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

Michał Król, Alberto Sonnino, Mustafa Al-Bassam, Argyrios Tasiopoulos, Ioannis Psaras
University College London, United Kingdom

***Abstract*—As cryptographic tokens and altcoins are increasingly being built to serve as utility tokens, the notion of useful work consensus protocols, as opposed to number-crunching PoW consensus, is becoming ever more important. In such contexts, users get rewards from the network after they have carried out some specific task useful for the network. While in some cases the proof of some utility or service can be proved, the majority of tasks are impossible to verify. In order to deal with such cases, we design "Proof-of-Prestige" (PoP)—a reward system that can run on top of Proof-of-Stake blockchains. PoP introduces "prestige" which is a volatile resource and, 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. PoP is resistant against Sybil and Collude attacks and can be used to reward workers for completing unverifiable tasks, while keeping the system free for the end-users. We use two exemplar use-cases to showcase the usefulness of PoP and we build a simulator to assess the cryptoeconomic behaviour of the system in terms of prestige transfer between nodes.**

I. Introduction

Following the recent success of Bitcoin [1], a plethora of cryptocurrencies have experienced an increase of popularity [2]. There are over 1,900 cryptocurrencies one can invest in with the total market cap exceeding 280B USD[1]. In contrast to resource-wasting Proof-of-Work (PoW) protocols [1] or Proof-of-Stake (PoS) binding users' minting power to the amount of their coins [3]–[5], a recent trend sees cryptocurrencies as an incentive method for users to perform useful work and create a shared economy environment. For instance, Filecoin [6], [7] rewards miners for renting storage capacity, while Golem [8] allows to rent out processing power and perform client's computation-heavy tasks. The vision is to create a decentralised system, where miners are incentivised 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] [10]. 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, in order to attract more users, blockchain-based platforms must keep this featue 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 and rewards contributors' useful work, while keeping it free for the end-users. For instance, on Steem [11] 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 in order to increase its income from advertisements, rewards the most successful authors with an automatic coin transfer.

Such a system presents multiple benefits. Beneficiaries do not have to pay for the content, the system remains open for any contributor to join, perform useful work and be paid according to their performance and contribution; the motivator on the other hand benefits from an open platform avoiding costly contracts, and contributor selection process. However, for that to work, the network must be able to automatically verify tasks. While completion of some tasks can be proven to a third party (*e.g.,* file storage [7]), 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. In order 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 stake, users can collude and cross-acknowledge their potentially non-existing work.

In this paper, we present Proof-of-Prestige (PoP) that aims to be a building block of cryptocurrencies based on useful work. 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 wide variety of use-cases, securing the system against Colluding and Sybil attacks and avoiding artificial inflation of miners' power by completing dummy tasks. 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 already existing PoS protocols. Proof-of-Prestige, does not deal with the fair exchange problem between beneficiary and benefactor [12], but rather offers a secure way to reward benefactors, by network operators. Our system builds on Proof-of-Stake and can be easily integrated in a distributed ledger using the latter (Section II). We present two variants of our system, simple and progressive mining, and show how they can be used in real-world scenarios.

[1]https://www.investing.com/crypto/currencies

We make the following contributions:

- Section III presents Proof-of-Prestige (PoP), a novel type of proof of useful work that can run on top of any Proof-of-Stake blockchain, and that is resilient against Sybil and Collude attacks.
- Section III-C and Section III-D introduce two types of minting power distribution, simple and progressive mining, to support a wide variety of scenarios.
- Section IV presents and analyzes two key use-case applications, a publisher platform and a file distribution system, that benefit from our framework.
- Section V provides an extensive evaluation of our protocol and prove its security and high performance.

II. Threat Model and Goals

We assume the following actors in our system:

- **Beneficiaries:** End-users of the system which transfer prestige to the contributors in exchange for a task to be performed (such as distributing a file).
- **Contributors:** Nodes which perform tasks for beneficiaries in return for prestige.
- **Motivators:** System operators or users that submit tasks to the network. Motivators do not benefit from services directly, but rather from expanded network size (*i.e.,* crowdsourcing their own tasks).

