme is based on user's own prestige and the prestige sum of its predecessors. It is fully resistant to topology manipulation using Sybil identities that do not have any prestige. Users also cannot increase their prestige gain by spreading their coins (and thus gain prestige) over several artificial identities.

Theorem 3. *In Progressive Mining, users cannot retain more prestige by splitting their coins into multiple identities.*

We present all the proofs in Appendix A.

4.4 Acknowledgments

Nodes generate *Acknowledgments* when benefiting from useful work. Contributors earn prestige by submitting a received *Acknowledgment* to the blockchain, showing that they provided some service to other nodes.

For *Simple Mining* (Section 4.2), acknowledgments are standard digital signatures. The beneficiary generates a signature on an ID uniquely identifying the task, the contributor's public key, and the agreed amount of prestige to transfer. This is transferred to the contributor who uploads it to the blockchain to trigger the prestige transfer.

For *Progressive Mining* (Section 4.3), acknowledgments are composite signatures [70]. Each node in the DAG branch composes its signature with the initial signature generated by the DAG root, forming a composite signature that contains the signature of each node involved in the task (Figure 5).

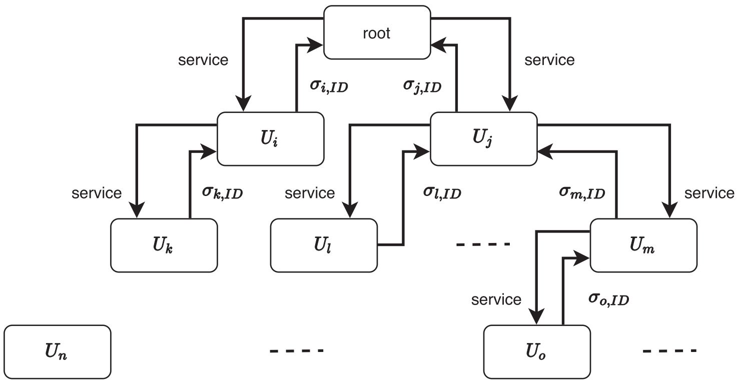

Fig. 5. Transfer of Proof of Prestige acknowledgments.

Composite signatures [70] are aggregate signatures satisfying the properties of (i) *incremental composition*, (ii) *one-way*, and (iii) *no-ordering*. *Incremental composition* means that a number of individual signatures can be combined into a single signature, and more signatures can be added to the composite at any time; *one-way* refers to the computational difficulty to obtain any sub-composite of signatures from only the composite signature; and *no-ordering* means that the composite signature does not maintain the order of the individual signatures composing it.

We adopt the definition of composite signature schemes given by Saxena et al. [70]. Let us consider a message-descriptor l made of pairs of (message,verification key): $l = \{(m_1, vk_1), \ldots, (m_i, vk_i)\}$. A composite signature scheme **SIG** is defined by the following set of algorithms:

- **SIG.Setup(1^λ) → ($params$)**: defines the system parameters $params$ with respect to the security parameter λ. These parameters are publicly available.
- **SIG.Gen($params$) → (sk, vk)**: generates their secret key sk and verification key vk from the public $params$.
- **SIG.Sign(sk, m) → (σ)**: outputs a single signature σ on the message m computed from the secret key sk.
- **SIG.Verify(l, σ) → (b)**: takes as input a message-descriptor l and a signature σ. Outputs a decision bit b about the validity of the signature if all the messages in l are unique; otherwise, it produces output: *invalid*.
- **SIG.Compose((l_1, σ_1), (l_2, σ_2)) → (σ)**: takes as input two pairs of (message-descriptors, signatures): (l_1, σ_1) and (l_2, σ_2). Outputs the composite signature σ on the message descriptors $l_1 \cap l_2$ if **Sig.Verify(l_1, σ_1)** $= valid$ and **Sig.Verify(l_2, σ_2)** $= valid$, and $l_1 \cap l_2 = \emptyset$; otherwise outputs $\perp$.

Saxena et al. [70] propose a provably secure composite signature scheme but whose signature size increases linearly with the number of compositions. They also posit that BGLS signatures [18] satisfy the definition of composite signatures under a weaker security assumption [70]. In this article, we use BGLS as composite signature scheme, as its signature size remains constant with the number of compositions.

When a new task is added, the DAG root node performs services to beneficiaries and collects their signatures $\sigma_{i,ID}$ over an ID uniquely identifying the task, the contributor's public key, and the agreed amount of prestige to transfer. The root then submits the signed message to the blockchain to receive prestige, while beneficiaries can turn into contributors and continue performing services for other users. These nodes start by sending $\sigma_{i,ID}$ to potential recipients. Each recipient checks the validity of the signatures and the task can be performed if the check passes. A beneficiary U_j generates his or her own signature and composes it with the previous signature $\sigma_{i,ID}$, obtaining a composite signature $\sigma_{j,ID}$. The beneficiary then sends the resulting composite signature to the contributor, who uploads it to the blockchain to update the prestige value of the previous contributors' and the root node. The above process continues as the DAG grows.

Before accepting a service using progressive mining, the beneficiary should verify that the contributor is already included in the DAG. To achieve this, the contributor transmits his or her own composite signature indicating the path from the DAG root to itself—this allows the beneficiary to register the transaction on the blockchain even if his or her predecessors do not do it. The properties of composite signatures prevent any single or subset of signature(s) from being removed from the composite [70]. Furthermore, the system will update the prestige of all nodes even if only the last sender in each branch uploads the composite signature to the blockchain. However, it is in the best interest of every contributor to upload the composite signature, in case no other subsequent node uploads theirs.

After benefiting from a service, the beneficiary might refuse to send back the acknowledgment or might attempt to generate an acknowledgment for another, colluding, node. While multiple solutions exist to ensure fair exchange between two mutually distrusting parties [9, 32, 66], a specific solution should be tailored to the nature of performed tasks. We design an offline-compatible solution allowing to atomically exchange files for acknowledgments in Section 7.

5 PRESTIGE ECONOMICS

In the previous sections, we explained the prestige flow between users. However, prestige is a volatile resource and does not represent real assets. High prestige value increases the chances of a node to be elected to submit a new block. That said, to incentivize users, it must be bound to a reward expressed in coins. In current blockchain payments systems, such as Bitcoin, the rewards come from (i) fees paid by users submitting transactions included in the block and (ii) new coins being discovered with each new block. The fees protect from Denial of Service attacks but also increase the exploitation cost of the system. However, new coins increase inflation and de facto reward the successful miner from money stored in the wallets of other users.

PoP is compatible with both reward methods described above, but, additionally, we introduce a third type of reward based on optional fees paid by motivators. Motivators directly benefit from increasing network size and can add coins as an additional reward for mining new blocks. The reward can be specified as an amount of coins distributed per block within limited duration (i.e., 1,000 blocks). Motivators can thus incentivize users to participate in a specific task when their services are needed the most.

This approach works if motivators want to increase rewards for the network as a whole. However, it is challenging to securely reward workers for a specific task. Rewarding all the participants of a task incentivizes users to join as many tasks as possible. For a temporary loss of prestige, they would acquire non-volatile coins. On the other side, making the reward dependent on the number of performed tasks or acquired prestige makes the system vulnerable to collusion attacks.

