Yan, Zheng; Peng, Li; Feng, Wei; Yang, Laurence T.

# Social-Chain: Decentralized Trust Evaluation Based on Blockchain in Pervasive Social Networking

*Published in:*
ACM Transactions on Internet Technology

*DOI:*
10.1145/3419102

Published: 01/02/2021

*Document Version*
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

*Please cite the original version:*
Yan, Z., Peng, L., Feng, W., & Yang, L. T. (2021). Social-Chain: Decentralized Trust Evaluation Based on Blockchain in Pervasive Social Networking. *ACM Transactions on Internet Technology*, *21*(1), Article 17. https://doi.org/10.1145/3419102

# Social-Chain: Decentralized Trust Evaluation Based on Blockchain in Pervasive Social Networking

ZHENG YAN, Xidian University & Aalto University
LI PENG and WEI FENG, Xidian University
LAURENCE T. YANG, St. Francis Xavier University

Pervasive Social Networking (PSN) supports online and instant social activities with the support of heterogeneous networks. Since reciprocal activities among both familiar/unfamiliar strangers and acquaintances are quite common in PSN, it is essential to offer trust information to PSN users. Past work normally evaluates trust based on a centralized party, which is not feasible due to the dynamic changes of PSN topology and its specific characteristics. The literature still lacks a decentralized trust evaluation scheme in PSN. In this article, we propose a novel blockchain-based decentralized system for trust evaluation in PSN, called Social-Chain. Considering mobile devices normally lack computing resources to process cryptographic puzzle calculation, we design a lightweight consensus mechanism based on Proof-of-Trust (PoT), which remarkably improves system effectivity compared with other blockchain systems. Serious security analysis and experimental results further illustrate the security and efficiency of Social-Chain for being feasibly applied into PSN.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: Trust evaluation, blockchain, consensus mechanism, pervasive social networking

**ACM Reference format:**
Zheng Yan, Li Peng, Wei Feng, and Laurence T. Yang. 2020. Social-Chain: Decentralized Trust Evaluation Based on Blockchain in Pervasive Social Networking. *ACM Trans. Internet Technol.* 21, 1, Article 17 (January 2021), 28 pages.
https://doi.org/10.1145/3419102

The work is supported in part by the National Natural Science Foundation of China under grants 61672410 and 61802293, the Academy of Finland under grants 308087, 314203, and 335262, the Key Lab of Information Network Security, Ministry of Public Security under grant C18614, the open grant of the Tactical Data Link Lab of the 20th Research Institute of China Electronics Technology Group Corporation, P.R. China under grant CLDL-20182119, the Shaanxi Innovation Team project under grant 2018TD-007, the National Postdoctoral Program for Innovative Talents under grant BX20180238, the Project funded by China Postdoctoral Science Foundation under grant 2018M633461, and the 111 project under grant B16037.
Authors' addresses: Z. Yan, the State Key Lab on Integrated Services Networks, School of Cyber Engineering, Xidian University, No. 2 South Taibai Road, Xi'an, China, 710071, the Department of Communications and Networking, & Aalto University, Konemiehentie 2, P.O. Box 15400, Espoo, Finland, 02150; email: zyan@xidian.edu.cn; L. Peng and W. Feng, the State Key Lab on Integrated Services Networks, School of Cyber Engineering, Xidian University, No. 2 South Taibai Road, Xi'an, China, 710071; emails: pengli_email@163.com, weifeng.ft@foxmail.com; L. T. Yang, the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada, B2G 2W5; email: ltyang@gmail.com.

1533-5399/2020/01-ART17 $15.00
https://doi.org/10.1145/3419102

## 1 INTRODUCTION

Pervasive Social Networking (PSN) provides instant social networking services at anytime and anywhere with the support of heterogeneous networks. It offers instant social services to not only acquaintances, but also familiar strangers and unfamiliar ones, especially in time-critical or mission-critical scenarios. For example, a user can employ PSN to seek urgent aids, instant data sharing, and recommendations from the people in vicinity. Compared with traditional social networking, PSN is advanced in terms of flexibility, accessibility, context-awareness and ubiquity, since PSN users can switch among different types of network carriers to gain high-quality social network services according to network conditions, available resources, and the contexts of social communications [48, 49]. PSN is becoming an ideal platform for various services, such as advertisement, sharing economy, disaster rescue, and so on.

PSN aims to provide social services in a trustless environment, where not any party can be fully trusted. Obviously, it confronts severe security and trust problems. PSN users may communicate with dishonest or malicious strangers. In this context, trust becomes crucially important, since it assists social decisions [48]. However, it is difficult to evaluate trust accurately and build trust relationships in PSN, especially among people located in different places without any face-to-face interactions or pre-established trust relationships [44]. In PSN, it lacks a centralized party to perform information collection, social data aggregation, and trust evaluation [50], which should be self-organized by involved parties in practice. Due to this reason, trust evaluation in PSN should be conducted in a decentralized way without depending on any trusted parties. Thus, many existing trust evaluation schemes [35, 42] relying on a centralized party become infeasible to be applied into PSN.

Blockchain is a promising technology of decentralization [31]. Its security relies on a consensus mechanism rather than the trust of a centralized party. Currently, blockchain has attracted intensive attention due to its advances in terms of transparency, tamper-resilience, and immutability [38, 39]. Because of the decentralized and automatic properties of blockchain, we consider employing the blockchain to build up trust in PSN, and the first step is to evaluate trust in a decentralized manner.

However, it is challenging to utilize the blockchain for trust evaluation. First, existing blockchain systems based on Proof-of-Work (PoW) [31] usually suffer from high resource consumption, low efficiency, poor scalability, and forking. Second, many improved solutions are still unsatisfactory. People have developed many consensus mechanisms to improve the performance of a blockchain system, such as Proof-of-Stake (PoS) [24], Delegated Proof of Stake (DPoS) [26], Practical Byzantine Fault Tolerance (PBFT) [8], and so on. Although these schemes reduce computational consumption, they either suffer from low efficiency, fail to achieve high scalability, or still confront forking. Third, many blockchain systems cannot really ensure decentralization [17]. They are not robust enough, facing collusion attacks or other attacks raised by a powerful malicious system party. Fourth and the most important, existing consensus mechanisms of blockchain do not embrace trust evaluation, thus cannot support trust consensus in the context of PSN. To conclude, existing blockchain systems are not feasible to be directly applied into PSN for the purpose of trust evaluation with expected performance.

In this article, we propose Social-Chain to leverage blockchain for trust evaluation in PSN. It is a blockchain-based decentralized trust evaluation system. We design a novel consensus mechanism called Proof-of-Trust (PoT). It is composed of four functional algorithms, i.e., Block Generation, Timestamp Validation, Mining Winner Selection, and Consensus Policy Setting. Among them, Block Generation determines when a miner can create a new block. Different from other consensus mechanisms, PoT enables miners to generate a new block by accumulating a certain amount of new trust evidence. This design avoids resource-consuming computation and helps accelerate block generation. Before a newly created block being broadcasted to all miners, the creating miner needs to select a random group of miners to verify its timestamp. The goal of Timestamp Validation is to prevent a dishonest miner from attaching a fake timestamp to the block. Mining Winner Selection uniquely selects a block from multiple candidates so a blockchain fork can be avoided. Specifically, we limit the total number of wins of an individual miner in a specific time period to ensure decentralization. Finally, we employ Consensus Policy Setting to help the miners determine whether a block can be finally accepted. In PoT, a newly generated block can be confirmed as the next block if and only if it is approved by a sufficient number of miners with a sufficient sum of trust values, as agreed by Consensus Policy Setting. The design of Social-Chain holds such a principle that the sufficient number of reputable miners decide the correctness of its blockchain. Thus, public verification on trust evaluation becomes open and transparent to every PSN node. In this way, we achieve consensus on a newly generated block and trust evaluation at the same time.

