Trust Mends Blockchains: Living up to Expectations

Leila Bahri
Royal Institute of Technology (KTH)
Stockholm, Sweden
lbahri@kth.se

Sarunas Girdzijauskas
Royal Institute of Technology (KTH)
Stockholm, Sweden
sarunasg@kth.se

***Abstract*—At the heart of Blockchains is the trustless leader election mechanism for achieving consensus among pseudo-anonymous peers, without the need of oversight from any third party or authority whatsoever. So far, two main mechanisms are being discussed: proof-of-work (PoW) and proof-of-stake (PoS). PoW relies on demonstration of computational power, and comes with the markup of huge energy wastage in return of the stake in cyrpto-currency. PoS tries to address this by relying on owned stake (i.e., amount of crypto-currency) in the system. In both cases, Blockchains are limited to systems with financial basis. This forces non-crypto-currency Blockchain applications to resort to "permissioned" setting only, effectively centralizing the system. However, non-crypto-currency permisionless blockhains could enable secure and self-governed peer-to-peer structures for numerous emerging application domains, such as education and health, where some trust exists among peers. This creates a new possibility for valuing trust among peers and capitalizing it as the basis (stake) for reaching consensus. In this paper we show that there is a viable way for permisionless non-financial Blockhains to operate in completely decentralized environments and achieve leader election through proof-of-trust (PoT). In our PoT construction, peer trust is extracted from a trust network that emerges in a decentralized manner and is used as a waiver for the effort to be spent for PoW, thus dramatically reducing total energy expenditure of the system. Furthermore, our PoT construction is resilient to the risk of small cartels monopolizing the network (as it happens with the mining-pool phenomena in PoW) and is not vulnerable to sybils. We evluate security guarantees, and perform experimental evaluation of our construction, demonstrating up to 10-fold energy savings compared to PoW without trading off any of the decentralization characteristics, with further guarantees against risks of monopolization.**

***Index Terms*—Blockchain, Proof of Work, Bitcoin, Distributed Ledger, PoW is expensive, Proof of Trust, PoW alternative**

I. Introduction

Blockchain technology appeared with the decentralized crypto-currency known as Bitcoin and proved that money could be created and exchanged within pure peer-to-peer (P2P) environments without the need of any form of centralized authority or trusted entity. More interestingly, Blockchain made this possible without the need for managing or certifying identities. The solution was an arrangement of techniques from cryptography along with rules and guidelines from diverse domains, such as distributed systems, economics, and game theory; all put together to form a functioning self-sustained decentralized currency system managed by the open public. Among all the pieces that form the technology, the heart of Blockchains is the consensus mechanism that provides a consistent state of the system among trustless and anonymous[1] peers.

Transactions in a Blockchain system are publicly announced to all the participating peers who are responsible for verifying the transactions they hear of and for adding the valid ones to a shared ledger. This shared ledger is organized in the form of blocks that are chronologically ordered and cryptographically chained to each other to generate a tamper-proof data structure called blockchain.[2] Managing this blockchain of transactions in a pure P2P environment, with the Byzantine nature that it imposes, requires a consensus mechanism to agree at each new epoch on the next block to extend the blockchain and on who will propose it.

In Bitcoin Blockchain system, as well as in most operational Blockchains available so far[3], consensus is achieved by a cryptographic hash puzzle that allows sealing blocks together and, even more importantly, that enables agreement on the next block that will be extending the blockchain. The cryptographic hash puzzle consists in generating a hash number that is smaller than a given agreed upon value in the Blockchain network, known as the difficulty level. The characteristics of cryptographic hashing provide both a tamper-proof quality to the blockchain, as every new block includes a hash of the previous block it extends, as well as a computationally intensive puzzle to produce (by chance) a hash number that is smaller than the given difficulty level. This enables a randomized *leader election* in which every peer is forced to prove computational capacity, and leaders are selected proportionally to the amount of work they performed. Indeed, at every new epoch, the first peer to solve the puzzle would have demonstrated a *proof-of-work (PoW)*, that the other nodes can easily verify, and based on which the peer is accepted as the leader that proposes the next block. PoW also provides an implicit defense against sybils by ensuring the "one CPU-one vote" rule.

PoW is very costly. Unfortunately, PoW is an energy exhaustive mechanism, where the difficulty level is adjusted based on the computational capacity of the system, thus as more nodes

[1]We refer here to public Blockchains where identities are not managed. Although such systems are technically anonymous, they practically can only guarantee pseudo-anonymity depending on usage patterns and peers practices.

[2]We differentiate between Blockchain, with big B, as the technology that comprises all the components of the system in terms of rules, mechanisms, protocols, etc., and between blockchain which refers to the ledger of chained blocks of transactions.

[3]e.g., Ethereum Blockchain

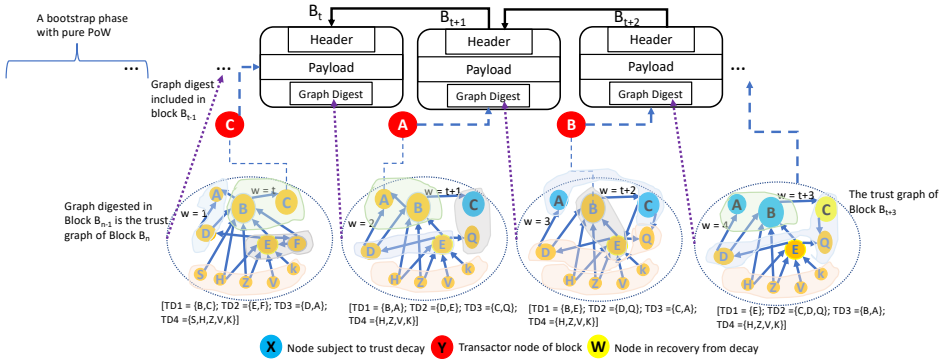

Fig. 1. An overview of the proposed PoT construction with its basic elements: first, a trust network is built based on trust links (outgoing edges) explicitly expressed by each node. At every epoch, the constructed version of the network is committed to the blockchain using a secure digest. New edges are subject to a damping strategy and are weighted based on their age, where w=0 when an edge first appears in a block version and is incremented as it continues appearing in subsequent block versions. A trust metric is computed for every node based on the emerged trust network. Nodes are assigned to trust divisions (TD) based on their achieved trust measure. Members of each trust division are entitled to run a low complexity PoW proportional to their trust. When a node is selected as the epoch's leader, its trust is subject to an exponential decay for the subsequent blocks with recovery proportional to its starting trust value.

come with more CPUs, cost per transaction inevitably grows. For instance, in 2016, it was estimated that Bitcoin miners had been mining at an average rate of 10^{18} hashes per second. In less than a year span, this rate has increased to 10^{21} order of magnitude.[4]

In addition to its prohibitive cost, putting PoW in practice has shown that it is easily vulnerable to the trap of centralization where a small elite of powerful/rich peers end-up dominating the system. This is demonstrated today in Bitcoin by the mining pools phenomena, and the giant mining data-centers that monopolized the system, effectively disincentivising any further participation of individual peers [1].