To overcome this problem, we introduce the concept of *"service domains"* (Figure 6). A domain $\mathcal{D}$ can be created by a motivator who wants to incentivize a subset of users. Tasks in a domain can be published only by its creator. Furthermore, to become workers or beneficiaries in the domain, users must be a part of it. To join a domain, a user must deposit some coins in the domain, which in turn, will generate local prestige. Each coin can be present in one domain only at any given time. Local prestige can be earned and spent only on local tasks and cannot leave the domain (local coins

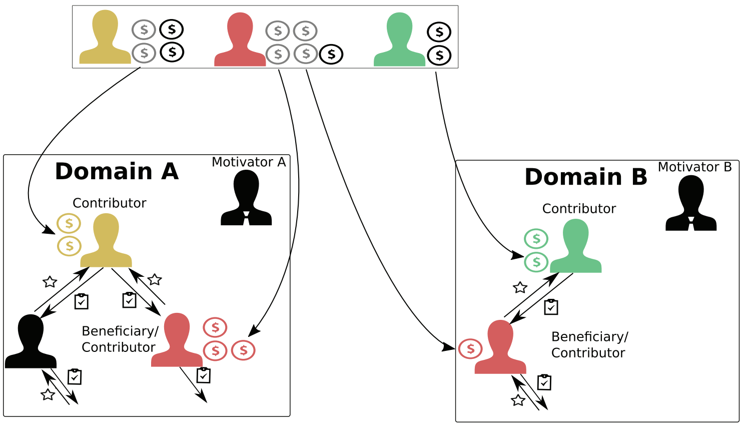

Fig. 6. Domain system. Users can put each coin into one domain only at any given time. The coins generate domain specific prestige that is taken into account when choosing the new miner.

can always be withdrawn). In this way, rewards from motivators can now be securely distributed proportionally to users' local prestige as Theorem 1 applies.

Each domain can be conceptually seen as a separate blockchain with its own users. The system can be implemented as a union of different chains connected by a protocol allowing migration of coins (see Section 9) or on top of a single ledger. There is no limit of deposit coins required to join a domain. The users are free to divide their coins to join as many domains as they desire and perform useful work. The total amount of local prestige is still taken into account for the consensus protocol. Considering the global system as one of the domains, the current prestige of each user can be calculated by:

$$P_i = \sum P_{ik}, \tag{8}$$

where P_{ik} is prestige of user U_i in domain $\mathcal{D}_k$. Because the coins can be assigned to one domain only at any given time, and the total amount of user prestige is a sum of their prestige in different domains, the process is similar to splitting coins into multiple Sybil identities and Theorem 1 applies again.

6 MOTIVATING USE CASES

We present two motivating examples of real-world applications. We provide a description of each platform, its current shortcomings and explain how PoP can help to solve them.

6.1 Content Publishing Platform—Simple Mining

Description: Many popular Content Publishing Platforms (CPPs) apply a hybrid publication model (e.g., Medium [36–38]). CPPs create only a small fraction of their content. The vast majority of available articles comes from independent creators (journalists and bloggers) who use CPPs as publishing platforms. While readers can access a few articles per month for free, most of the content is hidden behind a paywall. Subscription fees are the main source of revenue for the company. Independent creators can become a part of a partner program and earn money when subscribing users read and up-vote their work.

Shortcomings: Independent creators need to fully trust the publisher when receiving payment for their work. Nothing is preventing CPPs, who act as a single point of trust, from artificially understating the number of users who read the content and issuing lower rewards. Furthermore, users who paid a flat subscription fee can be used as click farms that up-vote all the articles of specified authors. Such strategy artificially inflates the rewards of those authors, leaving less money to be divided among honest creators.

Solution: Introducing the PoP Simple Mining scheme into CPPs could solve those problems. Users who pay subscription fees receive coins from CPPs and start generating prestige. Readers generate cryptographic acknowledgments (Section 4.4), inflating authors' prestige and allowing the authors to reliably verify the size of their audience. PoP automatically distributes rewards, preventing any forms of manipulating the payments. Prestige generated by users is limited in time. Cross-recommendations no longer make sense, as they do not increase the combined reward of the involved parties. Readers are thus incentivized to vote for only the best articles they find to receive high-quality content in the future. Finally, a blockchain-based system does not involve legal fees (i.e., signing a contract with an independent creator), lowering operating costs of the platform.

6.2 File Distribution—Progressive Mining

Description: In the traditional Web delivery model, content is pulled from the Content Delivery Network (CDN) server upon a user's request. Using CDNs, however, involves substantial fees and scales badly with an increasing number of users.

Recently, multiple decentralized file distribution platforms have been proposed. Systems like Filecoin/incentives for file propagation (IPFS) [24] or NOIA [5] do not build their own infrastructure but rather rely on a network of decentralized workers. Users themselves contribute to the distribution of the content directly between their devices without involving third-party CDNs. Content publishers submit their content for propagation together with a monetary fee. Decentralized file distribution platforms divide those fees among their workers based on the amount of performed work. Effectively, users participate in an *"incentivized content propagation network,"* where they get rewarded for contributing their resources to the network.

Shortcomings: It is difficult to reliably calculate the amount of content each worker has delivered. Existing systems either do not provide IPFS or are vulnerable to Sybil and collusion attacks (NOIA).

Solution: PoP progressive mining provides an efficient solution to those problems. In the beginning, a file is directly transferred from the creator to a few users, initiating a distribution DAG with the creator as its root (right part of Figure 4). The propagation then continues directly between user devices expanding the DAG. Each time users receive a file, they act as beneficiaries and must generate an acknowledgment for the contributor who sent the file. To reduce the risk of a malicious beneficiary walking away without generating an acknowledgment, each file is partitioned into small chunks. A new chunk is sent only if an acknowledgment for the previous one has been received.

PoP protects again colluding users exchanging content between them and incentivizes timely data propagation. A contributor transmits a file to a beneficiary and is rewarded not only for this one-hop transfer but for all the subsequent transfers in their subtree. At the same time, content creators/publishers can significantly reduce the cost of content distribution.

7 FAIR EXCHANGE

We extend the File Exchange use-case presented above. PoP guarantees that the file sender will be rewarded for its worked only if he or she receives an acknowledgment from the receiver. However, the receiver can decide to walk away just after file transmission. One solution would be to divide files into small chunks and send the next part only if acknowledgment for the previous one was received. Such a solution reduces the scale of the problem but does not eradicate it completely. To counter this issue, we adopt FairSwap [32] and modify it to our needs.

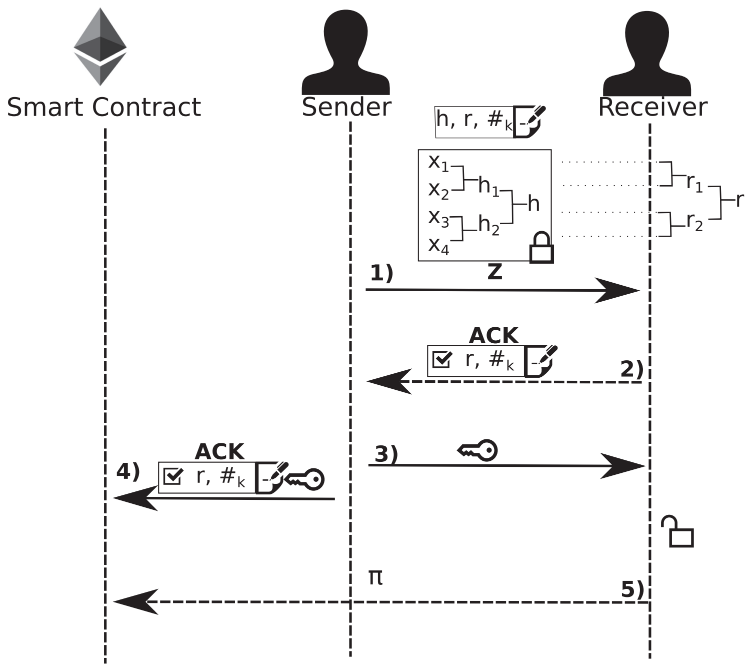

Fig. 7. Offline-compatible fair exchange scheme based on FairSwap.