We also design a practical trust evaluation algorithm to evaluate the social trust of PSN nodes during block generation. This algorithm fits into the decentralized scenario of PSN and can mitigate some potential attacks. We use the blockchain to store trust evidence and perform trust evaluation based on the data recorded in blockchain. Thus, we can ensure trustworthy execution of trust evaluation.

Different from other blockchain consensus mechanisms, the novelties of PoT lie in the following facts: First, it integrates trust evaluation into blockchain consensus, thus simultaneously achieves trust consensus during block generation. Second, it avoids heavy computation consumption, e.g., caused by cryptographic puzzle calculation, thus very efficient. Third, by employing Mining Winner Selection, it can uniquely decide a block winner to effectively get rid of forking and ensure decentralization. Together with Timestamp Validation, PoT can offer high security. Finally, it dynamically adjusts consensus conditions based on the statistics of node trust, thus provides essential trustworthiness on Social-Chain management. Through trust evaluation, trust consensus, and associating trust with the PSN nodes, honest behaviors will be highly encouraged and malicious nodes can be evaded in the competition of block generation.

Social-Chain can be easily extended to support various valuable features. For example, to support context-aware trust evaluation, we involve context ID (e.g., expressed by social application ID plus social purpose ID) into trust evidence and local trust reports, thus it is possible for the miner to calculate trust values by linking them to concrete contexts. Besides, Social-Chain can also support authentication on trust by enabling PSN users to check the trust values recorded in the blockchain. In addition, Social-Chain is also feasible to support trust-based PSN data access control and trust-driven action execution based on its open trust ledger. Specifically, the contribution of this article can be summarized as follows:

- We design Social-Chain, a novel blockchain-based decentralized social trust evaluation system in PSN.
- We design a novel consensus mechanism named Proof-of-Trust for block generation and confirmation, which achieves trust consensus with high efficiency and security.
- We propose a trust evaluation algorithm for Social-Chain based on social behaviors, which can resist traditional attacks.
- We theoretically prove the security of Social-Chain consensus mechanism with regard to effectivity, safety, liveness, decentralization, and fault tolerance.
- We implement Social-Chain in both Windows and Android Platforms, based on which we conduct serious experiments to evaluate its performance. The experimental results demonstrate the effectiveness and efficiency of Social-Chain.

The rest of the article is organized as follows: We briefly review related work in Section 2. Section 3 states the system model, security model, and research assumptions of Social-Chain. In Section 4, we describe the detailed design of Social-Chain, followed by security analysis and experimental performance testing results in Section 5. We conclude the article in the last section.

## 2 RELATED WORK

This section reviews the state-of-the-art work of both trust evaluation and blockchain consensus mechanisms.

### 2.1 Trust Evaluation

We first review trust evaluation methods in PSN. Then, we summarize exiting blockchain-based trust management methods and analyze their weakness regarding application in PSN.

*2.1.1 Trust Evaluation in PSN.* Social networking based on Mobile Ad hoc Networks (MANET) has been studied in the literature to make it pervasive, e.g., AdSocial [36]. However, trust and reputation aspects in PSN were not seriously considered in early studies. Later on, the concept of data-centric trust in volatile environments, such as MANET, was introduced to evaluate node trust based on the data reported by nodes [34]. PerChatRep [42] and PerContRep [43] are two PSN applications for pervasive social chatting and pervasive content recommendation, respectively. They adopt a hybrid reputation system architecture, where reputation is evaluated in a distributed way. These two systems evaluate both local and general trust by aggregating local and global social networking experiences, but need the support of a centralized trusted party.

In recent years, there appear a number of studies on PSN trust evaluation. Zhang et al. [54] proposed a factor-enrichment-based hybrid trust framework for trust measurement in e-commerce on-line social networks. Jiang et al. [22] proposed a flow-based trust evaluation scheme named GFTrust. It addresses path dependence with network flow and models trust decay with evidence leakage associated with each node, thus converts a trust evaluation task with path dependence and trust decay into a generalized network flow problem. Shen et al. [37] developed a hierarchical evaluation system to support secure and trustworthy PSN by involving multiple and variable nodes. But the above works did not discuss how to implement trust evaluation in a purely decentralized manner.

Guo et al. [19] proposed a trust-based privacy-preserving friend recommendation scheme in online social networks. In this scheme, various attributes are used to find matched friends and establish social relationships with strangers via a multi-hop trust chain. But how to apply this scheme into PSN requires additional investigation? MobiFuzzyTrust [21] was proposed to semantically infer trust from one mobile user to another that may not be directly connected in a trust graph of a mobile social network by considering social context and by applying fuzzy linguistic technique. The above studies mainly focus on how to evaluate trust, but do not consider transparency and trustworthiness of evaluation. Liu et al. [27] introduced a new concept, Quality of Trust (QoT), which considers such attributes as trust, social relationships, and recommendation roles. They modeled an optimal social trust path selection problem with multiple end-to-end QoT constraints as a Multi-constrained Optimal Path (MCOP) selection problem and proposed an algorithm for Optimal Social Trust Path selection. But this study did not discuss how to support decentralization and public auditing in the context of PSN trust evaluation.

*2.1.2 Trust Management Based on Blockchain.* Recently, there are several researches focusing on using blockchain to achieve decentralized trust management. BLESS [41] aims to build a blockchain-based social credit framework for constructing trustworthy communities, but it lacks concrete design and analysis without describing an efficient consensus algorithm that can be practically used. Yang et al. [51, 52] proposed a blockchain-based decentralized trust management system in vehicular networks, in which vehicles can query trust values of neighbors and then assess the credibility of received messages. A joint Proof-of-Work and Proof-of-Stake miner election method was designed to lower the computational cost of consensus mechanism, but detailed security and feasibility analysis were not given. Mao et al. [29] introduced a blockchain-based credit system to enhance the effectiveness of supervision in a food supply chain. This system adopts a deep learning model to output binary classification based on given credit evaluation texts. But this work does not analyze the performance of the blockchain system. Notably, all of above works did not comprehensively consider how to provide a trustworthy, lightweight, and decentralized way for trust evaluation. No existing literature feasibly integrates social trust evaluation into a blockchain consensus mechanism, which is the main contribution of Social-Chain.

We design Social-Chain, a blockchain-based decentralized trust evaluation system for PSN. Different from existing works, we use the blockchain to store the trust evidence and perform trust evaluation based on the data recorded in blockchain. It ensures that the algorithm of trust evaluation is invoked in an automatic way, which supports trustworthy trust evaluation in the context of a trustless environment.

## 2.2 Consensus Mechanisms

*2.2.1 Consensus Mechanisms for Permissionless Blockchain.* Consensus mechanisms is the core of blockchain. The underlying consensus mechanism of Bitcoin is PoW, which achieves scalability and some level of security. Nevertheless, it suffers from high resource consumption, low efficiency, and low throughput [12]; it is prone to forks, selfish mining, double-spending, and other attacks [13]. Therefore, various PoW improvements or alternatives have been developed for permissionless (or public) blockchains, such as PoS [6, 24] and Proof-of-Creditability (PoC) [16].