Each user is represented by an identity and can act as contributor, beneficiary and/or motivator. We assume 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.

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 some defined period of time.

Proof of Prestige has 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 incetivisation.* 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.
- *Collude attack resistance.* Users cannot increase their total amount of prestige by colluding with each others.

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 [13], Tendermint [14], Hashgraph [15]). As prestige is a volatile resource, it cannot be used with PoS protocols which require stakers to deposit and lock their coins for a long period of time (such as Casper FFG [16]), so that 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. However, it is compatible with protocols such as Ouroboros [5] which do not require deposits, and adapt to the new stake mapping in the system every epoch, where coins (and prestige) do not have to be locked up and can be transferred at any time.

III. Proof of Prestige

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

- **Coins** - similar to other crypto-currencies (such as Bitcoin or Ether) or ERC20 tokens [17]. Coins can be directly sent to or received by other users via transactions, as is the case with all crypto-currencies. More importantly, *coins generate prestige over time.*
- **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. Contributors gain prestige by completing tasks, and beneficiaries lose prestige by acknowledging services.

Both values are related and can influence one another. The coins generate prestige over time up to a *Static Value* (see Section III-A), 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 thus increase their total amount of coins. Prestige represents a much more volatile and renewable resource, while the amount 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.*

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 amount 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, higher probability of minting new coins), this does not exclude users with less coins from building up prestige if they contribute to the system with useful work.

A. Prestige

When user U_i joins the system (*i.e.,* creates their identity), they start 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 their wallet C_i, so that richer users generate prestige at higher rate. At the same time, in order 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 S_i we have that $S_i = g(S_i)$:

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

In fact, $|g'(S_i)| < 1$, *i.e.*, $g'(\cdot) = 1 - d < 1$, and therefore S_i is an attractor fixed point indicating the convergence of prestige DSS to S_i. The amount of prestige that a user can generate from coins is thus limited to S_i—called *Static Value*—where increasing decay evens up prestige generated from coins (Figure 1). The static 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 Static 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 1) and therefore, increase their chances of minting the new block. In turn, minting a new block results in extra coins and thus, in higher *Static Value* (see User U_2 in Figure 1).

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 static value, the decay exceeds prestige gained from coins and the user loses prestige until they reach their static value again. The reduction in prestige is another desired property, as it incentivises 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 static values (Figure 1).

Our system needs to be secure and resistant against Sybil and Collude Attacks (Section II). 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 her coins into multiple identities.*

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

$$S_{i,1} + S_{i,2} + ... + S_{i,k} = \sum_{q=1}^{k} C_{i,q}/d = C_i/d = S_i.$$

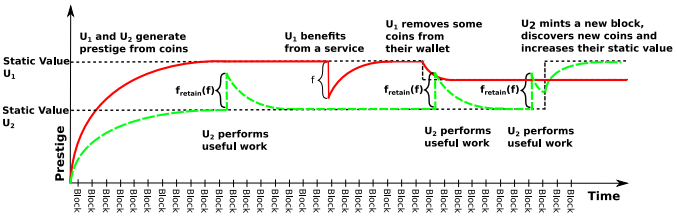

Fig. 1: Example of evolution of the prestige and static value with time (expressed in blocks).

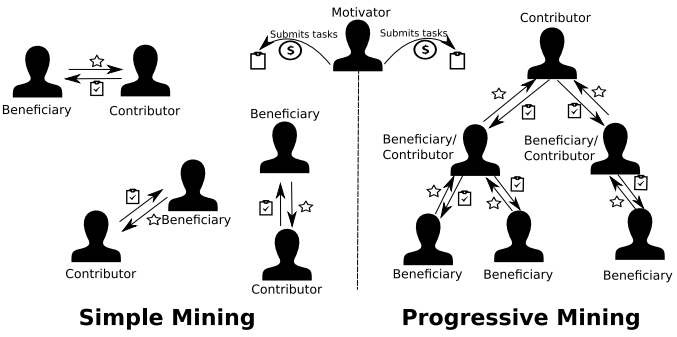

Fig. 2: Simple and Progressive Mining.