FairSwap is a protocol allowing atomical exchange of digital goods (such as files) for a payment (expressed in cryptocurrencies). The seller starts by dividing a file to be sent into n chunks and creating a Merkle Tree from it so that $x = (x_1, x_2, \ldots, x_n)$, $h = root(MTree(x))$. The chunked file, together with all the intermediary Merkle nodes are then encrypted using key k creating encoding $z = Enc(k, (x, MTree(x)))$. Encoding z and h are then sent to the buyer. The buyer can now create a Merkle Tree from encoding z and submit it to the smart contract $r = root(Mtree(z))$ together with a conditional payment for the file. If r is correct, then the seller continues by submitting key k to the smart contract, making it automatically available for the buyer. At this point the buyer is able to decrypt z, access file x, and calculate its Merkle Tree root h'. If $h' = h$, then the file is correct, the payment will be eventually unlocked by the smart contract, and the seller receives the money.

However, if $h' \neq h$, then the buyer will construct a proof of misbehavior π pointing out parts of z containing files chunks that differ from the declared ones. The proof can be then submitted to the smart contract together with a Merkle proof attesting that invalid chunks were a part of encoding z. That the smart contract uses r to verify if the claim is correct and k to decrypt invalid parts of z. If the transmitted file was indeed corrupted, then the funds will be returned to the buyer.

FairSwap guarantees that the buyer receives the money if and only if the transmitted file was correct making it useful in PoP. However, the original protocol requires both parties to be online and communicate with the smart contract during multiple rounds. In PoP, we want to allow peers to exchange files using P2P technologies also when both parties are not connected to the Internet.

We thus modify the FairSwap protocol described above (Figure 7) to fit our needs. As in the original version, the file sender starts by computing z and sends it to the receiver. Additionally, the sender creates a message containing h, r, and a hash of the encryption key $\#_k$, signs it, and also transfers it to the receiver (step 1). The receiver validates correctness of r and responds with its own signed message (step 2) containing an acknowledgment (as descrribed in Section 4.4), r, and

$\#_k$. At this point, the sender has an acknowledgment and can respond with the decryption key k used to create z (step 3). However, a malicious sender could still refuse to send the key. To prevent such behavior, the smart contract require the whole message from step 2 to be submitted together with k (step 4). The submission is considered valid only if k hashes to $\#_k$. If $\#_k$ and k are not correct, then the receiver can submit a proof of misbehavior π (step 5) exactly like in the original version of the protocol [32].

The modified version enables to exchange files for PoP acknowledgments. Furthermore, the file transfer can happen without both parties being connected to the blockchain. If the sender behaves correctly, then the receiver can decrypt its file right away. However, even a malicious sender will be forced to reveal the decryption key to receive a reward. To avoid waiting for a long time, the acknowledgment can contain a validity timestamp[4] negotiated during first steps between sender and receiver.

8 EVALUATION

To evaluate the behavior of Proof-of-Prestige, we developed a discrete event Python3 simulator.[5] The simulator maintains a list of users and their prestige and coins and periodically creates new blocks, iterating through all the users in the system and updating their prestige values. Furthermore, the simulator can generate random iteraction graphs (where users perform services for each other) or read traces provided by the user.

In the following, we try to answer the following questions:

(1) How do system parameters (decay, gained prestige) influence prestige acquired by the users?
(2) How does the system behave when applied to real-world traces of human interaction?
(3) What are the rewards users can expect by participating in the system and how is this value determined by their wealth and the amount of work they perform?

8.1 Impact of Parameters

We start by investigating the behavior of the prestige correction function shown in Equation (1) and the impact of the decay d parameter and the amount prestige gained by users with each block.

Decay Parameter, d: Figure 8 shows the influence of the decay parameter d on prestige gain. We create four users with different amounts of coins and d values. All the users start with zero prestige $P^0 = 0$; we add prestige $\delta P_i = 200$ at block $t = 100$ to each user and remove prestige by $\delta P = -200$ at block $t = 150$. The number of coins determines the threshold value (Equation (3)), but the number of coins does not influence the time needed to converge back to the threshold value. However, increasing the decay parameter d lowers the threshold value and reduces the time required by each user to reach their threshold value from $P = 0$. The same applies when users receive ($t = 100$) or spend ($t = 150$) prestige. A higher decay parameter will make prestige go back to their threshold values faster. In contrast, reducing the decay parameter increases the value of prestige gained from useful work in comparison to prestige gained from coins.

Gained Prestige: Figure 9 introduces four users with the same amount of coins $C = 100$ and prestige set to the corresponding threshold value $P_i = T_i$. We then inject different amounts of prestige per block to each user, reflecting the case where each user has provided services of different value, and let the system run for 10,000 blocks. We measure the sum of gained prestige for this period for different values of the decay parameter $0 < d < 1$ (x-axis). The impact of gained prestige

[4]expressed in blocks.
[5]https://gitlab.com/mharnen/pop.

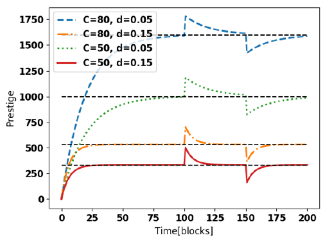

Fig. 8. Prestige over time.

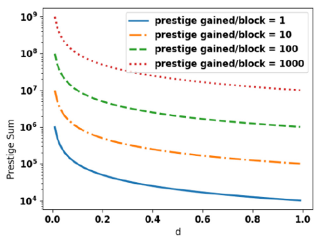

Fig. 9. Prestige sum above threshold value.

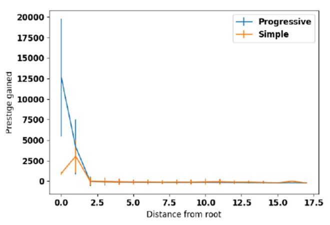

Fig. 10. Distance from root.

(and thus of useful work) increases exponentially for small values of d, while it decreases for higher values of d (notably for $d \approx 1$, where all the gained prestige is removed in the next block).

8.2 System Behavior with Real-world Traces

We perform tests using traces from content distribution simulations performed in the city of Helsinki [68]. The traces contains 118 DAGs with user polls ranging from 100 to 811 participants. In those scenarios, content is transmitted from publishers to buses running across the city and spreading the content to passengers and nearby pedestrians. Walking users can also transmit the content between each other. We measure amounts of prestige acquired by participants using both simple and progressive mining. The graphs present average values from all the traces. Initial prestige values follow a uniform distribution from 0 to 100, while the branch power parameter is set to $b = 0.5$.

Distance from the DAG root: Figure 10 presents the average prestige gained by each user as a function of distance from the DAG root (the initial content distributor). In simple mining, the root gains significantly more prestige than other users as it acts only as a contributor and does not benefit from (and thus pay for) services. For the other nodes, the distance does not influence the prestige gain and the standard error is low.

In contrast, in progressive mining, users located close to the root have the highest average prestige gain, while with increasing distance, the rewards decrease. This is the desired behavior, as users close to the root have larger subtrees and collect fees from useful work of their successors. Progressive mining takes into account multiple factors when calculating the reward (e.g., base prestige and distance from the root), which results in increased standard error.