PoS greatly reduces computational overhead, which has received extensive attention from both academia and industry. Its security is based on game theory. However, the cost of node misbehaving and forging a blockchain in PoS is low. When a new node joins a blockchain system, it needs to select a trustworthy node as a restoring point to synchronize data. Compared with PoW, PoS is less secure, easier to fork, and more prone to centralization. Moreover, PoS relies on cryptocurrency as incentive for mining and cannot work well in the blockchains without any cryptocurrency. In response to these problems, Buterin et al. proposed the Casper PoS consensus protocol [7], where a consensus node needs to pay a certain amount of deposit. When the consensus node behaves maliciously, it will be punished, thereby increasing the cost of behaving evilly and thus enhancing security. However, this protocol also relies on cryptocurrency as incentive. To prevent from the centralization caused by the introduction of the restoring points, Badertscher et al. [4] proposed to divide a blockchain into several time periods, each time segment is further divided into several time slots, and each time slot contains one block creation. In each time slot, a block generator is selected by a pseudo-random number generator [4]. Therefore, the generator of each block in a time period can be verified by the random number included in the corresponding block, thus the centralization problem is avoided. In response to blockchain throughput optimization, some scholars proposed to divide the nodes into several regions, and each region contains several consensus nodes and processes different transactions [28, 33, 53]. However, dividing consensus miners into several groups (called shard) may incur many problems, such as transaction verification difficulty, data loss, and security sacrifice.

*2.2.2 Consensus Mechanisms for Permissioned Blockchain.* Some schemes construct blockchains using PBFT [8]. However, the PBFT algorithm generates a large amount of communication overhead, thus having poor scalability. It is not suitable to be used for constructing a public blockchain. Miller et al. combined threshold encryption and PBFT and proposed a high-throughput consensus mechanism named HoneyBadgerBFT. However, it still cannot support a large number of consensus nodes [30]. The consensus mechanisms are usually designed for such scenarios where the number of consensus nodes is limited and the nodes are relatively trusted. However, in a open PSN system, there exists a large number of nodes, many of which cannot be trusted. Therefore, HoneyBadgerBFT cannot work well in PSN due to performance degradation.

*2.2.3 Other Consensus Mechanisms.* Some schemes combine the consensus mechanism of public blockchain with PBFT to solve the forking problem while ensuring high efficiency. Among them, a representative algorithm includes DPoS [26]. However, DPoS suffers from several security risks, such as collusion attack raised by proxy nodes, Denial of Service (DoS) attack, dishonest behaviors, and possibility of centralization. Besides, a powerful entity can corrupt a large proportion of states and still has high probability to control the whole blockchain, which incurs centralization. In response to these problems, some schemes use Verifiable Random Function (VRF) to determine a proxy node. Among them, Algorand consensus mechanism selects a set of independent consensus nodes for each block with the VRF to determine a new block by applying a BA* algorithm [18], thereby solving the forking and greatly improving throughput. For an attacker, it is difficult to locate the consensus nodes and launch security attacks, so Algorand can effectively resist DoS attacks. However, Algorand still faces some security threats. It requires honest consensus nodes to hold more than two-thirds of currencies. When the number of offline nodes is large or weak synchronization occurs, blockchain cannot ensure security. However, its consensus algorithm cannot completely rule out the possibility of centralization. Another consensus mechanism based on the VRF is Dfinity [20]. Difinity uses the VRF to divide consensus nodes into several groups. After that, the blockchain system determines the next block generation group with a random number each time. The group selects one proposer with internal random polling, which is verified by the remaining nodes in the group. Difinity guarantees high throughput and low communication latency. However, intra-group consensus still generates high communication traffic. Since the proposer node is determined in advance, it is vulnerable to single-point attacks.

Existing consensus mechanisms are not suitable for being applied into PSN. For one thing, most of them rely on on-chain cryptocurrency as mining incentive. For another, they cannot well solve the problems of centralization and forking at the same time. As for permissioned consensus mechanisms, they cannot work in an open and public PSN environment. In summary, although blockchain consensus has been improved in terms of throughput and verification latency in recent works, whether they can ensure decentralization and guarantee security remains for further investigation. How to design a lightweight, efficient, and secure consensus mechanism in PSN and support its trust evaluation is still an open research issue.

## 3 PROBLEM STATEMENT

In this section, we introduce the system model and security model of Social-Chain, as well as our research assumptions. For easy presentation, we also describe the notations used in this article.

### 3.1 System Model

The structure of our system is shown in Figure 1. The system contains a number of nodes that make use of heterogeneous networks for PSN. A node can be either a user node or a miner (i.e., mining node or consensus node) or play both roles. The user nodes participate in PSN social activities, while the miners are responsible for maintaining Social-Chain, such as collecting trust evidence, performing trust evaluation, and reaching consensus on block generation. Each node contains a number of functional modules. Concretely, PSN apps are used to perform different kinds of social

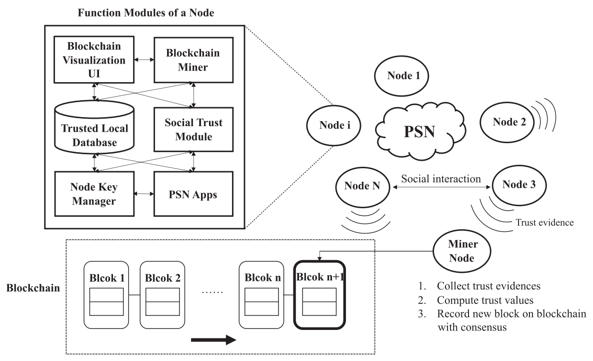


Fig. 1. System model.

networking for different purposes. Blockchain Visualization UI is applied to display the contents of blockchain. Blockchain Miner is responsible for performing the tasks that should be done by a mining node. Node Key Manager is responsible for generating a personal key pair, hashing data, checking data integrity, and signing/verifying signatures. Social Trust Module is applied to record social networking data, handle communications between the underlying node and other miners, and report local trust and trust evidence to the miners. All data related to the above functional modules, e.g., blockchain data, local records about social networking, public-private keys, are stored in Trusted Local Database. The social interactions between nodes will generate trust evidence. The miner nodes in the system always listen to the network and collect incoming trust evidence. Once the volume of collected evidence exceeds a pre-defined threshold, the miner node will run a common block generation algorithm and try to attach a new block to the blockchain through consensus. A valid new block issues a reward to the generator of a previous block and contains node trust information.

## 3.2 Security Model

As analyzed, a centralized architecture is not suitable for PSN. For one thing, a centralized party is not always available in PSN. For another, in practice, PSN nodes are usually rational and profit-driven, thus may not behave honestly and cannot be fully trusted. Additionally, any nodes could be attacked or intruded. Therefore, based on the system model, we present the security model of PSN as follows: PSN nodes do not trust with each other, they behave rationally and make decision according to the facts recorded in the blockchain. The most reputable nodes make decision together to achieve expected trust. There exist malicious nodes in PSN. They try to interfere with normal operations of PSN, which negatively impacts message transmission and trust evaluation. In this article, two types of malicious behaviors are specially considered, i.e., Sybil attack and bad-mouthing attack.

(a) *Sybil attack:* In a public blockchain, with no logically central and trusted authority to vouch for a one-to-one correspondence between a node and its identity, it is possible for a node to present by holding more than one identity, which is defined as Sybil attack [32]. The Sybil attack can severely compromise the authority of trust evaluation, thereby undermining the performance of Social-Chain.

(b) *Bad-mouthing attack:* Adversaries deliberately collude to provide negative feedback on a victim or positive feedback on malicious nodes, resulting in its lower/destroyed or higher/increased reputation. This attack has been discussed in many existing trust management or reputation systems [11, 23].