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

B. 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.

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 $x_{gained}^i = \sum_{j=0}^{m} f_{retain}(x_{ji})$. 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 III-E). 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 lower or equal to the prestige retained by the contributor U_j, so that $x_{ij}^k \geq f_{retain}(x_{ji})^k$, 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 Collude Attacks.*

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.

C. 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 transfered prestige f so that $f = x_{ij}^t = -x_{ji}^t$ (Figure 2). 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 Collude Attacks.

D. 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, U_i will receive some prestige for each service performed by U_j and other nodes that U_j provided the service to (Figure 2). 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 towards the root.*

This type of rewarding scheme is useful in scenarios such as file propagation, where receiving a file allows to distribute it 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. P_b is the sum of the prestige values of the predecessors of P_i multiplied by a branch power parameter b:

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

Such a branch power function incentivises 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 towards the root. If $P_b^i = 0$, 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 towards 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.*

Proof. Without loss of generality assume that user i divides her coins into 2 identities according to $C_{i,1}$ and $C_{i,2}$, such that $C_{i,1} + C_{i,2} = C_i$. hen, 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:

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

On the other hand, 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. Hence, users cannot increase their prestige by creating multiple identities in progressive mining, intuitively due to the prestige payments that is forced to submit to her fake identities that in turn they retain only a portion of the prestige.

□

E. 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 III-C), 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 III-D), acknowledgments are composite signatures [18]. 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.

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 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 their 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 in order 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. In order to achieve this, the contributor transmits their 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 their predecessors do not do it. The properties of composite signatures prevent any single or subset of signature(s) from being removed from the composite [18]. 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], [12], [19], a specific solution should be tailored to the nature of performed tasks. We thus sketch some solutions in Sec. IV and leave more detailed discussion for future work.

F. 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, in order to incentivise 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 (DoS) attacks, but also increase the exploitation cost of the system. On the other hand, 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., 1000 blocks). Motivators can thus incentivise users to participate in a specific task when their services are needed the most.

IV. Use Cases

We present two applications that leverage PoP to support a reliable and secure incentive mechanism: a Publisher Platform

as a system that uses simple mining, and a File Distribution system as a system that needs progressive mining.

A. Simple mining - Publisher Platform

Steem [11] is a popular publishing platform that uses a combination of Proof-of-Brain and Delegated Proof-of-Stake [20] to reward content creators (contributors) and readers (beneficiaries). Readers can vote for interesting content using their Steem Power acquired by committing their coins to a thirteen-week vesting schedule. The best creators and readers are then rewarded by newly created coins. Steem is thus resistant to Sybil attacks (fake identities do not have coins and thus cannot acquire Steem Power), but is vulnerable to collude attacks. As users can recommend unlimited number articles, they are incentivized to cross-recommend their work with other users in order to maximize their reward.

Introducing our PoP with Simple Mining into Steem could solve this problem. Replacing Steem Power with prestige, introduces a limit on the number of content items that readers can vote for within a specified period of time. Increased amount of coins replenishes prestige faster so richer users have still higher voting power as in the current scheme. 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 in order to receive high quality content in the future.

B. Progressive mining - File Distribution

We base our File Distribution use case on already existing systems requiring a secure rewarding scheme for file propagation such as Filecoin/IPFS [6] or NOIA [21]. Currently those systems either do not provide incentives for file propagation (IPFS), or are vulnerable to Sybil attacks (NOIA). A content creator (acting as a motivator) wants to distribute and prefetch large video files among mobile phone users at the edge of the network in order to increase its viewership. In the current/traditional client-server model, content is pulled from the Content Delivery Network (CDN) server upon the user's request. Using CDNs, however, involves substantial fees and scales badly when crossing the mobile data link. In our File Distribution system, users themselves contribute to the distribution of the content directly between mobile devices without involving the network or third party CDNs. Effectively, users participate in an *"incentivised content propagation network"*, where they get rewarded for contributing their resources to the network. At the same time, content creators/publishers can significantly reduce the cost of content distribution.