Base Prestige: We investigate the prestige gain as a function of base prestige (Figure 11). Simple mining is not influenced by base prestige, while in progressive mining, with increasing base prestige, a user can collect higher rewards. This mechanism prevents Sybil attacks and rewards users who invested more in the system. However, a small fraction of high base prestige users experiences prestige loss. Those are users that in spite of having high prestige, benefit from services, and do not perform any useful work, which is another desired behavior.

Number of Completed Tasks: Next, we investigate the effect of useful work on user prestige gain (Figure 12). For both progressive and simple mining, the average gain increases linearly with the number of services performed for other users. This experiment proves that for both mining modes, users with low base prestige and located further from the root in the distribution tree, can gain significant amounts of prestige by being useful for the network.

8.3 Reward for Users

Contributor Involvement Probability: We investigate prestige dynamics over time with both simple (Figure 13) and progressive (Figure 14) mining to provide a global view of the system. We

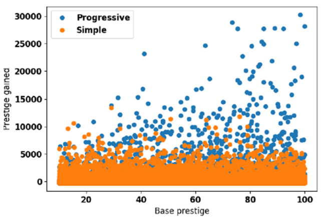

Fig. 11. Base Prestige.

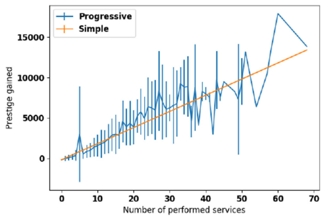

Fig. 12. Useful work.

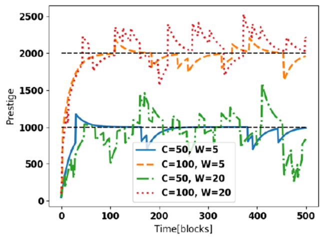

Fig. 13. Prestige evolution for work Simple Mining.

introduce different poor ($C = 50$) and rich ($C = 100$) users having $W = 5\%$ or $W = 20\%$ probability of performing useful work with each new block in a random DAG containing 100 nodes, service fee $f = 200$, and decay parameter $d = 0.05$. At the beginning of the simulation, the prestige values of each user go from 0 toward their corresponding threshold value. In simple mining, users gain steady and moderate amounts of prestige, and we do not observe differences between poor and rich users. In comparison, users performing progressive mining can reach much higher prestige gains, and an increased amount of coins (and thus base prestige) allows richer users to maximize their reward. The amount of performed useful work W is important in both schemes, but its impact is higher in progressive mining, where poor, but active, users can reach prestige values similar to much richer, but less active, nodes.

Contribution vs. Coins Tradeoff: We conclude by investigating the total value of acquired prestige over 1,000 blocks by rich ($C = 50$) and poor ($C = 10$) users having different probabilities of performing useful work ($W = 5\%$ and $W = 25\%$) for different values of the parameter d (Figure 15). For small values of d, useful work is more important than money. Poorer, but more active, users can thus acquire a substantial amount of prestige, eventually surpassing rich users. This effect is decreased with increased values of d. On the other end of the spectrum, for high values ($d > 20$) the sum of acquired prestige depends mostly on user money and is independent of the amount of performed useful work.

Rewards: To investigate potential rewards for users using Proof-of-Prestige, we focus on the File Distribution use case presented in Section 6.2 and apply it to popular a BBC Series: *Bodyguard*. BBC does not include advertisements in their content—thus, each consecutive download increases the broadcaster's cost. With the number of viewers ranges from 14M to 17M,[6] and the file size of approximately 250 MB, the cost of delivering one series season using a CDN equals 4.7M$.[7] For each episode of the series, we create a DAG with the corresponding size of users with a random amount of base prestige ranging from 1 to 10,000 and distribute the money spent on CDN to users proportionally to their prestige after performing useful work. In this scenario, users transfer files to a maximum of eight users. Figure 16 shows the reward distribution for different values of f and b. Surprisingly, changing those parameters has a negligible effect on the majority of the nodes. User receive but also transfer upstream different amounts of prestige. Such behavior influences mainly the most active nodes located close to the DAG root. With high f and b, those nodes can acquire significantly higher amounts of prestige and thus collect higher rewards. The most active users collect up to 30$ reward, while the distribution remains free for all the users.

[6]https://www.barb.co.uk/viewing-data/four-screen-dashboard/.
[7]http://cdncomparison.com/.

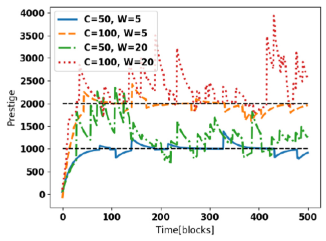

Fig. 14. Prestige evolution for work in Progressive Mining.

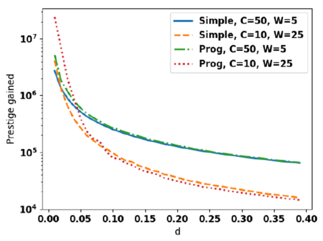

Fig. 15. Prestige sum acquired by different users.

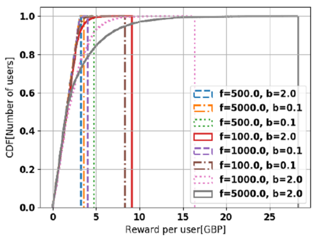

Fig. 16. Reward Distribution.

9 PROTOTYPE

To evaluate the practicality of our solution, we deploy a full content-distribution platform based on the use-case presented in Section 6. The prototype contains a mobile application allowing users to exchange content and a blockchain component calculating and distributing rewards. We deploy two PoP implementations: (i) as an Ethereum [76] smart contract (PoW) and (ii) a native implementation within Tendermint [54]—a consensus protocol used by Cosmos SDK [26] (PoS).

We describe each component in details and aim to answer the following questions:

(1) How difficult is it to implement and deploy PoP?
(2) Can PoP be used by mobile, power-constrained devices?
(3) What is the monetary cost of using PoP on top of a PoW blockchain?
(4) What is the computational overhead for miners to support PoP when natively included in the PoS implementation?

9.1 Mobile Application

We develop a device-to-device file sharing application [55] for Android phones and connect it with PoP. We implement an epidemic routing protocol to automatically discover and transfer content between users.

Contributor devices advertise their content using a Bloom filter encoded into Bluetooth Low Energy beacons. When a beneficiary user discovers a file they are interested in, they send a request, and the file is transferred between devices using WiFi direct [25]. Once the beneficiary finishes downloading data, he or she verifies its integrity by checking its digital signature. If the signature is correct, then the beneficiary transmits a signed acknowledgment to the contributor. The contributor device collects acknowledgments and can submit them later to a full node.

Due to a limited amount of memory, mobile devices are unable to keep blockchain data in their storage. We thus adapt our scheme so that Mobile devices require to store only the account credentials (public and private key). For integration with Ethereum, the application requires a contract interface in the form of a lightweight Application Binary Interface (ABI) file and uses web3j [57] library for interacting with the contract, credentials generation, and signing acknowledgments with the private key. When interacting with Cosmos, the mobile application sends regular transactions encoded in JSON.

Simple Mining requires a separate acknowledgment for each interaction—the number of submitted ACKs equals the number of performed tasks. The size of an ACK for simple mining is about 98 bytes; it is computed as the sum of the size of the composite signature[8] (33 bytes), the task ID (32 bytes), the contributor public key (33 bytes), and the amount of prestige transferred (set to

[8]We implement composite signatures with BGLS signatures [18].