Alternatives to PoW. These downsides of PoW has raised concerns about its sustainability, as well as motivated the investigation of other more energy efficient alternatives that would provide similar or better security and decentralized consensus guarantees. However, most of the suggested alternatives[5] rely either on the ownership of special physical hardware (e.g. [2], [3], [4]), or trade-off the decentralization of the system towards a permissioned Blockchain model with consensus aciheved by variants of Practical Byzantine Fault Tolerance (PBFT) protocols, such as in [5] [6]). One of the plausible alternatives to PoW is the concept of Proof-of-Stake (PoS) [7]. In PoS block leaders are selected based on the amount of stake they own in the Blockchain system. The game theoretical premise behind the approach is that peers with the largest share, in terms of owned crypto-coins, have interest into better serving the system to maintain its credibility and value, hence incentivized to behave according to the protocols. PoS promises a cleaner and much cheaper consensus mechanisms compared to PoW, however it requires binding Blockchain systems to a crypto-currency. Besides, it boosts the rich-gets-richer effect with a mindset of, "the more you own the more you are in charge", or "the more you own, the more legitimacy you have for leadership", further shifting the system towards centralization.

[4]See the graph of hash rates statistics of Bitcoin over the years at: https://blockchain.info/charts/hash-rate?timespan=2years

[5]Most alternatives are discussed in forums and a few only have been formalized in research papers.

Blockchains hit a dead-end. Although currently there is a surge in interest in Blockchain technologies from different domains with aspirations for establishing more decentralized systems, Blockchain seems to hit a dead-end with reliance on crypto-currencies as its main incentive and basis for security. The excessively costly PoW results in expensive coins that prohibit usage within non-financially incentivized domains. While PoS sounds good in theory, to date, there is no operational Blockchain ledger based on PoS alone without combining it with PoW or some PBFT solution, not least because of the aforementioned centralization effects as well as reliance on the necessity to have an underlying crypto-currency. Finally, there are frantic attempts to justify blockchain usage with permissioned model which in effect undermines the whole decentralized nature of the system, shifting it back to the centralized territory. If Blockchain technology is to apply to wider application domains and break out from this dead-end, a new capacity is required to overcome the aforementioned challenges. There is a pivotal need for thinking outside of the box and discovering new consensus mechanisms for Blockchains that are cheaper and more energy efficient than PoW, that do no rely on ownership of any specialty hardware, that do not trade off the permissionless virtue, and that are not bound to a crypto-currency for their operation.

Trust is the light at the end of the tunnel. Trust is inherent to any collaborating environment, as peers can form opinion about each other through time based interactions and participation in shared undertakings [8]. Trust is expensive to acquire, but is also easy to lose, thereby providing a powerful and robust incentive mechanism to maintain correct behavior in the system, especially in environments where participation in the system is incentivized by a common goal and is not linked to monetary gains as in crypto-currency based models. Yet, when decentralization is properly and fully considered, such as the in premissionless Blockchain model, trust is no more discussed as an option and *trustless* is implied as if a necessary ingredient to maintain decentralization. If trust can emerge and be managed in a purely decentralized setting, then this implication does not need to be true. Research from domains such as mobile ad-hoc networks already provides the

answer, with decentralized reputation management schemes [9]. We argue that decentralization does not need nor require trustlessness.

Mending Blockchains with trust. We propose to capitalize on natural collaborative trust as a form of stake that peers build through their participation in the public Blockchain system, without the need of underlying crypto-currency assumption. We demonstrate the usage of a proof-of-trust (PoT) metric as a waiver for the amount of work that peers need to demonstrate for leader election. The objective is to minimize the amount of energy spent on PoW as more trusted peers appear in the network. That is, install a concept of "the more trusted you are, the less work you are required to perform." We assume application scenarios where Blockchain is intended to provide a decentralized service among peers collaborating to achieve a common goal and where trust can be expressed among parties, such as in the health or education domains. This stands as a stark contrast to current ego-centric financial blockchains where the sole interest and game-theoretic incentive of being elected as a block leader is to earn crypto-currency. We have already laid out the vision and initialized discussions around the challenges towards achieving this goal in [10], without going into details of any viable solution. In this paper, we address the challenges towards achieving these goals and materialize a PoT construction as a Blockchain consensus viable solution based on naturally emerging trust in the community.

In *our PoT construction* (see overview on Figure 1), every peer participating in the system individually indicates the trust towards some other peers it deems trustworthy. The trust is continuously announced giving the rise of a *trust network* that emerges in a decentralized manner and is stored on the blockchain itself, providing a consistent and tamper-proof view to all the participants. I.e, each peer can locally extract a trust metric from the trust network embedded in the blockchain, that will provide a consistent and unambiguous trust measure for any peer across all the network. At each epoch, this trust measure is used as a waiver for the difficulty of PoW to be solved by each of the peers. Therefore, the probability to be elected as the leader of a given epoch is proportional to the gained trust in the network. To prevent sybil attacks and to avoid top trusted peers from maintaining their rank throughout the epochs and thus monopolizing the system, a *decay penalty* is applied to each node right upon its election as an epoch leader with a recovery proportional to its starting trust value. Moreover, to further minimize the effect of sybil attacks, our protocol ensures that the trust network will never evolve in such a way which would allow newcomer nodes to reach the top-trusted ranks of the system within a predefined number of epochs, allowing timely detection of malicious behavior by the majority of the system players.[6] Our PoT mechanism while still using PoW, is assured to use PoW only by a small ranked set of participating peers, thus drastically reducing power costs. In the rest of the paper, we formalize our proposed construction and demonstrate how each of its components is designed to address the challenges of establishing a PoT Blockchain system.

Overall, we summarize our contributions as:

- *A PoT based consensus mechanism through trust networks for public, permisionless blockchains.* We define the emergence mechanism and maintenance of the trust network and we formalize a PoT protocol that relies on calibrating PoW by trust. The protocol offers resiliency against monopolizing the network by the elite and protection of the network against sybils.
- *Experimental evaluation of the PoT construction on real world trust networks of varying sizes.* We study how each of the parameters regulating PoT protocol affect the efficiency and security of the Blockchain with evolving trust networks of varying sizes. Our results demonstrate the viability of our PoT construction, achieving a decrease in the probability of monopolization by up to 80% as compared to PoW without trust. Furthermore, the construction is shown to limit the impact of sybils to less than 2% for more than 30 epochs even when they connect to honest nodes and represent 20% of the trust network.

The rest of this paper is organized as follows: in Section II we discuss PoT w.r.t decentralization with an overview of the issues and how we address them. In Section III we formalize our PoT construction based on the results drawn from Section II, and we define a PoT protocol that waives PoW based on trust. We provide experimental evaluation of the construction in Section IV. We finally conclude the paper in Section V with a discussion on future work and further research directions that this work opens.