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 2). The propagation then continues directly between user devices expanding the DAG. Each time a user receives a file, he acts as beneficiary and must generate an acknowledgment for the contributor who sent the file. In order 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.

This type of content propagation is a typical use-case for progressive mining: 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.

V. Evaluation

In order to evaluate the behaviour of Proof-of-Prestige, we developed a Python3 simulator that we will release as open-source software. In the following, we investigate a number of factors that influence users' prestige fluctuation and present one scenario for each of those factors.

Decay Parameter, d: We start by investigating the behaviour of the prestige correction function shown in Equation (1). Figure 3 shows the influence of the decay parameter d on prestige gain. We create 4 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 static value (Equation (3)), but the number of coins does not influence the time needed to converge back to the static value. On the other hand, increasing the decay parameter d lowers the static value and reduces the time required by each user to reach their static 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 static 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 4 introduces 4 users with the same amount of coins $C = 100$ and prestige set to the corresponding static value $P_i = S_i$. 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 (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).

Distance from the DAG root: We create 1000 users involved in 100 random DAGs, where each edge represents performing useful work between nodes. Initial prestige values follow a uniform distribution from 0 to 100, while the branch power parameter is set to $b = 0.5$. Figure 6 presents the average prestige gained by each user and standard error as a function of distance from the DAG root. 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 a desired behaviour, 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, distance fom the root) which results in increased standard error.

Base Prestige: We investigate the prestige gain as a function of base prestige (Figure 7). Simple mining is not influenced

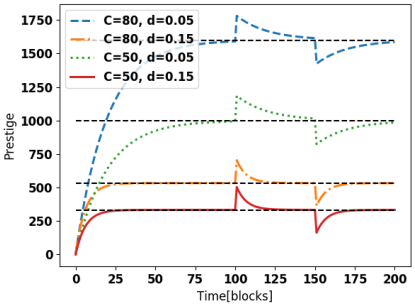

Fig. 3: Prestige over time.

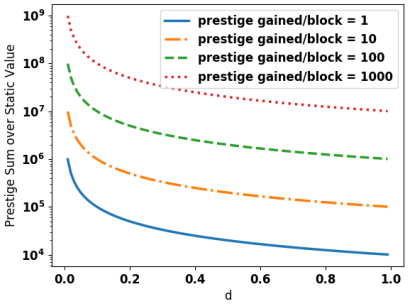

Fig. 4: Prestige sum above static value.

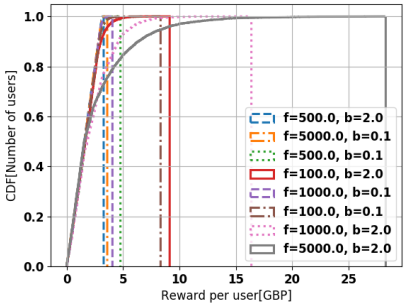

Fig. 5: Reward Distribution.

by base prestige, while in progressive mining, with increasing base prestige, a user can collect higher rewards. This mechanism prevents the Sybil attack and rewards users who invested more in the system. However, a small fraction of high base prestige users experience 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 behaviour.

Number of Completed Tasks: Next, we investigate the effect of useful work to user prestige gain (Figure 8). 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.