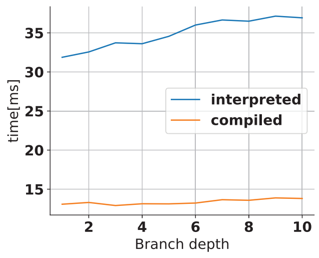

Fig. 17. Message signing time on mobile devices.

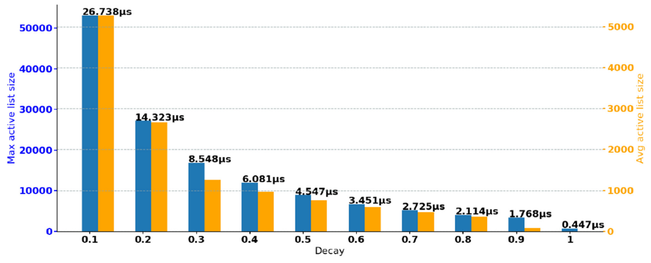

Fig. 18. Average and maximum active list size for different values od the decay. We also present list processing time for the maximum values.

4 bytes). Progressive mining requires an acknowledgment for each leaf in the DAG, since composite signatures contain information about all the nodes between the root and the leaf; and the size of the ACK increases linearly with the depth of the DAG. The ACK size is, therefore, $(32 + 66 \times n_i)$ bytes, where n_i is the depth of the DAG following path i.

Figure 17 presents the time required to sign acknowledgments using BGLS [18] on a Samsung Galaxy A50 with 4x1.7-GHz Cortex-A53 CPU, 4 GB of RAM and using Android v10. The results are shown for different DAG depths. Signing simple mining acknowledgments is equivalent to progressive mining acknowledgments with the DAG depth equal to 1. Android uses just-in-time compilation, continuously analyses the code being executed, and identifies parts of the code where the speedup gained from compilation would outweigh the overhead of compiling that code. We present results for both the interpreted and the compiled version. Android automatically optimizes the signing function when it is invoked multiple times (in our experiments 3 times). The signing time increases with message size but remains below 40 ms even for the largest messages and unoptimized code.

9.2 Ethereum: PoW

We deploy PoP as a smart contract on the Ethereum Ropsten testnet [34]. We are thus unable to collect transaction fees and newly minted coins, but our scheme can distribute rewards submitted by motivators.

Smart Contract: The smart contract acts as an intermediary between users and the PoP server (described below). It enables users to deploy ether into PoP. Each 1000wei translates into 1 PoP coin. The smart contract generates events when a user deploys or withdraws funds or registers a transfer between devices. Furthermore, beneficiaries can use contract methods to register their file transfers by sending signed acknowledgments. If the signature is correct, then the contract also generates a corresponding event. Finally, the contract allows motivators to register their rewards for specific file distributions. However, how the rewards are distributed is decided by the contract owner (in our case the PoP server). In such a setup fair reward distribution is not guaranteed by the blockchain; however, users cannot lose any coins and can withdraw their funds at any point.

Gas Cost: Table 2 presents methods implemented by the contract as well as the cost of their invocation. The cost is divided into Transaction and Execution cost.[9] The former depends on the amount of data transferred to the blockchain (with base a cost of 20,000 gas), while the latter depends on the computational operations that are executed in the virtual machine. The cost of all the methods (including the contract deployment) is lower than 1USD. The most important

[9]Based on https://ethgasstation.info/. PoP does not rely on timely execution; therefore, we use EthereumSlow confirmation time.

Table 2. Smart Contract Methods and the Invocation Cost

Function	Transaction Cost	Execution Cost
deployContract	655,980gas ($0.607)	460,532gas ($0.426)
deployMoney()	28,408gas ($0.026)	7,136gas ($0.007)
withdrawMoney()	20,708gas ($0.019)	14,436gas ($0.013)
putReward()	26,607gas ($0.025)	5,335gas ($0.005)
registerFile()	23,127gas ($0.021)	831gas ($0.001)
registerTransfer()	26,537gas ($0.025)	1,297gas ($0.001)

functions are *registerFile()* (executed once per file submitted for propagation) and *registerTransfer()* (executed once per file transfer between two peers). The rest of the methods must be executed only once and thus their cost is negligible. Both frequently executed functions can be executed 1,000 times for less than 1USD. This cost is 4 orders of magnitude lower than potential rewards from content providers (Section 8).

PoP Server: By default, Ethereum does not allow to automatically invoke functions with new blocks.[10] To reduce the cost of calculating users prestige on-chain, we outsource this operation to an external server. The servers are connected to a full Ethereum node using web3py [35] and listens for events generated by the blockchain such as a new block, new file transfer or money deployed into the smart contract. The server, as the contract owner, performs cyclic (every 1,000 blocks) payments proportional to prestige of each user.

9.3 Cosmos: PoS

We implement PoP natively on top of Cosmos blockchain [26] and deploy it on a Dell Latitude 5590 laptop with an Intel Core i7-8650U CPU and 16 GB of RAM. Cosmos is a decentralized network of independent parallel blockchains. The platform has several properties making it suitable for PoP deployment:

- uses Tendermint [54] Byzantine Fault Tolerant (BFT), Proof-of-Stake consensus protocol that is fully compatible with PoP.
- the Tendermint BFT engine is connected to the application by a socket protocol called the ABI. This protocol can be wrapped in any programming language, making it possible for developers to choose a language that fits their needs.
- enables connection blockchains through a protocol called Inter-Blockchain Communication protocol (IBC). IBC leverages the instant finality property of Tendermint consensus to allow heterogeneous chains to transfer value (i.e. tokens) or data to each other. This allows us to implement domains (Section 5) and enables easy user migrations.

We adapt the file-sharing mobile application developed for Ethereum, deploy Cosmos account keys and use REST/JSON to interact with the blockchain. We develop a GO application running on top of the consensus protocol that exposes a REST interface to the mobile client and submits data to the blockchain using ABI.

The main overhead results from updating the prestige values of all the users at the beginning of each block and increases linearly with the number of users. However, prestige does not increase nor decrease when users are at their threshold value (Equation (1)). Thus, our module maintains a list of users currently not at their threshold values. We refer to it as an *active user list*. Users are added to

[10]Such functionality is offered by several external services (i.e., https://github.com/ethereum-alarm-clock) but requires paying a fee with every new block.

the active user list when they receive/send a money transfer or become contributors/beneficiaries. In turn, users are removed from the active user list, once they reach their threshold values. Such an approach allows us to significantly reduce the overhead of updating prestige values and ignore the inactive users. Modifying accounts can be performed in parallel, we thus implement the function as a set of GO coroutines enabling usage of multiple cores.

To evaluate the potential overhead on each miner, we extract the history of the Ethereum blockchain[11] and replay all the transactions on the Cosmos prototype. We analyze first 8,163,516 blocks translating into more than 3 years of transaction history.[12] We then observe the number of users in the active user list of each block and measure the time required to update prestige values.

Figure 18 presents the maximum/average observed length of the active list and the processing time for different decay values. With increasing decay, prestige values reach their threshold values faster and are thus removed earlier from the active list reducing its size. For the lowest decay value (0.1), the maximum observed list size equals to 53,124 users. This value lowers with the increasing decay to reach 760 users for decay equals to 1. Updating the prestige of a user according to Equation (1) requires only simple arithmetic operations (an addition, a multiplication, and a subtraction). Even for the largest observed active list, the computational overhead is lower than 30 μs on a regular laptop.