II. Decentralization and consensus based on trust

The concept behind achieving a PoT Blockchain is to have a mechanism by which trust values of participating peers can be computed and agreed on in a decentralized fashion. These trust values will represent the stake that peers own in the system, and that could be used as part of the basis for achieving consensus.

PoW is energy exhaustive and is susceptible to centralization over time, but its decade of operation in Bitcoin has also made it the only consensus mechanism to prove itself in practice within completely open and premissionless P2P environments. In PoW, it is simple and easy to reach consensus; i.e., whoever proves more work is simply accepted by the others as the next leader. At equal importance, PoW also provides the ability to verify the resulting blockchain and to independently evaluate its validity. That is, anyone who gets a copy of a blockchain can easily verify its compliance to the protocol in place and know whether it is valid or not. In designing our PoT alternative, the main goal is not to trade off any of these decentralization qualities, that we formalize under two main properties, *verifiability* and *independence*:

Verifiability property. Verifiability states that once a peer proves its eligibility to be the next leader, all peers can verify

[6]We build on the results in [11] which suggest that controlled network evolution facilitates sybil detection in mobile ad hoc networks.

this proof. It also states that anyone who gets a version of the blockchain can verify its validity.

Independence property. Independence states that both leader eligibility and blockcain validty verification can be performed by any peer using publicly available information without the need for any other collaboration; hence independently of any other peer.

To achieve this, two main problems need to be addressed: 1) defining a mechanism for decentralized trust management; 2) defining a consensus protocol that uses trust network.

A. Decentralized trust management

Trust represents the importance of peers as perceived by some entity, which in our case is the open community. Building trust is a process which could be underatken under different models, and the literature already offers a variety of trust models (see for e.g., [12], [13], [14]). For community-based trust, the starting point is usually a given trust network that encodes who trusts whom in the system, and that is generally modeled as a directed graph of which the nodes represent the peers, and the edges represent the trust relationships between them. The more incoming links a node has in the network, the more trusted it is. Therefore, the first step is defining the model based on which the underlying trust graph will emerge.

Trust networks could emerge in a variety of ways, such as being extracted from a social network, inferred from the interactions between peers in the underlying system, being explicitly expressed by the peers announcing whom they trust, etc. Within a public permissionless model where identities are not centrally managed, the only viable options are either to track interactions between peers as recorded in the blockchain itself, or to allow a deliberate emergence of a trust network where every peer individually announces whom she trusts at every epoch. The first option requires that peers who manage the Blockchain system be also stakeholders in the transactions that take place and that are recorded in the blockchain. To not make any assumptions with that regard, in this paper we opt for the second option where *peers freely and individually express whom they decide to trus*t. Therefore, each peer announces their trusted peers in the form of outgoing trust links to them. These links are broadcasted in the network and are independently collected by all the other peers to form the resulting trust network. In order to have a unified and accessible version of the trust network, it should be stored in the blockchain itself. However, blockchains are known to suffer from size limitations imposed on blocks for network throughput considerations.[7] Therefore, our solution is based only on a secure digest of the trust network recorded in the blockchain, thereby achieving the desired result in a scalable fashion. As we explain in Section III below, an evolution of the trust network is generated, at every epoch, by the current block leader. A block leader is elected based on the version of the trust graph agreed upon in the previous epoch (i.e., as per the network digest committed in the preceding block).

[7]In Bitcoin, the block size is limited to a maximum of 1MB. Any block bigger than that is considered invalid without further consideration.

The literature contains a plethora of algorithms for computing trust values based on a trust network (e.g., [13] [15], [16], [17]). Any of these could be picked off of the shelf, given that it provides acceptable running times.

B. Dynamics of a trust based consensus

Despite its main downside of high energy expenditure, PoW provides nice security and operability properties under a fully decentralized setting. More importantly, PoW provides a clocking functionality with a guaranteed time window that is large enough for a distributed system to effectively function in epochs without the need of global synchronization. PoW provides a decentralized chronological ordering functionality, as a new block cannot appear until the block it extends have appeared, and as running the PoW ensures that some time (work) has been made between every two consecutive blocks. For this, we do not suggest replacing PoW completely by trust, but rather waiving its difficulty for the highly trusted nodes. That is, the more trusted nodes perform low difficulty PoW, whereas the less trusted nodes have to settle for higher difficulty level. This will disincentivise the less trusted nodes to join PoW altogether, making the pool of active miners restricted to a smaller sub-set represented by the more trusted nodes. Two main pitfalls appear: 1) waiving PoW by trust may represent centralization problem in the hands of the smaller trusted sub-set; 2) low difficulty PoW makes it cheaper to create sybils. We address these pitfalls using two strategies: a trust decaying penalty applied to block leaders, and a damping strategy applied to new trust links in such a way that newcomer nodes cannot abruptly appear in the top trusted sets.

Decaying trust.: Peers are assigned to groups based on their achieved trust measure. We call these groups *trust divisions*. Members of each trust division are entitled to mine at a lower difficulty level proportional to their trust, giving peers of the highest trust divisions higher chances to mine blocks in the system. In order to avoid the pitfall of centralization in the hands of top trust divisions, we need to ensure that they are dynamic through time and that their populations are continuously changing. After running experiments on two real world trust networks we consistently find high fraction of the same nodes popping up in top ranks. Therefore, a naive straightforward PoT protocol that only waives PoW by trust would result in having a small portion of the population taking hold of all the network. More grievous, this will boost the rich gets richer phenomena. Besides, nodes have interest in being selected over and over again and a malicious block leader may manipulate the trust graph by not including links that may penalize her own rank, to increase chances for being selected immediately again. A viable PoT construction needs to contain mechanisms to counter such behavior, which we achieve by applying a *trust decay penalty* for each block leader. This ensures that any peer that has mined a block will fall back to low trust divisions for a number of blocks, thus dramatically decreasing its chances of being elected again in the near future and opening room in high trust divisions for other nodes to get upgraded. Furthermore, this equally limits the impact of

sybils as they cannot maintain high trust even if they manage to achieve it by having trust links from honest trusted nodes. We detail this further under Section III-C5.

Damping new links.: In PoW, the difficulty level is adjusted proportional to the computational capacity available in the system. This disincentives peers from creating sybils as doing so will result in dividing their computational power, thus effectively diminishing their mining capabilities. In PoT, trusted peers are entitled to mine at lower difficulty levels, making it also more feasible to create sybils. However, in order to benefit from lower difficulty PoW peers need first to achieve high trust in the trust network. Unfortunately, trust can be maliciously boosted using sybils and the literature provides many studies on this spam farm phenomena [18]. Solutions are provided mostly basing on trust seeding, where a set of trusted nodes is identified and the trust network evolves around it. This includes white-listing, vouching, invitation based confirmation, etc [19]. We consider that honest nodes in the network will actively participate in the process of detecting sybils and removing them from the network by adopting state-of-the-art sybil defense mechanisms. In order to enable this, sybil nodes should not able to abruptly achieve high trust in the system so that honest nodes have the chance to observe and contain their impact. Therefore, in PoT we introduce a control mechanism on the evolution of the trust measure of newcomer nodes by adopting a *damping* strategy to new trust links, where their impact grows proportionally to the number of consecutive epochs for which they have survived. We detail this further in the following section.