Contributor Involvement Probability: We investigate prestige dynamics over time with both simple (Figure 9) and progressive (Figure 10) mining to provide a global view of the system. We 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, prestige values of each user go from 0 towards their corresponding static 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 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 11). For small values of d, useful work is more important than money. Poorer, but more active users, can thus acquire 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 from 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 IV-B and apply it to popular a BBC Series - *Bodyguard*. BBC does not include advertisement in their content — thus, each consecutive download increase the broadcaster's cost. With the number of viewers ranges from 14M to 17M[2], and the file size of approximately 250MB, the cost of delivering one series season using a CDN equals 4.7M$[3]. For each episode of the series, we create a DAG with the corresponding size of users with random amount of base prestige ranging from 1 to 10000 and distribute the money spent on CDN to users proportionally to their prestige after performing useful work. In this scenario users transfer files to maximum 8 users. Figure 5 shows the reward distribution for different values of f and b. Surprisingly, those parameters have a negligible effect on the majority of the nodes. With changing parameters, user receive, but also transfer upstream different amounts of prestige. Such a behaviour influences mainly the most active nodes located close to the DAG root. With high f and b those nodes can acquire significantly higher amounts of prestige and thus collect higher rewards. The most active users collect up to 30$ reward, while the distribution remains free for all the users.

Volume of Data Submitted to the Blockchain: Finally, we approximate the amount of data submitted to the blockchain, which varies between simple and progressive mining. 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 102 bytes; it is computed as the sum of the size of the composite signature[4] (33 bytes), the task ID (32 bytes), the contributor public key (33 bytes), and the amount of prestige transferred (set to 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, $(33+69\times n_i)$ bytes, where n_i is the depth of the DAG following path i. While this represents a substantial amount of data, acknowledgments can be processed off-chain using platforms such as Plasma [23] and update users' prestige periodically,

[2]https://www.barb.co.uk/viewing-data/four-screen-dashboard/
[3]http://cdncomparison.com/
[4]We implement composite signatures with BGLS signatures [22].

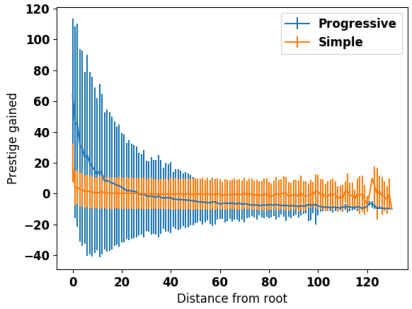

Fig. 6: Distance from root

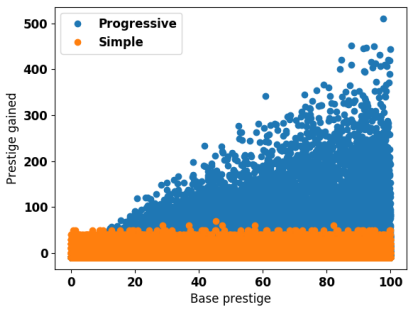

Fig. 7: Base Prestige

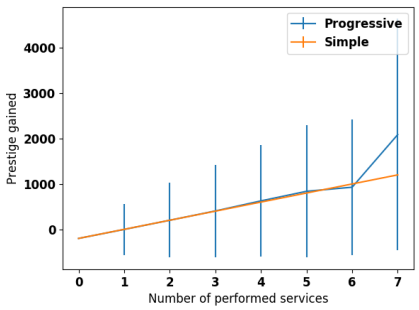

Fig. 8: Useful work

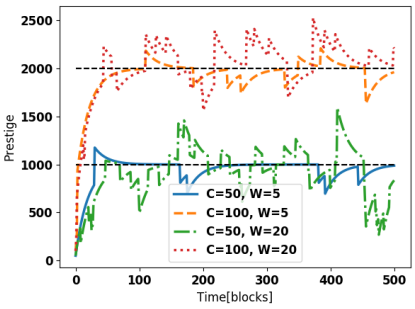

Fig. 9: Prestige evolution for work Simple Mining

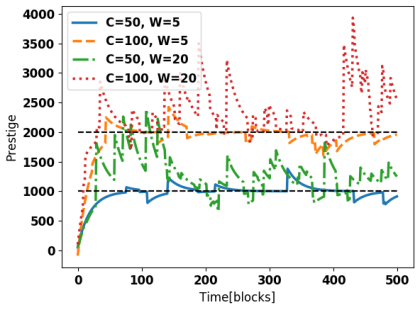

Fig. 10: Prestige evolution for work Progressive Mining

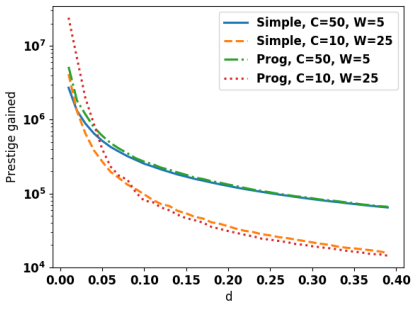

Fig. 11: Prestige sum acquired by different users.

significantly decreasing the volume of information kept on the main-chain.