10 RELATED WORK

There are currently multiple systems focusing on rewarding miners for useful work. The largest group focuses on file storage where a prover can create proof that a file was correctly written [31] and stored [13, 19, 30, 39, 41, 50, 56, 61, 63, 64, 73, 78] and can be retrieved [58, 79].

Alternatively, several platforms allow the prover to convince that the prover has access to some storage space [3, 12, 22, 33, 39, 45, 51, 64, 67]. Additionally, one can require that the proof implies that the space also was erased [51, 67] or some function can only be computed in the forward direction [33].

Some solutions replace traditional Proof-of-Work with useful mathematical tasks that are easy to verify, such as polynomials evaluation [15, 46, 77], solving linear equations [62], or handling costly encryption [80]. All of those systems support only specific tasks and cannot be adapted to more general cases. PoP presents an alternative to those solutions, as it is able to support any type of task.

Another family supports broader range of tasks (such as custom computation tasks [4, 9, 10, 23, 28, 53, 60] but relies on Trusted Execution Environments (TEEs) and Remote Attestation Protocols [27, 75] to verify that the computations are being run on a genuine platform and that the results are correct. However, TEEs are not available on every platform, requiring users to trust hardware vendors, and are susceptible to side channel attacks [42, 74]. In contrast, PoP does not make any assumptions on users hardware.

Some open platforms rely on centralized third parties acting as consents to validate tasks or perform conflict resolution [2, 7, 11]. However, for those system to work correctly, the third parties must be trusted by all network participants. Such an approach contrasts with the idea of an open and trustless system lying behind the blockchain and exposes the network to multiple collusion attacks.

As a substitute for rewards, multiple works implement blockchain-based reputation systems [21, 44, 49, 59, 72]. While indicating trustworthy and malicious nodes, reputation systems do not directly incentivize correct useful work and may suffer from large-scale collusion attacks.

[11]The Cosmos blockchain is still in its infancy and does not have a significant user base.
[12]Average block interval in Ethereum equals 12 s.

Finally, there exist multiple industrial projects in which the network operator (motivator) wants to reward users for performing useful work that is difficult or impossible to prove to a third party, making those system susceptible to Sybil attacks. Such tasks include content creation [8], content distribution [5], or providing hardware for game players [6]. The need for a blockchain-based reputation system/reward scheme has also been expressed for multiple systems ranging from the cloud [43, 48, 69] to IoT devices [47, 71]. PoP can be a valuable addition to those system, allowing reliable rewards to users for truly performed tasks.

11 CONCLUSION

We presented PoP—a reward system that can run natively on top of Proof-of-Stake or as a smart contract on top of Proof-of-Work blockchains. We introduce the notion of *Prestige* as a volatile and renewable resource that is generated from coins and useful work and can be spent to benefit from services. In PoP, each user's probability of minting a new block is directly determined by his or her prestige.

In contrast to PoS, where the amount of coins (stake) a user has is the only resource that determines who mines the new block, PoP allows the network to reward contributors for their useful work acknowledged by beneficiaries. Our scheme is resistant to Sybil and collusion attacks and can be used in multiple scenarios without requiring to prove task completion to the network. Rather, a task is considered to be completed once confirmed by its beneficiary. Simple and progressive mining, the two variants of PoP, allow support of a vast range of tasks, such as computation outsourcing, P2P file exchange, bandwidth/retrieval markets, incentivized content creation, and more.

We prove the mathematical properties of our scheme using a discrete event simulator. Our evaluation confirmed that within both schemes users with low amounts of coins who contribute to the network can acquire significant amounts of prestige, similar to rich and "lazy" users. PoP reduces inequalities present in PoS and incentivizes users to perform useful work. Finally, we implement and deploy a content distribution platform including mobile devices and blockchain modules implemented on top of Ethereum (PoW) and Cosmos (PoS). PoP introduces a few-millisecond transaction signing time on mobile devices, sub-millisecond computational overhead for miners in Cosmos, and less than 0.013$ smart contract invocation cost for users in Ethereum.

APPENDIX

A PROOFS

THEOREM 1. *Each user has no prestige gain incentives to divide their coins into multiple identities.*

PROOF. Let user i divide their coins into $1, 2, \ldots, k$ identities according to $C_{i,1}, C_{i,2}, \ldots, C_{i,k}$ respectively, such that $\sum_{q=1}^{k} C_{i,q} = C_i$. Then the aggregated prestige of multiple identities at their limit will be

$$T_{i,1} + T_{i,2} + \cdots + T_{i,k} = \sum_{q=1}^{k} C_{i,q}/d = C_i/d = T_i.$$

That is, the total prestige generated in the long run by multiple identities equals the prestige achieved by a single identity. □

THEOREM 2. *There are no prestige gains produced by prestige transfers between users.*

PROOF. We denote by P_i^t, P_j^t the prestige of contributor i and beneficiary j when there is no prestige transfer from i to j and by $\bar{P}_i^t$, $\bar{P}_j^t$ the prestige of contributor i and beneficiary j when the transfer x_{ij}^k is taking place upon time-slot $t > k$. Then:

$$\begin{aligned}
\bar{P}_i^t + \bar{P}_j^t &= P_i^{k-1} - x_{ij}^k + \sum_{q=k}^{t} \delta\bar{P}_i^q + P_j^{k-1} + x_{ij}^k + \sum_{q=k}^{t} \delta\bar{P}_j^q, \\
&= P_i^{k-1} + \sum_{q=k}^{t} \delta\bar{P}_i^q + P_j^{k-1} + \sum_{q=k}^{t} \delta\bar{P}_j^q, \\
&= P_i^{k-1} + C_i - d(P_i^k - x_{ij}^k) + \sum_{q=k+1}^{t} \delta\bar{P}_i^q \\
&+ P_j^{k-1} + C_j - d(P_j^k + x_{ij}^k) + \sum_{q=k+1}^{t} \delta\bar{P}_j^q, \\
&= P_i^k + \sum_{q=k+1}^{t} \delta\bar{P}_i^q + P_j^k + \sum_{q=k+1}^{t} \delta\bar{P}_j^q, \\
&\vdots \\
&= P_i^t + P_j^t.
\end{aligned}$$

That is, prestige transfers do not affect the total prestige that exists in the system. □

The result of Theorem 2 can be easily extended for the general case of N users and all time-slots.