III. PoT construction

In order to achieve all the requirements discussed above and avoid the pitfalls, we build our PoT construction on two pillars: 1) a decaying trust mechanism applied to block leaders in order to prevent dominance of the network by a small trusted clan and to detain the effect of sybils; 2) a controlled evolution of the trust network to further limit sybils by preventing their brisk rise in the trust ladder, thereby facilitating their detection.

An overview of the construction with all its elements is provided on Figure 1. The Blockchain system starts by going first through a trust bootstrapping phase during which consensus is achieved by pure PoW for a few number of blocks. During this bootstrap phase, a trust graph emerges in parallel to the usual creation of transaction blocks. At every new epoch (block) peers in the network (i.e., miners) individually broadcast their outgoing trust links. In addition to transaction payload, every new block also commits a digest of the evolution of the trust graph based on the trust links announced thus far. New trust links in the network are subject to a *damping strategy*, assigning to them weights that represent their age in the system in terms of number of consecutive blocks during which they have appeared. This damping mechanism favors old and consistent trust links, making it slower for newcomer nodes to boost their trust using new links. After the bootstrapping phase, which ends after a fixed number of blocks s defined by the protocol, the PoT phase starts and consensus changes from PoW to PoT. In PoT, every peer has a trust value extracted from the evolved trust graph. The trust graph is encoded in the blockchain itself, making it possible for each peer to have a unified view and to be able to retrieve trust values in an ambiguous way.

Peers are ordered by their trust and are accordingly assigned to a *trust division* based on the range where their trust value falls. Each trust division has an associated PoW difficulty that its members use for their mining. When a peer hits a solution for PoW using the mining difficulty level of the trust division she belongs to, she announces her block which the other peers verify and accept as a valid extension of the consensus blockchain. If more than one valid block is heard of at the same time, the rule is to go for the block from the peer with the highest trust value. Once a peer mines a block and it gets appended to the blockchain, this peer's trust gets subjected to an exponential decay relative to the starting trust value. This ensures that every block leader mines at a higher difficulty level for a number of blocks after the one she has already mined, decreasing by that their chances to mine blocks in the near future.

In what follows, we formalize the elements of the proposed PoT construction as well as the PoT consensus protocol.

A. The trust network

The trust network is composed of the trust links that peers have deliberately announced in the Blockchain system. At every epoch, the trust network corresponds to the version that would have been committed in the last block of the blockchain. We represent the trust network, at every epoch t, as a weighted graph where nodes represent the peers in the Blockchain and the edges the trust links announced with their age being their corresponding weight. We represent this *trust graph*, at epoch t as, $TG_t = (V, E, \mathcal{W})$, where

- V is the set of nodes (or peers), where each node $v_i \in V$ announces itself to the network using a pair of cryptographic signature keys, (SK_i, VK_i). SK_i is the signing key secret to v_i and VK_i is its corresponding verification key, which also represents an identity for V_i in the system.
- E is the set of relationship edges such that $e_{ij} \in E$ denotes a direct trust link from node v_i to node $v_j \in V$ that is announced and signed by v_i.
- $\mathcal{W}$ is a function that assigns for every edge $e \in E$ a weight $w_e(t) \in [0, 1]$.

We consider a trust algorithm that generates trust values for nodes based on a weighted trust graph, such as for example the ones in [17], [16], [15], or any other trust algorithm. We refer to such algorithm by $\mathcal{TA}$. Executing $\mathcal{TA}$ on the trust graph generates for each node a trust value: $\mathcal{TA}(TG_t)$: $\forall v_i \in V, v_i.trust$.

A digest of the trust graph TG_t at epoch t is committed in the block produced during that same epoch. TG_t becomes the trust graph based on which PoT consensus (see sub-section III-C5 below) is achieved during epoch $t+1$. The data of the graph itself could be stored on any medium, either locally at

the level of peers, in a cloud storage, or both. In any case, each peer will make sure to get hold of the right trust graph for each epoch based on the digest committed in the block of one previous epoch.

B. The blockchain

The blockchain, as a data structure, consists of a sequence of blocks ordered in time and sealed to each other with a cryptographic lock. Blocks are generated in epochs, with the rate of one block per epoch. System transactions are broadcasted in the network, and peers have an agreed on mechanism to evaluate transactions validity based on the considered application. In every epoch, every peer prepares her/his block of valid transactions that she heard of, and participates in the leader election mechanism put in place. In PoW, this done by solving a hash puzzle using brute force. The first peer to find a valid solution announces it to the rest of the network, and is also considered the leader elected for the current epoch. The PoW hash puzzle consists at finding a *nounce* such that hashing the prepared block along with the nounce results in a hash value that is smaller than the set difficulty level. To formally define a PoW blockchain, we first define its *valid block*. We adopt similar notations to [20].

Definition 1: Valid PoW Block. Let $H() : \{0,1\}^* \rightarrow \{0,1\}^k$ be a cryptographic hash function mapping input from $\{0,1\}^*$ to $\{0,1\}^k$. A *valid PoW block* is defined as a triple $\mathcal{B} = (h, b, ctr)$, where $h \in \{0,1\}^k$, $b \in \{0,1\}^*$, $ctr \in \mathbb{N}$, and $\mathcal{B}$ satisfies a validity predicate, as follows:

$$valPoWblock^D := (H(ctr, h, b) \leqslant D) = 1.$$

The parameters $D \in \mathbb{N}$ refers to the difficulty level of the block $\mathcal{B}$.

A blockchain, denoted as $\mathcal{C}$ is a *sequence of valid blocks*, such that the rightmost block is called the head of the chain. Let us denote the head as $\mathcal{C}.head$. Any chain $\mathcal{C}$, with head $\mathcal{C}.head$, can be extended to a new longer chain $\mathcal{C}'$ by attaching a new block $\mathcal{B}' = (h', b', ctr')$, such that $h' = H(ctr, h, b)$: $\mathcal{C}' = \mathcal{C}||\mathcal{B}'$. The head of the new chain $\mathcal{C}'$ becomes the new block $\mathcal{B}'$: $\mathcal{C}'.head = \mathcal{B}'$.

By definition of the blockchain, the elements h and ctr, of block $\mathcal{B} = (h, b, ctr)$, are parameters proper to management and security: h represents the hash of the previous block and serves the purpose of locking blocks to each other and ctr represents nounce that proves the performed work. On the other hand, element b is proper to the application and refers to transaction payload.