VI. Related Work

There are currently multiple systems focusing on rewarding miners for useful work. The largest group focuses on prooving file storage where a verifier sends a file to a prover and later requests a proof that the prover really stored the file [7], [24]–[30]. Alternatively, several platforms allow the prover to convince that the prover has access to some space [31] [32] [33] [34] [35] [36]. Additionally one can require that the proof implies that the space also was erased [33], [35], or some function can only be computed in forward direction [34]. In all those cases, the network must be able to reliably verify the completed task to reward miners which narrows the scope of supported tasks to a small group. Some solution replace traditional Proof-of-Work with useful mathematical tasks that are easy to verify such as poynomials evaluation [37], [38]. However, such systems support only one type of tasks and does not accept custom ones requested by users.

Another family supports broader range of tasks (such as custom computation tasks) [9] [10] [39] [40], but relies on Trusted Execution Environments (TEEs) and Remote Attestation Protocols [41] [42] to verify that the computations are being run on a genuine platform and the results are correct. However, TEEs are not available on every platform, require users to trust hardware vendors and are susceptible to side channel attacks [43] [44]. In contrast, PoP does not make any assumptions on users hardware. Some open platforms rely on centralized 3rd parties acting as consents to validate tasks or perform conflict resolution [8] [45] [46]. However, for those system to work correctly, the 3rd parties must be trusted by all network participants. Such an approach contrasts with the idea of open and trustles system lying behind blockchain and exposes the network to multiple colluding attacks.

Finally, there exists 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 3-rd party making those system susceptible to Sybil attacks. Such tasks include content creation [11], content distribution [21] or providing hardware for game players [47]. PoP can be a valuable addition to those system allowing to reliably reward users for truly performed tasks.

VII. Conclusion

We presented *Proof-of-Prestige* (PoP)—a reward system that can run on top of any Proof-of-Stake blockchain. We introduce the notion of *Prestige* that is a volatile and renewable resource, 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 their 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 Collude 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.

We presented two variants of our scheme—simple and progressive mining, and showed how they can be used in real-world scenarios. 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 incentivises users to perform useful work.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," *arXiv preprint arXiv:1711.03936*, 2017.

[3] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake.," *IACR Cryptology ePrint Archive*, vol. 2016, p. 919, 2016.

[4] B. David, P. Gazi, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol," *IOHK paper*, 2017.

[5] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*, pp. 357–388, Springer, 2017.

[6] Y. Combinator, "June 30th 2017."ipfs, coinlist, and the filecoin ico with juan benet."."

[7] P. Labs, "Proof of replication technical report," tech. rep., 2017.

[8] "Golem whitepaper." https://golem.network/doc/Golemwhitepaper.pdf, 2016.

[9] M. Al-Bassam, A. Sonnino, M. Król, and I. Psaras, "Airtnt: Fair exchange payment for outsourced secure enclave computations," *arXiv preprint arXiv:1805.06411*, 2018.

[10] M. Król and I. Psaras, "Spoc: Secure payments for outsourced computations," *Proc. NDSS Workshop on Decentralized IoT Security and Standards, San Diego, CA*, 2018.

[11] "Steem whitepaper." https://steem.io/, 2018.

[12] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 967–984, ACM, 2018.

[13] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, ACM, 2017.

[14] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, 2014.

[15] L. BAIRD, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," 2016.

[16] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *CoRR*, vol. abs/1710.09437, 2017.