THEOREM 3. *In Progressive Mining, users cannot retain more prestige by splitting their coins into multiple identities.*

PROOF. Without loss of generality, assume that user i divides his or her coins into two identities according to $C_{i,1}$ and $C_{i,2}$, such that $C_{i,1} + C_{i,2} = C_i$. Hence, the prestige retained by the multiple identities of user i out of the prestige transferred from user j, x, will be

$$\begin{aligned}
f_{retain,1}(x) - x + f_{retain,2}(2x - f_{retain,1}(x)) \\
= x\frac{P_{i,1}}{P_{i,1} + bP_{i,2} + P_b^i} - x \\
+ x\left(2 - \frac{P_{i,1}}{P_{i,1} + bP_{i,2} + P_b^i}\right)\frac{P_{i,2}}{P_{i,2} + P_b^i}, \\
= x\frac{P_{i,1}P_{i,2} + P_{i,2}P_b^i + bP_{i,2}^2 - bP_{i,2}P_b^i - (P_b^i)^2}{(P_{i,1} + bP_{i,2} + P_b^i)(P_{i,2} + P_b^i)}, \\
\leq x\frac{P_{i,1} + P_{i,2}}{P_{i,1} + P_{i,2} + P_b^i}, \\
= x\frac{P_i}{P_i + P_b^i}, \\
= f_{retain(x)}.
\end{aligned}$$

where the inequality applies, since

$$\begin{aligned}\frac{P_{i,1}P_{i,2} + P_{i,2}P_b^i + bP_{i,2}^2 - bP_{i,2}P_b^i - (P_b^i)^2}{(P_{i,1} + bP_{i,2} + P_b^i)(P_{i,2} + P_b^i)} \\ \leq \frac{P_{i,1} + P_{i,2}}{P_{i,1} + P_{i,2} + P_b^i} \iff \\ -P_b^i \left(bP_{i,1}P_{i,2} + P_{i,1}P_b^i + P_{i,2}(bP_{i,2} + P_b^i + bP_b^i) + (P_b^i)^2\right) \\ \leq P_{i,1}(P_{i,1}P_b^i + bP_{i,2}P_b^i + (P_b^i)^2).\end{aligned}$$

However, from Theorem 1 we know that multiple identities have no prestige gains, i.e., $P_{i,1} + P_{i,2} = P_i$, and therefore the third equality is valid. Users cannot increase their prestige by creating multiple identities in Progressive Mining. The prestige payments that a user is forced to submit to its fake identities retain only a portion of the prestige. □

REFERENCES

[1] 2015. ERC-20 token standard. Retrieved from https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.

[2] 2016. Golem whitepaper. Retrieved from https://golem.network/doc/Golemwhitepaper.pdf.

[3] 2018. Chia network. Retrieved from https://chia.network.

[4] 2018. iExec whitepaper. Retrieved from https://iex.ec/whitepaper/iExec-WPv3.0-English.pdf.

[5] 2018. Noia whitepaper. Retrieved from https://drive.google.com/file/d/1IfdKbai7hkScw_Zj6-kbZPoxNCmQzKaR/view.

[6] 2018. PlayKey whiteaper. Retrieved from https://cdn.playkey.net/img/playkeynet/ico/Whitepaper_1_31_En.pdf.

[7] 2018. SONM documentation. Retrieved from https://docs.sonm.com/.

[8] 2018. Steem whitepaper. Retrieved from https://steem.io/.

[9] Mustafa Al-Bassam, Alberto Sonnino, Michał Król, and Ioannis Psaras. 2018. Airtnt: Fair exchange payment for outsourced secure enclave computations. *arXiv:1805.06411*. Retrieved from https://arxiv.org/abs/1805.06411.

[10] Fritz Alder, N. Asokan, Arseny Kurnikov, Andrew Paverd, and Michael Steiner. 2019. S-faas: Trustworthy and accountable function-as-a-service using intel SGX. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*. 185–199.

[11] Adam Angelo, Pascal Thellmann, and Deniz Dalkilic. 2018. Rewarding the Token Economy. Technical Report.

[12] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. 2014. Proofs of space: When space is of the essence. In *Proceedings of the International Conference on Security and Cryptography for Networks*. Springer, 538–557.

[13] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. 2007. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, 598–609.

[14] Leemon Baird. 2016. The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance. Technical Report.

[15] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. 2017. Proofs of useful work. *IACR Cryptol. ePrint Arch.* 2017 (2017), 203.

[16] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. 2017. Consensus in the age of blockchains. *arXiv:1711.03936*. Retrieved from https://arxiv.org/abs/1711.03936.

[17] Iddo Bentov, Rafael Pass, and Elaine Shi. 2016. Snow white: Provably secure proofs of stake. *IACR Cryptol. ePrint Arch.* 2016 (2016), 919.

[18] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'03)*, Vol. 2656. Springer, 416–432.

[19] Kevin D. Bowers, Ari Juels, and Alina Oprea. 2009. Proofs of retrievability: Theory and implementation. In *Proceedings of the ACM Workshop on Cloud Computing Security*. ACM, 43–54.

[20] Vitalik Buterin and Virgil Griffith. 2017. Casper the friendly finality gadget. *CoRR* abs/1710.09437 (2017).

[21] Maqsood Ahamed Abdul Careem and Aveek Dutta. 2019. SenseChain: Blockchain based reputation system for distributed spectrum enforcement. In *Proceedings of the 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN'19)*. IEEE, 1–10.

[22] Binyi Chen, Yilei Chen, Kristina Hostáková, and Pratyay Mukherjee. 2019. Continuous space-bounded non-malleable codes from stronger proofs-of-space. In *Proceedings of the Annual International Cryptology Conference*. Springer, 467–495.

[23] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. 2017. On security analysis of proof-of-elapsed-time (poet). In *Proceedings of the International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 282–297.

[24] Y. Combinator. [n.d.]. IPFS, CoinList, and the Filecoin ICO with Juan Benet. June 30th 2017.

[25] Marco Conti, Franca Delmastro, Giovanni Minutiello, and Roberta Paris. 2013. Experimenting opportunistic networks with WiFi direct. In *Proceedings of the 2013 IFIP Wireless Days Conference (WD'13)*. IEEE, 1–6.

[26] Cosmos. [n.d.]. Cosmos SDK. Retrieved from https://cosmos.network/.

[27] Victor Costan and Srinivas Devadas. 2016. Intel SGX explained. *IACR Cryptol. ePrint Arch.* 2016, 086 (2016), 1–118.

[28] Hung Dang, Dat Le Tien, and Ee-Chien Chang. 2019. Towards a marketplace for secure outsourced computations. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, 790–808.

[29] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2017. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. *IOHK Paper* (2017).

[30] Roberto Di Pietro, Luigi V. Mancini, Yee Wei Law, Sandro Etalle, and Paul Havinga. 2003. LKHW: A directed diffusion-based secure multicast scheme for wireless sensor networks. In *Proceedings of the International Conference on Parallel Processing Workshops*. IEEE, 397–406.

[31] Dan Dobre, Ghassan O. Karame, Wenting Li, Matthias Majuntke, Neeraj Suri, and Marko Vukolić. 2019. Proofs of writing for robust storage. *IEEE Trans. Parallel Distrib. Syst.* 30, 11 (2019), 2547–2566.

[32] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. 2018. FairSwap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 967–984.

[33] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. 2011. One-time computable self-erasing functions. In *Proceedings of the Theory of Cryptography Conference*. Springer, 125–143.

[34] Ethereum. [n.d.]. Ropsten network. Retrieved from https://github.com/ethereum/ropsten.

[35] Ethereum. [n.d.]. Web3Py. Retrieved from https://github.com/ethereum/web3.py.

[36] Williams Ev. [n.d.]. The Medium Model. Retrieved from https://blog.medium.com/the-medium-model-3ec28c6f603a.

[37] Williams Ev. [n.d.]. Medium Partner Program. Retrieved from https://medium.com/creators.

[38] Williams Ev. [n.d.]. The rationalization of publishing. Retrieved from https://medium.com/@ev/the-rationalization-of-publishing-dc001d509de8.

[39] Ben Fisch. 2019. Tight proofs of space and replication. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 324–348.

[40] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 51–68.

[41] Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. 2002. Cryptographic primitives enforcing communication and storage complexity. In *Proceedings of the International Conference on Financial Cryptography*. Springer, 120–135.

[42] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. 2017. Cache attacks on intel SGX. In *Proceedings of the 10th European Workshop on Systems Security*. ACM, 2.