Now that all required elements are defined, we shall instantiate the PoT protocol.

C. The PoT protocol

Under the PoT protocol, in addition to the transaction payload and the header meta-data as in PoW, every block additionally includes a signature by the peer that commits it to the blockchain, as well as a digest of the trust graph constructed during the current epoch. That is, given block $\mathcal{B}_t = (h, b, ctr)$, that is committed at epoch t by node $v_i \in TG_{t-1}.V$, b is made up of the following elements:

- the transactions payload, p,
- the digest of the trust graph constructed at t, $dig(t) = H(TG_t)$, where $H()$ is a cryptographic hash function,
- a digital signature by node v_i, $\delta = sign_{SK_i}(h, ctr)$.

The protocol operates in epochs. We assume the system is bootstrapped in the very first few epochs, $t = 1, t = 2, ...t = s$ where s is the length of the bootstrapping phase, by pure PoW. During this bootstrapping phase, each new block miner adds, in addition to transaction payload, the evolved version of the trust graph as the collection of trust links announced thus far. We assume that by epoch $t = s$ the trust graph TG_s converges to a trust network based on which peers can already be differentiated from each other based on their achieved trust values. Therefore, starting from epoch $t = s + 1$ the effective PoT phase starts, and peers execute trust algorithm $\mathcal{TA}$ on TG_s, the trust graph digested in the block mined at $t = s$, to retrieve the corresponding trust values based on which PoW will be waived.

1) Trust divisions: We need to generate the different difficulty levels for each peer based on their achieved trust. This could be done by applying a continuous waiver function that generates difficulty values that are inversely proportional to the input trust. However, this will create too much discrepancy between the difficulty levels at which every single peer mines, even when their trust values are not so different. Another alternative is to apply a categorical waiver, in which peers are assigned to groups that correspond to a certain trust range. All members of each group mine at the same difficulty level. We call these groups *trust divisions*. Trust divisions could be established based on defined ranges of trust (e.g., trust higher than value x delimits the first trust division, between values x and y delimits the second trust division, etc.), or on ranges from the probability distribution of trust given by the trust graph (e.g., the fist trust division corresponds to the 90th percentile, the second trust division to the 75th percentile, etc.). Regardless of how trust divisions are established, we consider a number r of trust divisions in the system denoted as $TD_1, TD_2, \ldots, TD_r$, where TD_1 is the first trust division representing the first highest trust nodes, TD_2 is the second trust division representing the second highest trust nodes, etc. Each trust division also corresponds to a difficulty level $diff(TD_x)$, such that, $diff(TD_x) > diff(TD_{x-1}), x \in \mathbb{N}^{[2,r]}$.

At every epoch t during the PoT phase ($t > s$), peers race to solve the next block to extend the blockchain using the difficulty level corresponding to the trust division they fall into, as defined by their own trust value achieved in epoch $t - 1$. Once a peer solves a new block it broadcasts it to the network. The other peers verify the validity of the block, and if valid accept it as the next extension to the blockchain. We note that any peer in the network can, unambiguously and consistently with all the other peers, retrieve the trust values for each peer, identify trust divisions membership, as well as the corresponding mining difficulties of each. This is because the trust graph at $t-1$ is made publicly known by consensus, hence the *independence* and the *verifiability* properties are

maintained.

2) Trust decay and recovery: When node v_i mines a block that is committed to the Blockchain at epoch t, v_i is thereafter subject to an activity slow down, so as to ensure that a small subset of highly trusted node does not dominate the network, as well as to limit sybil attacks. This also plays as a deterrent factor for v_t not to work around the trust graph constructed during epoch t by favoring the trust links that would boost its own trust. Therefore, the trust of v_t is subjected to a decay procedure with quick decay and slow recovery proportional to its initial trust at epoch t. For this, we make use of exponential decay functions. An exponential decay function is expressed as: $N(t) = N_0 \cdot e^{\lambda t}$, where t is time, $N_0 = N(0)$ is the value of the decaying quantity at time 0, and λ is the decay constant. Larger λ will make the quantity fade much more quickly. As we want a slow recovery, we set $\lambda \in]0, 1]$. The fraction $\frac{1}{\lambda}$ represents the mean life time of the decaying quantity; that is, the time at which the quantity is diminished to e^{-1} times its initial value. Accordingly, at time $\frac{k}{\lambda}$ the quantity will be diminished to e^{-k} times its initial value. Time in our system is represented in terms of number of blocks, and so k could be chosen so that the fraction $\frac{k}{\lambda}$ represents the number of blocks during which a node remains subject to the decay procedure.

At every epoch t, the $TrustDecay$ procedure is executed by peers to apply a decay factor to the miners of recent blocks. The procedure takes as input the decay length factor k, the decay constant λ, the current epoch t, and the current blockchain. The procedure rolls back $\frac{k}{\lambda}$ blocks starting from t and decays the trust value of the leader of each of the visited blocks.

procedure TRUSTDECAY(k, λ, t, the blockchain)

Set integer, $\alpha = \frac{k}{\lambda}$
if $\alpha \geq t$ **then** (not enough blocks in history)
$\alpha = 1$
else$\alpha = t - \alpha$
end if
for c `from` t `to` α **do**
$v_c.trust = v_c.trust - v_c.trust \cdot e^{\lambda(t-c)}$ (v_c is the leader of block mined at epoch c)
end for
end procedure

3) Damping new edges: In order to contain the escalation of new nodes up the trust ladder, damping is applied to newly appearing edges. This is achieved by assigning weights to edges proportionally to the number of consecutive epochs for which they survive; i.e., the weight of an edge starts at 0 when it first appears and increases by a step $\tau \in [0, 1]$ each time it is again in the next epoch until it reaches 1. That is, $\forall e \in TG_t.E$ weight is assigned as follows:

$$w_e(t) = \begin{cases} w_e(t-1) + \tau, & \text{if } e \in TG_{t-1}.E \\ 1, & \text{if } \{e \in TG_{t-1}.E \wedge \\ & \quad w_e(t-1) + \tau > 1\} \\ 0, & \text{otherwise.} \end{cases}$$

4) Valid PoT block: Given all the elements presented above, a valid PoT block is defined as follows:

Definition 2: Valid PoT Block. Let $\mathcal{B}_t = (h, (p, dig(t), \delta), ctr)$, where $\delta = sign_{SK_t}(h, ctr, dig(t))$ with SK_t being the signing key of node $v_t \in TG_{t-1}.V$ proposing the block $\mathcal{B}_t$. Let TD_x be a trust division, and $diff(TD_x)$ be the difficulty level applied to TD_x. $\mathcal{B}_t$ is considered a *valid PoT block* at epoch t, iff: $\mathcal{B}_t$ is a valid PoW block with difficulty $diff(TD_x)$ and v_t belongs to trust division TD_x, based on its trust value $v_t.trust$. That is, $\mathcal{B}_t$ satisfies the following validity predicate:

$valPoTblock^d(\mathcal{B}_t) := (valPoWblock^d(\mathcal{B}_t) \wedge v_t \in TD_x \wedge d = diff(TD_x)) = 1.$

The difficulty levels of trust divisions could be set using different heuristics, exactly like the difficulty in Bitcoin PoW is designed to adjust so as to keep the duration of epochs to be roughly 10 minutes. This duration is also known as the *block time*, referring to the average time it takes for the network to produce a new block. From an efficiency perspective, the smaller the block time the better it should be as transactions get confirmed more rapidly. On the other hand, a very small block time could affect the stability and the consistency of the system, mostly because of network latency. Besides, the block time is also related to the security of the blockchain. We postpone this discussion to the next subsection, and we focus here on the fact that the setup of trust divisions and their corresponding difficulty levels could be statically defined in the protocol, as they can be dynamically adjusted in the network, as it is with the difficulty level in Bitcoin for instance.

5) PoT consensus: Given all the elements provided thus far, we summarize the PoT consensus protocol in Table I. At the start of every epoch t, the state of the consensus blockchain $\mathcal{C}$ is assumed to be known to all the peers in the network. This includes knowledge of the trust graph at epoch $t-1$. Trust divisions and their corresponding difficulty levels are also known as part of the protocol. First, nodes compute the trust algorithm $\mathcal{TA}$ on the trust graph at $t-1$ and then apply the $TrustDecay()$ procedure to decay the trust of recent block leaders. All nodes that appear in TG_{t-1} are eligible to participate in mining the block at epoch t. They engage in the process of solving a PoW hash puzzle with the difficulty level corresponding to the trust division under which they fall. When node $v_z \in TG_{t-1}.V$ finds a solution, it broadcasts the proposed block $\mathcal{B}_z$ to the rest of the network. If the block is a valid PoT one, it is accepted by the other peers and is appended to the blockchain $\mathcal{C}$. If not, then the peers ignore the non valid proposed block and continue back to step 3 by which all eligible nodes continue trying to find a valid block to propose. In case the proposed block is valid and accepted as the new head of the blockchain, the block leader v_z is subjected to the trust decaying procedure from the next epoch. Epoch t ends, and the trust graph encoded in $\mathcal{B}_z$ is now the effective trust graph for epoch $t+1$.

Concurrent blocks and branches. More than one valid block could be solved at the same time; that is, peers could simultaneously hear of more than one valid block. In such a case, the rule is to go with the block suggested by the highest trust peer. Besides, due to network latency and the dynamic

TABLE I
POT CONSENSUS PROTOCOL

1.	*Start of epoch* t:	
	These elements are known:	the trust graph TG_{t-1}, the trust divisions with their difficulty levels, as well as the consensus blockchain, $\mathcal{C}$
2.	$\forall v_i \in TG_{t-1}.V$	Compute algorithm $\mathcal{TA}$ on TG_{t-1} and apply $TrustDecay()$ procedure to get effective $v_i.trust$ for every v_i
3.	$\forall v_i \in TG_{t-1}.V$	Solve PoW with difficulty $diff(TD_x)$, such that $v_i.trust$ qualifies it to belong to DT_x
4.	$v_z \in TG_{t-1}.V$	Solves candidate block $\mathcal{B}_z$. v_z is qualified to mine at difficulty d
5.	$v_z \rightarrow$ all the network	Broadcast $\mathcal{B}_z$
6.	*All peers in the network*	Verify $\mathcal{B}_z$ as per Definition 2
		If $validPoTblock^d(\mathcal{B}_z)$ do step 7; otherwise, go back to step 3
7.	*All the network*	Extend blockchain: $\mathcal{C}.head = \mathcal{B}_z$
8.	End of epoch t	

nature of the connectivity network, some peers may hear of a valid block $\mathcal{B}$ only, while others may hear about $\mathcal{A}$ only. This is also exhibited in existing PoW systems, as blockchains tend to develop in multiple branches. As all peers will eventually end up learning about each other branches, the rule to solve this issue is to always go for the longest branch heard of so far. In our PoT construction, the priority goes to the most trusted branch; that is, the longest branch rule is replaced by the most trusted branch, where the trust of a branch is the sum of the trust values of the miners of its blocks as in the epoch when each one of them mined his/her block.

D. Security analysis

A blockchain is said to be secure if it ensures the *persistence* and the *liveliness* of its content [20]. Persistence dictates that if an honest peer states that a block is part of the blockchain, any other honestly responding peer will also report the same; whereas liveliness refers to the quality that if a valid block is appended to the consensus blockchain it will become persistent. Persistence and liveliness are both related to how old (also interpreted as how deep) a block is in the blockchain. The older (deeper) a block is in the blockchain the higher is the certainty by which its persistence and liveliness hold, as making any change to that block requires rebuilding all the blocks that follow it. In PoW rebuilding blocks requires computational power that is fast enough to compete with the rest of the network that is continuously extending the blockchain. In PoT, rebuilding blocks requires either a high trust measure to qualify for low difficulty PoW, or ownership of a computational power that can cover the gap between low difficulty mining allowed for trusted nodes and high difficulty mining available for non trusted nodes. We assume that this gap between the two difficulty levels is large enough so as no existing computational power can cover it. Therefore, rebuilding blocks requires achieving a proof of trust which cannot be acquired without announcing and signing incoming trust links from some of the peers in the rest of the network. More importantly, a malicious peer needs not only to achieve a high trust measure to qualify for higher trust divisions, but also needs to maintain this trust over time. This is prevented through the trust decay mechanism; thus, for an attack to succeed, the attacker needs to consistently own/collude with the nodes in the highest trust divisions; i.e., needs to own at least 50% of nodes in every trust division. Therefore, while a 50% attack in PoW can happen with the ownership of more than 50% of the total computational power in the system, in PoT this needs consistent ownership of 50% of nodes in at least all top trust division that have chances to be elected. Such sybil penetration is hard to conceive under honest-majority trust networks [12] [14].

TABLE II
SUMMARY OF THE THREE DATASETS USED IN OUR EXPERIMENTS

Dataset	Number of nodes	Number of edges
Dataset1	5,881	31,677
Dataset2	1,899	59,835
Dataset3	1,632,803	30,622,564

IV. EXPERIMENTAL ANALYSIS

We run experimental analysis with two main objectives: 1) study the effect of the decaying procedure on the distribution of block miners across trust divisions; 2) evaluate the impact of controlling the evolution of newcomer node trust w.r.t the emergence of sybil behavior.