We consider that an adversary of Social-Chain is mildly adaptive [20], which means that the adversary can corrupt the timestamp validation group, but the corruption takes longer time than the validation period of the group, since the validation group is randomly generated and hard to be identified in a short time. We will further discuss the mildly adaptive assumption in Section 5.2. We also assume honest majority in the Social-Chain system, which means the adversary is not able to corrupt the majority of nodes in Social-Chain.

## 3.3 Research Assumptions

We make the following assumptions in our study based on the system and security models:

- There is no centralized node in PSN responsible for identity and key management;
- We suppose a mining node could be some edge devices that have sufficient computation and storage resources;
- A node's public/private key pair is generated in a secure way and stored in the Trusted Local Database of a node, which is well protected. Non-authorized parties cannot access it;
- Adversaries are not able to break the hash function with a non-negligible probability;
- The communication channels among PSN nodes are secure;
- Due to space limitation, in this article, trust evaluation privacy (including both identity privacy and data privacy) is beyond our focus, since it is another line of our research. Herein, we assume that the real identities of social networking nodes are hidden, and a node pseudonym (e.g., a node public key) is applied in PSN as its identity. With regard to privacy preservation with unlinkability in decentralized trust evaluation, we will report our solution in another article;
- We assume that each node can obtain synchronized timestamp (e.g., from public GPS signals or a reliable time system [14, 15]). Besides, Social-Chain nodes are hard to forge timestamp. This statement is reasonable, since all messages transmitted are monitored by its neighbors and network infrastructures like base stations. Also note that Social-Chain applies a Timestamp Validation algorithm to verify the truth of the timestamp attached to a newly generated block.
- We assume that malicious nodes only occupy a small proportion in Social-Chain.

## 3.4 Notations

For easy presentation, Table 1 summarizes the notations used in the article.

## 4 SYSTEM DESIGN

This section provides the detailed design of Social-Chain. First, we briefly overview Social-Chain and introduce its block structure. Then, we describe its trust evaluation method, followed by the PoT consensus mechanism.

## 4.1 Overview

Social-Chain achieves efficient and decentralized trust evaluation in PSN with the proposed PoT consensus mechanism. The security and effectivity of Social-Chain relies on the PoT consensus mechanism and game theory rather than a centralized party. To be specific, Social-Chain achieves security, effectiveness, and efficiency based on the following justifications:

- Social-Chain leverages a timestamp validation algorithm to ensure the trustworthiness of the timestamp attached to a newly generated block, with which mining nodes can select a unique block from a number of candidates. PoT relies on a trustworthy timestamp to efficiently confirm a block and avoid forks.

Table 1. Notations

| Notations | Definitions |
|---|---|
| $N_i$ | The node $i$ |
| $PK_i$, $SK_i$ | The public and private keys of node $i$ |
| $SIG(m, SK)$ | The signing algorithm working on data $m$ with private key $SK$ |
| $H(\cdot)$ | The hash function |
| $T_k$ | The timestamp of block $k$, which is the signing time of $B_k$ by its creator |
| $B_k$ | The block $k$ |
| $B\_ID_k$ | The ID of block $k$ |
| $TV_{i,k}$ | The trust value of node $i$ in block $k$ |
| $LTV_{i,j}$ | The local trust value of node $j$ held by node $i$ |
| $TO_{i',k}$ | The token issued to node $i'$ for its creation of $B_{k-1}$, which appears in block $k$ |
| $TE_{i,j}$ | The trust evidence in terms of node $i$ on node $j$ |
| $EV$ | The threshold volume of trust evidence |
| $CB_k$ | The content of $B_k$ |
| $IN_{i,j}$ | The times of communication interactions |
| $CV_{i,j}$ | The volume of communication size |
| $F()$ | The trust evidence generation function |
| $T_{TE_{i,j}}$ | The generation time of trust evidence $TE_{i,j}$ |
| $Thr_M$ | The threshold number of approving miners |
| $Thr_T$ | The threshold total sum of the trust values of approving miners |
| $w$ | The size of block sequence window |

- Social-Chain guarantees that the data used for trust evaluation are recorded in the blockchain with consistency and are not modified. PoT ensures trust consensus on trust evaluation results based on commonly referred data.
- Adopting a proper incentive mechanism in Social-Chain can motivate the miners to constantly generate new blocks, so the persistence of Social-Chain is ensured.
- To prevent Sybil attack, Social-Chain sets the initial trust value of a newly joint node as zero. It requires a node to solve a PoW puzzle before it works as a valid miner.

## 4.2 Block Structure

The structure of block $k$ is designed and shown in Figure 2. It contains its previous block's ID, $B\_ID_{k-1}$, which is the hash code of $CB_{k-1}$, i.e., $B\_ID_{k-1} = H(CB_{k-1})$; the timestamp of block $k$, $T_k$, a reward token,

$$TO_{i',k} = \{B\_ID_{k-1}, PK_{i'}, SIG(H(B\_ID_{k-1}, PK_{i'}), SK_X), PK_X\},$$

which can be treated as an award, issued to the miner of the previous block, where $i'$ denotes the miner that creates block $k-1$ and $SK_X$ denotes a set of nodes' private keys. The token is signed by an expected number of nodes whose sum of trust values is above a threshold; the block IDs of used tokens; a list that records the newly updated trust values of nodes; trust evidence records that consist of all trust evidence ordered based on node public keys and reported after the previous block is generated. Evidence can be empty for some nodes in case there are no social interactions happening on them after previous block generation. For saving storage, the evidence can be saved in another place in a secure way (e.g., the cloud and Inter-Planetary File System). Only the abstract of the evidence is recorded in the blockchain. For example, it is possible to employ a secure cloud data storage service with flexible access control based on trust [45–47] to implement

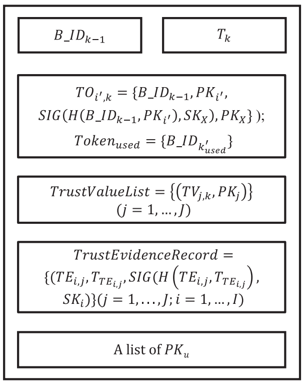

$B\_ID_{k-1}$
$T_k$
$TO_{i',k} = \{B\_ID_{k-1}, PK_{i'}, SIG(H(B\_ID_{k-1}, PK_{i'}), SK_X), PK_X\}$ ;
$Token_{used} = \{B\_ID_{k'_{used}}\}$
$TrustValueList = \{(TV_{j,k}, PK_j)\}$ $(j = 1, \ldots, J)$
$TrustEvidenceRecord = \{(TE_{i,j}, T_{TE_{i,j}}, SIG(H(TE_{i,j}, T_{TE_{i,j}}), SK_i)\}(j = 1, \ldots, J; i = 1, \ldots, I)$
A list of $PK_u$

Fig. 2. Block structure.

this design; only trustworthy miners can access trust evidence data. The trust evidence is a set of signatures on local trust values, interaction times, and communication data volumes issued by the PSN nodes with evidence generation timestamp. In Social-Chain, we do not disclose the details of social networking contents, only statistics with the concern of node privacy. In addition, updated node public keys are also announced in the block if there are any.