[43] Sheikh Mahbub Habib, Sebastian Ries, and Max Muhlhauser. 2010. Cloud computing landscape and research challenges regarding trust and reputation. In *Proceedings of the 2010 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing*. IEEE, 410–415.

[44] Liviu-Adrian Hîrţan, Ciprian Dobre, and Horacio González-Vélez. 2020. Blockchain-based reputation for intelligent transportation systems. *Sensors* 20, 3 (2020), 791.

[45] Trond Hønsi. 2017. *SpaceMint-A Cryptocurrency Based on Proofs of Space*. Master's thesis. NTNU.

[46] Xing Hu and Chunming Tang. 2015. Secure outsourced computation of the characteristic polynomial and eigenvalues of matrix. *J. Cloud Comput.* 4, 1 (2015), 7.

[47] Steve Huckle, Rituparna Bhattacharya, Martin White, and Natalia Beloff. 2016. Internet of things, blockchain and shared economy applications. *Proc. Comput. Sci.* 98 (2016), 461–466.

[48] Kai Hwang and Deyi Li. 2010. Trusted cloud computing with secure resources and data coloring. *IEEE Internet Comput.* 14, 5 (2010), 14–22.

[49] Atia Javaid, Maheen Zahid, Ishtiaq Ali, Raja Jalees Ul Hussen Khan, Zainib Noshad, and Nadeem Javaid. 2019. Reputation system for iot data monetization using blockchain. In *Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, 173–184.

[50] Ari Juels and Burton S. Kaliski Jr. 2007. PORs: Proofs of retrievability for large files. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. Acm, 584–597.

[51] Nikolaos P. Karvelas and Aggelos Kiayias. 2014. Efficient proofs of secure erasure. In *Proceedings of the International Conference on Security and Cryptography for Networks*. Springer, 520–537.

[52] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Proceedings of the Annual International Cryptology Conference*. Springer, 357–388.

[53] Michał Król and Ioannis Psaras. 2018. Spoc: Secure payments for outsourced computations. In *Proceedings of the NDSS Workshop on Decentralized IoT Security and Standards*.

[54] Jae Kwon. 2014. Tendermint: Consensus without mining. *Draft v. 0.6, Fall* 1, 11 (2014).

[55] DataHop Labs. [n.d.]. Mobile application. Retrieved from https://play.google.com/store/apps/details?id=network.datahop.localsharing.

[56] Protocol Labs. 2017. *Proof of Replication Technical Report*. Technical Report.

[57] Web3 Labs. [n.d.]. Web3J. Retrieved from https://github.com/web3j/.

[58] Julien Lavauzelle and Françoise Levy-dit Vehel. 2019. Generic constructions of PoRs from codes and instantiations. *J. Math. Cryptol.* 13, 2 (2019), 81–106.

[59] Dongxiao Liu, Amal Alahmadi, Jianbing Ni, Xiaodong Lin, and Xuemin Shen. 2019. Anonymous reputation system for iiot-enabled retail marketing atop pos blockchain. *IEEE Trans. Industr. Inf.* 15, 6 (2019), 3527–3537.

[60] Lakshmi Padmaja Maddali, Meena Singh Dilip Thakur, R Vigneswaran, MA Rajan, Srujana Kanchanapalli, and Batsayan Das. 2020. VeriBlock: A novel blockchain framework based on verifiable computing and trusted execution environment. In *Proceedings of the 2020 International Conference on Communication Systems & NetworkS (COMSNETS'20)*. IEEE, 1–6.

[61] Maidsafe. 2018. Maidsafe whitepaper. Retrieved from https://github.com/maidsafe/Whitepapers/blob/master/Project-Safe.md.

[62] Panpan Meng, Chengliang Tian, and Xiangguo Cheng. 2019. Publicly verifiable and efficiency/security-adjustable outsourcing scheme for solving large-scale modular system of linear equations. *J. Cloud Comput.* 8, 1 (2019), 1–13.

[63] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. 2014. Permacoin: Repurposing bitcoin work for data preservation. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP'14)*. IEEE, 475–490.

[64] Tal Moran and Ilan Orlov. 2019. Simple proofs of space-time and rational proofs of storage. In *Proceedings of the Annual International Cryptology Conference*. Springer, 381–409.

[65] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[66] Henning Pagnia and Felix C. Gärtner. 1999. *On the Impossibility of Fair Exchange without a Trusted Third Party*. Technical Report. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany.

[67] Daniele Perito and Gene Tsudik. 2010. Secure code update for embedded devices via proofs of secure erasure. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, 643–662.

[68] Ioannis Psaras, Vasilis Sourlas, Denis Shtefan, Sergi Rene, Mayutan Arumaithurai, Dirk Kutscher, and George Pavlou. 2017. On the feasibility of a user-operated mobile content distribution network. In *Proceedings of the IEEE 18th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'17)*. IEEE, 1–9.

[69] Paul Resnick and Richard Zeckhauser. 2002. Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. In *The Economics of the Internet and E-commerce*. Emerald Group Publishing Limited, 127–157.

[70] Amitabh Saxena, Janardan Misra, and Aritra Dhar. 2014. Increasing anonymity in bitcoin. In *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, 122–139.

[71] Jianjun Sun, Jiaqi Yan, and Kem ZK Zhang. 2016. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financ. Innov.* 2, 1 (2016), 26.

[72] Eric Ke Wang, Zuodong Liang, Chien-Ming Chen, Saru Kumari, and Muhammad Khurram Khan. 2020. PoRX: A reputation incentive scheme for blockchain consensus of IIoT. *Fut. Gener. Comput. Syst.* 102 (2020), 140–151.

[73] Huaqun Wang, Debiao He, Anmin Fu, Qi Li, and Qihua Wang. 2019. Provable data possession with outsourced data transfer. *IEEE Trans. Serv. Comput.* (2019).

[74] Nico Weichbrodt, Anil Kurmus, Peter Pietzuch, and Rüdiger Kapitza. 2016. AsyncShock: Exploiting synchronisation bugs in intel SGX enclaves. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, 440–457.

[75] Johannes Winter. 2008. Trusted computing building blocks for embedded linux-based ARM trustzone platforms. In *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing*. ACM, 21–30.

[76] Gavin Wood. 2016. Ethereum: A Secure Decentralised Generalised Transaction Ledger EIP-150 Revision. Retrieved August 9, 2017 from http://gavwood.com/paper.pdf.

[77] Dawei Xie, Haining Yang, Jing Qin, and Jixin Ma. 2019. Privacy-preserving and publicly verifiable protocol for outsourcing polynomials evaluation to a malicious cloud. *Int. J. Dig. Crime Forens.* 11, 4 (2019), 14–27.

[78] Hao Yan, Jiguo Li, and Yichen Zhang. 2019. Remote data checking with a designated verifier in cloud storage. *IEEE Syst. J.* 14, 2 (2019), 1788–1797.

[79] Xiao Zhang, Shengli Liu, and Shuai Han. 2019. Proofs of retrievability from linearly homomorphic structure-preserving signatures. *Int. J. Inf. Comput. Secur.* 11, 2 (2019), 178–202.

[80] Hui Zheng, Jun Shao, Guiyi Wei, Li Hu, Bianjing Pan, Kai Liu, and Xiaohang Mao. 2019. Attribute-based encryption with publicly verifiable outsourced decryption. In *Proceedings of the International Conference on Network and System Security*. Springer, 552–566.

Received November 2019; revised August 2020; accepted August 2020