[17] "Erc-20 token standard." https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md, 2015.

[18] A. Saxena, J. Misra, and A. Dhar, "Increasing anonymity in bitcoin," in *International Conference on Financial Cryptography and Data Security*, pp. 122–139, Springer, 2014.

[19] H. Pagnia and F. C. Gärtner, "On the impossibility of fair exchange without a trusted third party," tech. rep., Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany, 1999.

[20] "Steem delegated proof of stake." https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper, 2018.

[21] "Noia whitepaper." https://drive.google.com/file/d/1IfdKbai7hkScw_Zj6-kbZPoxNCmQzKaR/view, 2018.

[22] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Eurocrypt*, vol. 2656, pp. 416–432, Springer, 2003.

[23] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, 2017.

[24] P. Golle, S. Jarecki, and I. Mironov, "Cryptographic primitives enforcing communication and storage complexity," in *International Conference on Financial Cryptography*, pp. 120–135, Springer, 2002.

[25] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 43–54, ACM, 2009.

[26] R. Di Pietro, L. V. Mancini, Y. W. Law, S. Etalle, and P. Havinga, "Lkhw: A directed diffusion-based secure multicast scheme for wireless sensor networks," in *Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on*, pp. 397–406, IEEE, 2003.

[27] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 598–609, Acm, 2007.

[28] A. Juels and B. S. Kaliski Jr, "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 584–597, Acm, 2007.

[29] Maidsafe, "Maidsafe whitepaper." https://github.com/maidsafe/Whitepapers/blob/master/Project-Safe.md, 2018.

[30] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *2014 IEEE Symposium on Security and Privacy (SP)*, pp. 475–490, IEEE, 2014.

[31] T. Hønsi, "Spacemint-a cryptocurrency based on proofs of space," Master's thesis, NTNU, 2017.

[32] "Chia network." https://chia.network, 2018.

[33] D. Perito and G. Tsudik, "Secure code update for embedded devices via proofs of secure erasure," in *European Symposium on Research in Computer Security*, pp. 643–662, Springer, 2010.

[34] S. Dziembowski, T. Kazana, and D. Wichs, "One-time computable self-erasing functions," in *Theory of Cryptography Conference*, pp. 125–143, Springer, 2011.

[35] N. P. Karvelas and A. Kiayias, "Efficient proofs of secure erasure," in *International Conference on Security and Cryptography for Networks*, pp. 520–537, Springer, 2014.

[36] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi, "Proofs of space: When space is of the essence," in *International Conference on Security and Cryptography for Networks*, pp. 538–557, Springer, 2014.

[37] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of useful work.," *IACR Cryptology ePrint Archive*, vol. 2017, p. 203, 2017.

[38] X. Hu and C. Tang, "Secure outsourced computation of the characteristic polynomial and eigenvalues of matrix," *Journal of Cloud Computing*, vol. 4, no. 1, p. 7, 2015.

[39] "iexec whitepaper." https://iex.ec/whitepaper/iExec-WPv3.0-English.pdf, 2018.

[40] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pp. 282–297, Springer, 2017.

[41] V. Costan and S. Devadas, "Intel sgx explained.," *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[42] J. Winter, "Trusted computing building blocks for embedded linux-based arm trustzone platforms," in *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pp. 21–30, ACM, 2008.

[43] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel sgx," in *Proceedings of the 10th European Workshop on Systems Security*, p. 2, ACM, 2017.

[44] N. Weichbrodt, A. Kurmus, P. Pietzuch, and R. Kapitza, "Asyncshock: Exploiting synchronisation bugs in intel sgx enclaves," in *European Symposium on Research in Computer Security*, pp. 440–457, Springer, 2016.

[45] "Sonm documentation." https://docs.sonm.com/, 2018.

[46] A. Angelo, P. Thellmann, and D. Dalkilic, "Rewarding the token economy.," 2018.

[47] "Playkey whiteaper." https://cdn.playkey.net/img/playkeynet/ico/Whitepaper_1_31_En.pdf, 2018.