A. Datasets and setup

We perform our expriments on three real world datasets encoding who trusts whom in a network. The three datasets are retrieved from [21], and Table II represents their summary. *Dataset1* refers to the bitcoinOTC dataset in [21] respectively. The original bitcoinOTC dataset contains 35592 edges representing trust values ranging from -10 to 10 per edge. In *Dataset 1*, we prune only those edges with a positive tust value. We do no consider these values in weithing edges as we assign our own based on different experimental setups as is shown below. *Dataset2* refers to the CollegeMsg dataset in [21], representing an email exchange graph. *Dataset3* is the largest with over 1.5 million nodes and over 30 million edges. It represents the Pokec dataset in [21] providing the friendship network of the most popular social network in Slovakia (Pokec). Friendships in Pokec are directed, so we consider them as trust links. *Dataset1* and *Dataset2* are temporal, with edges timestamped based on their order of appearance in the network; thus it is possible to slice each of the two datasets into evolutionary snapshots. For *Dataset3* we used the results from [22] to reconstruct possible evolution timelines of a given network based on degree assortativity. We get 50 evolution snapshots for *Dataset3*, and 10 for each of *Dataset1* and *Dataset2*. We

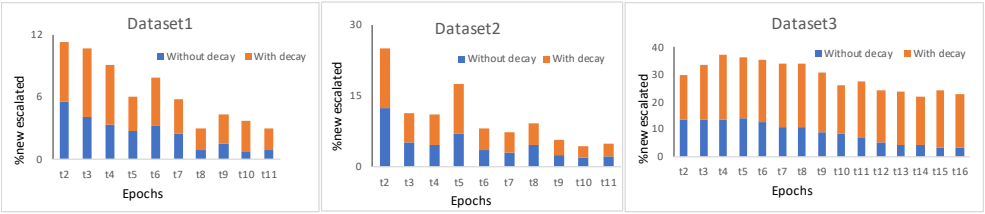

Fig. 2. The effect of $TrustDecay()$ procedure on the dynamics of highest trust divisions in terms of percentage of new nodes appearing between epochs with and without applying decay on the three datasets

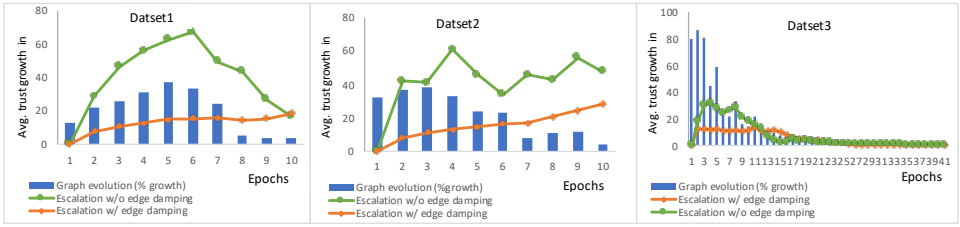

Fig. 3. The effect of applying edge damping on the escalation of newcomer nodes across trust divisions (growth of nodes trust across time). The x-axis represents time in epochs. The y-axis represents the average % of growth of nodes trust as the graph evolves relative to the first epoch.

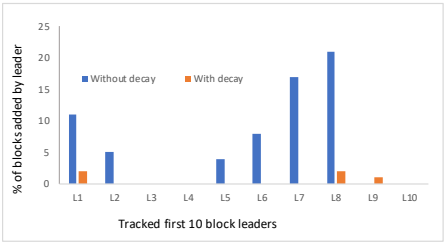

Fig. 4. Number of blocks added over a 100 epoch observation window by same set of 10 tracked block leaders.

consider each of the snapshots as the version of the trust graph in a block.

We simulate PoT operation by using Pagerank algorithm [17] to generate trust measures for each peer. For the $TrustDecay()$ procedure we set the decay constant λ to 0.5 and k to 5, making the decay procedure applied 10 blocks back at each epoch. For damping new edges, we use the increment step $\tau = 0.1$, so that an edge needs to appear for 10 consecutive epochs before its weight saturates at 1. For trust divisions, we set them based on percentiles with a 5% step: TD_1 corresponds to nodes whose trust value in the 95th percentile, TD_2 to nodes whose trust value in the 90th percentile but not in the 95th, etc. In order to simulate the results of low difficulty PoW, we use a Bernoulli distribution process to first select a trust division with probability equal to the average trust of the peers it contains. Then we pick up uniformly at random one of the peers from the highest trust division that passed the Bernoulli trial. As the lowest trust divisions have very minimal chances to be elected, we focus all our experiments on studying the dynamics of the five first trust divisions; i.e., up to the 75th percentile.

B. Dynamics of trust divisions

We start by studying the dynamics of trust divisions in terms of the percentage of new nodes appearing in a trust division compared to the previous epoch, with and without using trust decay. The highest trust divisions are characterized by slow evolution, with only a few new nodes added to their sets at each epoch. We run the experiment with the setup explained above on the three datasets twice, first by applying the $TrustDecay()$ and second by not considering any decay. We observe, at each epoch, the percentage of new nodes appearing in trust divisions TD_1 and TD_2, which we refer to as the trusted set (i.e., nodes in the 90th percentile). Figure 2 provides the observed results, contrasting the percentage of new nodes appearing at each epoch when decay is applied to when it is not applied. As we can see on the figure, applying decay introduces up to 80% more changes in the set of highly trusted nodes compared to no decay, as well as ensures a constant change in the trusted set. This is especially important as the trust graph starts stabilizing with fewer edges being added over time. For example, for *Dataset3* the graph starts to stabilize starting from epoch t10 and changes to the trusted set at that epoch is less than 7% when decay is not applied. However, applying decay increases the changes in the trusted set to above 19%.

To further study the effect of decay on preventing nodes from dominating the system, we simulate the development of the blockchain on *Dataset1* and we focus on the set of nodes selected as leaders for the first 10 blocks (L1, L2, .., L10). By the 11th epoch, we stabilize the graph at its final evolution snapshot and we keep simulating the development of the blockchain for 100 blocks more, selecting block leaders by the simulated low difficulty PoW over top trusted divisions as explained under subsection IV-A above. We run the same process twice, with and without decay. Under each case, we track the leaders of the first 10 blocks and observe when they appear as leaders again in the upcoming 100 blocks. We provide the results in Figure 4, where we can see that 66 out of the 100 blocks in the observation window have been added by one of the first 10 leaders. However, when decay was applied, only 3 of the 10 first block leaders were again selected during the following 100 blocks with a total of only 5% of the blocks added by the same set, making it robust to risks of monopolization.