## 4.3 Trust Evaluation

In our design, we use $IN_{i,j}$ and $CV_{i,j}$ as the credibility of local trust value $LTV_{i,j}$ to calculate $TE_{i,j}$ with Equation (1), since interaction times and communication volume imply the closeness of social relationship:

$$TE_{i,j} = F(IN_{i,j}, CV_{i,j}, LTV_{i,j}) = \theta 1(IN_{i,j}) * \theta 2(CV_{i,j}) * LTV_{i,j}. \tag{1}$$

Herein, we apply the Rayleigh cumulative distribution function $\theta(g) = 1\text{-}exp\frac{-g^2}{2\sigma^2}$ to model the impact of number $g$. Parameter $\sigma$ can be set as different values in $\theta 1(x)$ and $\theta 2(x)$ to scale the impact of $IN_{i,j}$ and $CV_{i,j}$ on $TE_{i,j}$. The trust evidence is signed by its provider, i.e., $(TE_{i,j}, T_{TE_{i,j}}, SIG(H(TE_{i,j}, T_{TE_{i,j}}), SK_i))$.

To overcome bad-mouthing attack in trust evaluation, we apply the deviation between personal trust and common average trust, as well as past trust value to tailor the contribution of individual trust evidence $TE_{i,j}$ in the trust value calculation. The trust evaluation on node $N_j$ $(j = 1, \ldots, J)$ is performed by the miners during the process of creating a new block based on Equation (2):

$$\begin{aligned} TV_{j,k} = & \frac{1}{(e^{-\frac{|k-k_j|}{\tau}} + 1) * O} \sum_{i=1}^{I} TE_{i,j} * (1 - dv_{i,j}) \\ & + \frac{e^{-\frac{|k-k_j|}{\tau}}}{e^{-\frac{|k-k_j|}{\tau}} + 1} TV_{j,k_j} * e^{-\frac{|k-k_j|}{\tau}}, \end{aligned} \tag{2}$$

where $k_j$ is the block sequential number of the latest previous $TV_{j,k_j}$ appeared in the blockchain; $O = \sum_{i=1}^{I}(1 - dv_{i,j})$. $dv_{i,j} = |TE_{i,j} \text{-} \frac{1}{I}\sum_{i=1}^{I} TE_{i,j}|$ is the trust evidence deviation, which represents the degree of deviation between an evidence value and the average evidence value. We use $1\text{-}dv_{i,j}$ to tailor $TE_{i,j}$ to overcome the negative influence on the accuracy of trust evaluation caused by bad-mouthing attack or malicious/distrusted evidence provided nodes. The higher $dv_{i,j}$ leads to a lower contribution of $TE_{i,j}$ to the trust evaluation. About this, we will provide evaluation based on experiments in Section 6.4.4. Parameter $\tau$ is applied to control the impact of time decaying. The

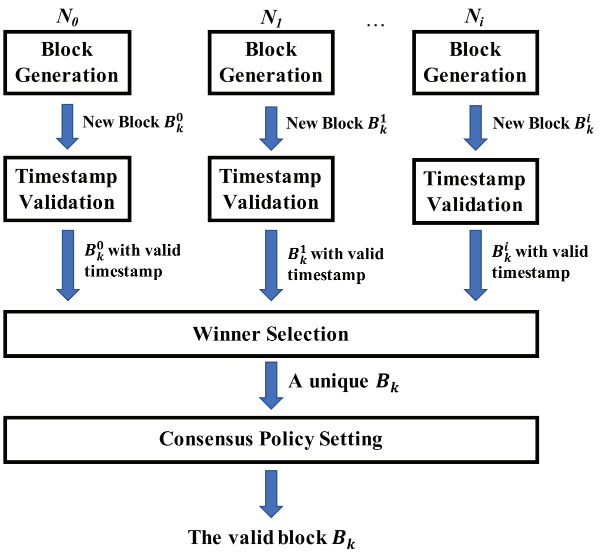


Fig. 3. The procedure of a block generation and validation.

effect of previous trust naturally decays over time. Thereby, the more recent the previous trust value evaluated, the more contribution it makes in the current trust evaluation, the level of which can be adjusted by $\tau$. Note that $TV_{j,k} \in [0, 1]$, the initial trust value $TV_{j,0} = 0$ and $LTV_{i,j} \in [0, 1]$.

In Social-Chain, the moment to generate a new block is decided when the volume of all collected pieces of evidence (i.e., the number of all $(TE_{i,j}, T_{TE_{i,j}}, SIG(H(TE_{i,j}, T_{TE_{i,j}}), SK_i))$ reaches an expected threshold $EV$ after the generation of the previous block. Thus, the trust evidence that should be used for trust evaluation during the process of generating a new block can be verified by all miners. $EV$ can be adjusted based on the agreement of all system miners.

## 4.4 PoT Consensus Mechanism

We propose a novel consensus mechanism, i.e., PoT, for Social-Chain, since existing consensus mechanisms suffer from the shortcomings as described above. As shown in Figure 3, it is composed of four algorithms, i.e., Block Generation, Timestamp Validation, Mining Winner Selection, and Consensus Policy Setting. Among them, Block Generation determines when a miner can create a new block. Different from other consensus mechanisms, PoT enables the miner to generate a new block when sufficient amount of trust evidence has been accumulated to avoid resource-consuming computation and accelerate block generation. Trust evaluation is conducted by the miner by verifying and aggregating the trust evidence to calculate node trust based on the pre-designed trust evaluation algorithm during block generation. The goal of Timestamp Validation is to prevent dishonest miners from attaching a fake timestamp to its generated block. Mining Winner Selection selects a unique block from multiple candidates so blockchain forks can be eliminated. Finally, we employ Consensus Policy Setting to help miners determine whether a valid block can be finally accepted and approved.

**Block Generation.** The algorithm used for generating a new block is shown in Algorithm 1. First, it checks if the volume of trust evidence is sufficient enough to run trust evaluation. When the number of all collected $TE_{i,j} (i = 1, \ldots, I; j = 1, \ldots, J)$ reaches $EV$, it verifies the correctness of all evidence signatures and performs trust evaluation for each involved node based on Equation (2). It then checks the validity of all updated public keys $\{PK_u\}$ and issues $TO_{i',k}$ to the previous

**ALGORITHM 1:** Block Generation

```
Input: $TE_{i,j} = F(IN_{i,j}, CV_{i,j}, LTV_{i,j}),\ (i = 1, \ldots, I; j = 1, \ldots, J); B_l(l = 0, \ldots, k-1)$. Output: $B_k, PK_c, SIG(H(B_k), SK_c)$.
1: while the number of all collected $TE_{i,j}(i = 1, \ldots, I; j = 1, \ldots, J)$ reaches $EV$ do
2:     Verify the correctness of all signatures on evidence.
3:     for $j = 1, \ldots, J; i = 1, \ldots, I$ do
4:         Collect $TE_{i,j} = F(IN_{i,j}, CCV_{i,j}, LTV_{i,j})$;
5:         Calculate $dv_{i,j} basedonTE_{i,j}$;
6:         Seek the lastest $TV_{j,k_j}$ in past blocks;
7:         Calculate $TV_{j,k}$ with (2).
8:     Check the correctness of all updated $PK_u$.
9:     Generate $TO_{i',k}$.
10:    Package $B_k \rightarrow (B_{ID_{k-1}}, T_k, TO_{i',k'}, TV_{j,k}, Token_{used}, PK_u)$.
11: return
```

block creator. With all previous procedures done, the algorithm constructs a new block, which is broadcasted in the network of Social-Chain. Upon receiving the new block, the miners try to reach consensus as discussed below.