C. Newcomer nodes escalation dynamics

We study the time it takes for newcomer nodes to escalate across trust divisions on the three datasets under two different scenarios: trust graph evolution 1) without applying damping to new edges, and 2) with damping where the weight of an edge starts at 0 when it first appears and is incremented by 0.1 whenever the edge appears again in the subsequent epoch. As the trust graph grows (new nodes and edges are added), we observe the percentage of nodes that have been escalated to an upper trust division compared to one previous epoch. Figure 3 shows the results for each of the three datasets, where the x-axis represents time in terms of epochs, and the y-axis refers to the average percentage of growth in node trust as the trust graph evolves. The results show that the percentage of escalated nodes when edges are not damped is hugely affected by the changes in the trust graph. On the other hand, the escalation is smoothed across epochs when the damping is applied; i.e., changes due to the evolution of the graph do not dramatically affect the escalation of nodes under the edge-damping scenario compared to what happens under the no-damping one. Besides, the growth of node trust under the edge-damping scenario is considerably reduced. For example, on the three datasets, the rate of node escalation was contained by up to 68% in average. This ensures that newcomer nodes cannot swiftly appear in the highest trust divisions making it possible for honest nodes to detect sybil behavior and respond to it before a sybil newcomer gains substantial chances to be elected for a block.

To get better insight on the eventual speed of infiltration of sybils and their potential impact on the system, we make further experiment based on *Dataset3*, by assuming 20% of the nodes to be sybil. Sybils are known to organize in patterns that are different than honest ones, however some sybils may succeed at infiltrating within areas of honest nodes, thus looking as honest themselves [11] [19]. Therefore, the best that sybils could achieve is to imitate honest behavior and to get incorporated within the clusters of honest nodes (have other honest nodes trust them). For this, at each epoch, we consider that 20% of newcomer nodes selected uniformly at random from the set of honest nodes connected to the trusted set are sybil and we label them as such. We label the other 80% as non-sybil. We track the escalation of the nodes labeled as sybil across the subsequent epochs and we observe their percentage under each of the trust divisions at each epoch. Figure 5 shows the results. As we can see on the figure, it is only after more than 30 epochs that no more than 2% of the sybil nodes escalate to the first trust division TD1. 10% of the sybil nodes made it to TD3 by the 26th epoch, whereas less than 0.5% made it to the first trust divisions by that time. We also note that this could be further contained by the edge damping factor. In our experiments we considered an increment of 0.1, making edges saturate after 10 blocks only. Lower increment to edge weight will provide higher containment.

The observed results demonstrate that the progress of newcomer nodes in PoT is slow enough to enable the detection and containment of sybil nodes by the community using any of the state-of-the-art sybil detection mechanisms [11]. Moreover, the decay strategy provides consistent changes in the trusted set even when the trust graph stabilizes with minimal evolution. Besides, the size of the first trust division is observed to be 7.8% in average, but also never more than 9.8% of the size of the trust graph at every epoch and for all datasets. This suggests that the pool of nodes doing PoW is limited to less than 10% of the population, resulting in corresponding energy savings compared to when all the population engage in PoW. Our results show that PoT Blockchains are viable, and are dramatically more energy efficient.

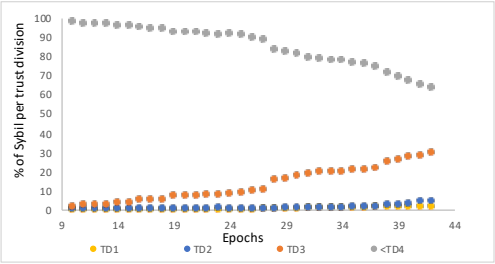

Fig. 5. Percentage of sybil nodes per trust division across epochs. TD1 represents the trusted set.

V. Conclusion and Future work

Blockchain is a promising technology for the development of more P2P applications under various domains; however, the hopes on what the technology can deliver seem to be overestimated. Most Blockchain platforms in operation so far have a financial nature as they rely on an underlying crypto-currency, where either PoW or PoS is used for consensus. PoW is a beast that survives on increasingly burning huge amounts of energy, while PoS relies on crypto-currencies and on the dynamics of wealth distribution with risks of over control by the richest peers. For non-financial Blockchains the only available solutions are restrained to the permissioned model, shifting the system back to centralization. In this paper, we leverage on natural trust to enable non-financial Blockchains without trading off the permissionless characteristic. Our PoT construction achieves consensus based on a trust network that emerges and that is managed in a completely decentralized fashion. We design our solution to be resilient both to the risk of monopolization by a small trusted cartel and to the impact of sybil nodes. Experimental evaluation on real world trust networks shows that PoT can make up to 10-fold savings on energy consumed by PoW without trading off the decentralization guarantees while being even more resilient to monopolization. Altogether, *PoT allows Blockchains to live up to community expectations*.

There are many possibilities to extend the model. For instance, PoT could be based on community membership and multiple blockchains managed collaboratively by different communities from the trust graph. As future work, we plan to work on these extensions as well as to analyze different trust ranking mechanisms other than pagerank.

REFERENCES

[1] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore, "Game-theoretic analysis of ddos attacks against bitcoin mining pools," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 72–86.

[2] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. van Renesse, "Rem: Resource-efficient mining for blockchains." *IACR Cryptology ePrint Archive*, vol. 2017, p. 179, 2017.

[3] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *Proceedings of the 1st Workshop on System Software for Trusted Execution*. ACM, 2016, p. 2.

[4] A. Biryukov and D. Khovratovich, "Tradeoff cryptanalysis of memory-hard functions," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2014, pp. 633–657.

[5] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 51–58.

[6] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, 2018.

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

[8] J. Golbeck, *Computing with social trust*. Springer Science & Business Media, 2008.

[9] J.-H. Cho, A. Swami, and R. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 562–583, 2011.

[10] L. Bahri and S. Girdzijauskas, "When trust saves enegry-a reference franework for proof-of-trust (pot) blockchains," in *The Web Conference 2018*. ACM Digital Library, 2018, pp. 1165–1169.

[11] A. Vasudeva and M. Sood, "Survey on sybil attack defense mechanisms in wireless ad hoc networks," *Journal of Network and Computer Applications*, 2018.

[12] J.-H. Cho, A. Swami, and R. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 562–583, 2011.

[13] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[14] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 47, 2013.

[15] R. Falcone, G. Pezzulo, and C. Castelfranchi, "A fuzzy approach to a belief-based trust computation," in *Workshop on Deception, Fraud and Trust in Agent Societies*. Springer, 2002, pp. 73–86.

[16] F. Hendrikx, K. Bubendorfer, and R. Chard, "Reputation systems: A survey and taxonomy," *Journal of Parallel and Distributed Computing*, vol. 75, pp. 184–197, 2015.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[18] Y. Du, Y. Shi, and X. Zhao, "Using spam farm to boost pagerank," in *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*. ACM, 2007, pp. 29–36.

[19] K. Zhang, X. Liang, R. Lu, and X. Shen, "Sybil attacks and their defenses in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 372–383, 2014.

[20] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications." in *EUROCRYPT (2)*, 2015, pp. 281–310.

[21] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[22] K.-K. Kleineberg and M. Boguná, "Evolution of the digital society reveals balance between viral and mass media influence," *Physical Review X*, vol. 4, no. 3, p. 031046, 2014.