In Social-Chain, a block mining winner is awarded a token if its mining work can be approved (i.e., signed) by an expected number of other miners whose total sum of trust values is above a threshold. The threshold is dynamically adjusted according to the total number of miners and the trust values of miners (refer to Algorithm 3 for details). The token can be used for exchanging some social benefits. We design the token issued to node $i'$ for its generation of block $k-1$ as below:

$$TO_{i',k} = \{B\_ID_{k-1}, PK_{i'}, SIG(H(B\_ID_{k-1}, PK_{i'}), SK_X), PK_X\}, \tag{3}$$

where $SK_X$ is a series of private keys that are used to sign the token, and $PK_X$ is the set of their corresponding public keys. The token $TO_{i',k}$ contains the underlying block's ID and the winner's public key. It is signed by an expected number of miners, which is decided by the sum of their trust values, otherwise, the token is not valid. But this token appears in the next block $k$ to prove the acceptance of previous block's generation. This design provides traceability on token issuing by making use of the advantage of blockchain. It also motivates the generation of the initial block, since the creator can gain a token that should be approved by all mining nodes with initial trust value as 0, and thus with a highest value. During token usage, it is easy to know its applicability and correctness by checking the blockchain.

**Timestamp Validation.** Obviously, it is hard to achieve synchronized and reliable timestamp in an open and decentralized environment. Quite possibly, a miner may forge the timestamp and cheat others. Herein, we present a simple countermeasure to this problem by applying Verifiable Computation Function (VRF). Each time the miner generates a new block, it attaches a timestamp to the block. Before it broadcasts the block to all miners, it needs to request signatures from a group of miners to validate the timestamp. Besides, to prevent collusion attack, the miners for confirming the timestamp should be randomly selected.

Suppose the block generated by a miner $M_i$ is $B_k^i$ and its timestamp is $TS_{i,k}$. We assume there is a pseudo-random function $\langle r, \pi \rangle \leftarrow Psu_{\xi}(\cdot)$, where $\xi$ is a seed for random generation and $\pi$ is a proof that indicates the random is correctly calculated. In our scheme, $\xi$ can be selected based on the last block's hash code. Then $M_i$ calculates a value $r = Psu_{H(B_{k-1})}(H(B_k^i)) \bmod M$, and then $M_i$ selects a group of miners for timestamp validation. For a miner $M_j$ whose public key is $PK_{M_j}$, $M_j$

is chosen for timestamp validation if and only if $PK_{M_j}$ satisfies:

$$\frac{H(r||PK_{M_j}) * M}{2^{hlen}} < M_e, \tag{4}$$

where $M_e$ is the expected size of the timestamp validation group, $M$ is the total number of miners, and $hlen$ is the length of the hash value of $H(\cdot)$.

$M_i$ then sends $B_k^i$ to each selected miner, which will check whether timestamp is within a tolerant period. Suppose the local time of miner $M_j$ is $TS_{j,k}$, if $|TS_{j,k} - TS_{i,k}| \leq T_{out}$, where $T_{out}$ is a threshold set by the system. $M_j$ considers $TS_{i,k}$ is valid, signs on $H(B_k^i)$, and sends the signature to $M_i$. Suppose the number of all selected miner is $M_s$; if a miner receives more than $M_s\tau_v$ signatures ($0.5 < \tau_v \leq 1$), the timestamp of its generated $B_k^i$ becomes valid. In our design, we set $\tau_v$ higher than 0.5 so no conflicting decision but a valid decision can be made by the group. We will show our proof on the effectivity of the timestamp validation in Section 5.

**Mining Winner Selection.** In case that multiple miners work out new blocks at the same time, we apply Algorithm 2 to select a winner, which can determine a unique block and thus effectively avoid forks. First, if the node has generated a block in previous $w$ blocks, it will not be allowed to generate another one to ensure decentralization and avoid such a situation that the blockchain is controlled by a powerful node. Second, the validity of the timestamp attached to the new block is verified by checking the correctness of timestamp signatures and the size of its verification group. Third, the node that generates the block at the earliest time wins. This rule aims to ensure the efficiency of block generation. In case two nodes generate the block at the same time, we give the node with higher social trust a higher priority to motivate honest behaviors in PSN. Considering the precision of the trust value and the timestamp are both millesimal, the probability of a tie after taking above two measures is negligible. This implies that the winning miner can be selected uniquely. Even though there is a tie regarding trust value, final selection will refer to the node's public key, e.g., the node with a bigger public key will be selected. When a miner $x$ selects a valid block, it approves the block by signing it $SIG(H(B_{ID_{k-1}}, PK_{i'}), SK_x)$ and then broadcasts it in Social-Chain.

**Consensus Policy Setting.** To achieve a trustworthy consensus on block generation, we need to make decision adaptive to system status. Refer to Algorithm 3: The threshold number of approving miners $Thr_M$ and the threshold total sum of the trust values of approving miners $Thr_T$ are automatically set based on the trust values of all registered miners and the number of miners. We set $Thr_M = \lfloor M^*(1\text{-}\frac{1}{M}\sum_{m=1}^{M} TV_m)\rfloor + 1$, where $M$ denotes the total number of miners and $TV_m$ denotes the trust value of miner $m$ , and $\lfloor M^*(1\text{-}\frac{1}{M}\sum_{m=1}^{M} TV_m)\rfloor = \max\{n \in \mathbb{Z}, n \leq M^*(1\text{-}\frac{1}{M}\sum_{m=1}^{M} TV_m)\}$, $\mathbb{Z}$ is a set of positive integers. The gist of our design is the higher the sum of the trust values of the approving miners, the lower number of approving miners is needed. We also set $Thr_T$ to ensure that the total sum of the trust values of approving miners should be above an expected level, i.e., $Thr_T = \frac{Thr_M}{M}\sum_{m=1}^{M} TV_m$. This design aims to improve the efficiency of consensus and enhance the trustworthiness of block generation, since we try to ensure that the new block generation should be approved by sufficient number of miners with sufficient trust. At the initial time of blockchain generation, all miner nodes' trust values are 0, thus the second block creation should be approved by all miners. It is obvious that the zero initial trust value also ensures that newly joined nodes cannot impact block approval, which can protect our system from Sybil attack to some extent. Based on Algorithm 3, the miners can verify if a newly created block has been approved by sufficient number of miners with sufficient trust. If a valid block is approved by sufficient miners by satisfying both $Thr_T$ and $Thr_M$, a valid $TO_{i',k}$ can be granted. Then, all miners will go ahead to generate the next block.

**ALGORITHM 2:** Mining Winner Selection

**Input:** $N_i$ and $N_j$, which announced creating a new block with timestamp $T_{N_i}$ and $T_{N_j}$; both generate blocks correctly.
**Output:** $N_w$, where $N_w$ is the winner.

```
1: if $N_i (N_j)$ generated a block in previous $w$ blocks then
2:     Reject $N_i (N_j)$;
3:     If both rejected, wait for next generator.
4: if $T_{N_i} \neq T_{N_j}$ then
5:     Set $N_w$ as the node that creates the block at earlier time.
6: else
7:     Set $N_w$ as the node with a higher trust value.
8: return
```

**ALGORITHM 3:** Consensus Policy Setting

**Input:** *$TV_m$: trust value of miner $N_m$; M: the total number of miners.*
**Output:** $Thr_M$, $Thr_T$.

```
1: Set $Thr_M = \lfloor M*(1-\frac{1}{M}\sum_{m=1}^{M} TV_m)\rfloor+1$, where $\lfloor M*(1-\frac{1}{M}\sum_{m=1}^{M} TV_m)\rfloor = \max\{n \in \mathbb{Z}, n \leq M*(1-\frac{1}{M}\sum_{m=1}^{M} TV_m)\}$,
   $\mathbb{Z}$ is a set of positive integers.
2: if $Thr_M \geq M$ then
3:     Set $Thr_M = M-1$.
4: Set $Thr_T = \frac{Thr_M}{M}\sum_{m=1}^{M} TV_m$.
5: return
```

## 4.5 Further Discussions

Social-Chain grants a token to a block creator to incent its miners. The token can be used to run a special social activity or gain some profits (e.g., coupons, permissions, social evidence access priority), at least offered by the nodes whose trust values are updated in the block. We use this incentive mechanism to encourage the mining nodes to maintain the blockchain, since earning a token can gain profits. Note that token consumption can also be recorded in the blockchain to resist double spending. The token holder can only use its token to claim specified profits as defined in the system.

Social-Chain is feasible to offer Trust Evaluation as a Service (TEaS) and aid trust management in other cyber systems, such as crowdsourcing, Internet of Things (IoT), integrated heterogeneous networks (HetNet). For example, a crowdsourcing service provider can employ the trust evaluated by Social-Chain to recruit reliable and honest workers to execute crowdsourcing tasks. It can support authentication on the trust of HetNet nodes or IoT devices for building up a trustworthy networking system. Therefore, Social-Chain has a promising prospect regarding its usage and applications. Though Social-Chain greatly reduces the computation overhead and overcomes the security problems of existing blockchains, it pays little attention to privacy issue. How to protect identity privacy and trust-related data privacy should be seriously considered in Social-Chain. We will solve these issues in our future work.

## 5 SECURITY ANALYSIS

This section analyzes the security of Social-Chain in terms of the following properties: effectivity of timestamp validation, liveness, safety, fault tolerance, and decentralization. Before our formal analysis, we first introduce the model of analysis.

## 5.1 Model of Analysis

In our analysis, we make several assumptions by fully considering practical PSN situations. First, we suppose nodes broadcast trust evidence when they finish social activities and do not communicate with their target communication nodes for at least 30 s. This is reasonable, because a long-time absence of interaction generally signifies the end of a social activity. There could be a large number of nodes that finish social activities at the same time, along with which lots of trust evidence is transmitted in the whole network. Second, each node is just able to receive $p$ percent of trust evidence broadcasted, where $0 \leq p \leq 1$. $p = 0$ means that the node does not get any evidence due to a reason such as network partition [9], and $p = 1$ means that it receives all pieces of evidence. Third, all honest nodes hold equal probability to generate the next block. At last, there exists an adversary that holds an advantage due to its more powerful computing and networking capability than other nodes.

**Network Model.** We adopt a strong synchrony model as described in Algorand [18] herein, which means that most messages (e.g., 95%) sent by honest nodes can be received by most other honest nodes (e.g., 95%) within a known time bound. The adversary could control the network of few honest nodes but cannot manipulate the network at a large scale. And network partition does not happen as assumed. We also describe the network model of information propagation in blockchain [10], which can be considered as a special type of strong synchrony model. Let $t_{i,j}$ be the time in seconds at which node $j$ learns about the existence of block $b$, since it has been sent by node $i$. Then, $t_{i,j}$ follows the exponential distribution as below:

$$P(t_{i,j} = t) = \frac{1}{\lambda} e^{-\frac{1}{\lambda} t}, \tag{5}$$

where $\lambda$ is the average time latency for a miner to receive the block since its generation. In Bitcoin, the average time latency is 12.6 seconds, namely, $\lambda$= 12.6. Latency varies in different blockchain systems.

**Trust Distribution Model.** We further define the trust distribution model of Social-Chain. In Social-Chain, node trust is formalized and varies within a range from 0 to 1. In our analysis, we assume node trust follows Gaussian distribution, with an average trust as $TV_{ave}$ and standard deviation as $\sigma_{TV}$. Then the probability that a node's trust is equal to a certain value $x$ can be calculated as:

$$P(TV = x) = \frac{1}{\sqrt{2\pi}\sigma_{TV}} e^{-\frac{(x - TV_{ave})^2}{2\sigma_{TV}^2}}. \tag{6}$$

Based on the aforementioned assumptions, the network model, and the trust distribution model, we mainly analyze five important security properties of Social-Chain, i.e., effectivity of timestamp validation, liveness, safety, fault tolerance, and decentralization. Among them, effectivity of timestamp validation helps in resisting the attacks caused by a fake timestamp. Liveness ensures that the blockchain system can keep running without interruption. Safety implies that the probability of blockchain fork is negligible, which is crucially important, since the blockchain requires data to be consistent across all nodes. Fault tolerance measures the percentage of compromised nodes that the blockchain system can tolerate, demonstrating the precondition of the security of a blockchain system. Decentralization ensures that the entire system will not be monopolized by a single powerful node or a group of nodes.

## 5.2 Effectiveness of Timestamp Validation

The effectivity of the timestamp validation is the basis of the following security proof. Therefore, we first prove that the timestamp validation ensures the accuracy and trustworthiness of the timestamp of each block to a certain degree. Our proof relies on the following four assumptions with justifications:

*Assumption 1:* The hash of the block created by an honest and uncompromised miner (i.e., $H(B_k^i)$ can be transmitted to the selected verification miners within a small and bounded period, and the upper bound is denoted as $t_t$. This assumption is reasonable, because the size of $H(B_k^i)$ is quite small, and the size of the timestamp validation group is small compared to the size of the whole blockchain network. Given the development of existing communication technologies such as 5G, Wi-Fi, and so on, an honest and uncompromised miner can ensure the low latency of each message's transmission.

*Assumption 2:* The pseudorandom function in our scheme can guarantee the randomness of output with ensured security. This assumption is reasonable, since VRF is widely applied and its effectiveness has been verified in both academia and industry.

*Assumption 3:* The local clock of each honest miner is highly synchronized. The largest deviation between local clocks of honest miners is a small value, denoted as $t_d$.

*Assumption 4 (Mildly Adaptive Adversary):* We consider an adversary of Social-Chain is mildly adaptive [20], which means that the adversary can corrupt the timestamp validation group, but the corruption takes longer time than the validation period of the group, since the validation group is randomly generated and hard to be identified in a short time.

Suppose the size of timestamp validation group is $M_S$, the proportion of honest miners is $\tau_h$, the real timestamp when $B_k^i$ is created is $TS_{i,k}$, and the timestamp attached to $B_k^i$ is $TS'_{i,t}$. For an honest verification miner $M_j$, $TS'_{i,k}$ is considered as valid if the following inequation holds:

$$|TS_{j,k} - TS'_{i,k}| \leq T_{out}, \tag{7}$$

where $TS_{j,k}$ is the time when $M_j$ receives the block $B_k^i$. We set $T_{out} = t_d + t_t$, which ensures that an honest timestamp can be always accepted by honest verification miners. In practice, a rational $M_i$ will always make $TS'_{i,k} < TS_{i,k}$ and try to make $B_k^i$ accepted by others, i.e., $TS_{i,k} - TS'_{i,k} \geq 0$. In this case, we prove the correctness of the following proposition:

*Proposition 1: If the timestamp of* $B_k^i$ attacked by $M_i$ is accepted by an honest miner, the deviation of the timestamp to the real local timestamp of $M_i$ is at most $2t_d + t_t$.

Proof. Suppose the clock of $M_i$ is $t'_d$ ahead of $M_j$, ($-t_d \leq t'_d \leq t_d$), and the transmission time of $H(B_k^i)$ timestamp is $t'_t$, and in this case, we have $TS_{j,k} = TS_{i,k} + t'_d + t'_t$. Then, we have:

$$\begin{aligned} &|TS_{j,k} - TS'_{i,k}| = |TS_{i,k} + t'_d + t'_t - TS'_{i,k}| \leq T_{out} \\ &\rightarrow -T_{out} \leq TS_{i,k} + t'_d + t'_t - TS'_{i,k} \leq T_{out} \\ &\rightarrow -T_{out} - t'_d - t'_t \leq TS_{i,k} - TS'_{i,k} \leq T_{out} - t'_d - t'_t. \end{aligned} \tag{8}$$

Since in practice, $TS_{i,k} - TS'_{i,k} \geq 0$, then, we have:

$$0 \leq TS_{i,k} - TS'_{i.k} \leq T_{out} - t'_d - t'_t. \tag{9}$$

Because $t'_t \geq 0$ and $-t_d \leq t'_d \leq t_d$, and we could estimate the range of $TS_{i,k} - TS'_{i,k}$ is as below:

$$TS_{i,k} - TS'_{i,k} \geq T_{out} + t_d = 2t_d + t_t, \tag{10}$$

where the equation holds if and only if $t'_d = -t_d$ and $t'_t = 0$. Thus, for letting a timestamp pass the verification of an honest miner, its deviation from the real local timestamp is at most $2t_d + t_t$.

**Timestamp Validation Condition.** The verification of a single miner cannot guarantee the trustworthiness of the timestamp attached to a block. Instead, it requires the proportion of honest

Table 2. Probability of ***Case 1*** ($M_S = 100$)

| $\tau_v$ | Probability of ***Case 1*** ($M_S = 100$) | | | | | |
|---|---|---|---|---|---|---|
| | $\tau_h = 0.5$ | $\tau_h = 0.55$ | $\tau_h = 0.6$ | $\tau_h = 0.65$ | $\tau_h = 0.7$ | $\tau_h = 0.8$ |
| 0.51 | 0.4602 | 0.8173 | 0.9729 | 0.9985 | 1.0000 | 1.0000 |
| 0.55 | 0.1841 | 0.5413 | 0.8689 | 0.9850 | 0.9995 | 1.0000 |
| 0.60 | 0.0284 | 0.1831 | 0.5433 | 0.8750 | 0.9875 | 0.9997 |
| 0.65 | 0.0018 | 0.0272 | 0.1795 | 0.5458 | 0.8839 | 0.9906 |
| 0.70 | 0.0000 | 0.0015 | 0.0248 | 0.1730 | 0.5491 | 0.8962 |
| 0.75 | 0.0000 | 0.0000 | 0.0012 | 0.0211 | 0.1631 | 0.5535 |
| 0.80 | 0.0000 | 0.0000 | 0.0000 | 0.0008 | 0.0165 | 0.1488 |

miners in the timestamp validation group is larger than $\tau_v$ ($0.5 < \tau_v \leq 1$). Considering the proportion of honest miners of all miners is $\tau_h$, there are three cases according to different values of $\tau_v$, as described below:

***Case 1:*** The proportion of honest miners is larger than $\tau_v$, and the probability that ***Case 1*** happens is $\sum_{n=\tau_v M_S+1}^{M_S} C_{M_S}^{n} (\tau_h)^n (1-\tau_h)^{M_S-n}$. In this case, an honest timestamp can always pass the verification.

***Case 2:*** The proportion of dishonest miners is larger than $\tau_v$, and the probability of ***Case 2*** is $\sum_{n=\tau_v M_S+1}^{M_S} C_{M_S}^{n} (1-\tau_h)^n (\tau_h)^{M_S-n}$. In this case, if all dishonest miners collude with each other, a dishonest timestamp can pass the verification.

***Case 3:*** For both honest and dishonest miners, their proportion is less than $\tau_v$, and the probability of ***Case 3*** is $1 - \sum_{n=\tau_v M_S+1}^{M_S} C_{M_S}^{n} (\tau_h)^n (1-\tau_h)^{M_S-n} + \sum_{n=\tau_v M_S+1}^{M_S} C_{M_S}^{n} (1-\tau_h)^n (\tau_h)^{M_S-n}$. In this case, a timestamp, whether it is honest or not, cannot pass the verification.

In ***Case 1***, $M_i$ can obtain the correct verification result, and in ***Case 2*** and ***Case 3***, $M_i$ obtains either a wrong verification result or no verification result. Therefore, in practice, we need to maximize the probability of ***Case 1***. Table 2 presents the probability of ***Case 1*** under different values of $\tau_v$ and $\tau_h$, when the group size is 100. From the table, we can see that when $\tau_v = 0.51$ and $\tau_h \geq 0.55$, the probability of ***Case 1*** can be larger than 97%, which means $M_i$ can obtain the correct result with a high probability.

## 5.3 Liveness

*Definition 1 - Liveness.* For any epoch after the blockchain starts, at least a new block will be created within a bounded period of time.

THEOREM 1. *Suppose set* $\{\mathrm{B}_k\}_{k=0,1,\ldots,N_{Block}}$ *is the created blocks that have been accepted by the miners; at least one valid block* $\mathrm{B}_{k+1}$ *will be generated by a miner within a bounded period of time.*

PROOF. As long as social networking is processing among nodes, trust evidences will be created by nodes constantly. The number of trust evidence generated per unit time can be denoted as $\alpha$, and the threshold of required amount of trust evidence to generate a new block is $EV$, then the time to generate a new block should be $\frac{EV}{\alpha}$ plus the time used for block generation, validation, and confirmation, which means there is always a new block created in Social-Chain if there are social activities and the liveness is assured.

## 5.4 Safety

*Definition 1 - Safety.* The system can achieve consistency within a bounded time.

THEOREM 1. *Suppose* $\{\mathrm{B}_k\}_{k=0,1,\ldots,N_{Block}}$ *is a set composed of the created blocks that have been accepted by the miners, where* $\mathrm{k} = 0, 1, \ldots, \mathrm{N}_{Block}$ *is the sequence number of a generated block, then*

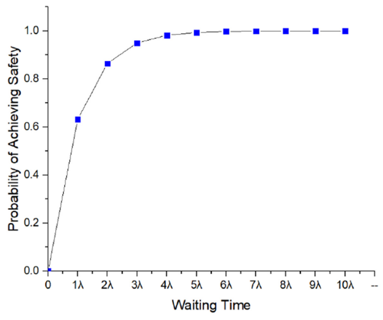


Fig. 4. Probability of achieving safety.

*after a bounded period of time, all honest miners will accept a common block* $\mathrm{B}_{k+1}$ *as the next block of the current blockchain.*

Proof. In the synchrony model, each message (including block, evidence, etc.) can be received by the majority of all miners (e.g., 95%) within a bounded time. Suppose the upper bound of the time is $t_b$; when a miner receives a new block $B^{'}$, it will stop mining and wait for a period with a length of $t_b$. In this way, if there exists another block $B^{''}$ that is generated ahead of $B^{'}$, during the waiting period $t_b$, $B^{''}$ will be received by the majority of the miners according to the definition of strong synchrony model. This ensures that the earliest generated block will surely be received and accepted by the most of the honest miners.

We further take the network model of Bitcoin in Reference [10] as an example to illustrate the safety of Social-Chain. Block propagation delay was demonstrated as the primary reason resulting in blockchain fork [10]. Suppose that the time difference between the new block generated by node $i$ and be received by node $j$ is $t_{i,j}$. Based on the network model, $t_{i,j}$ follows an exponential distribution, and we then have $P(t_{i,j} = t) = \frac{1}{\lambda}e^{-\frac{1}{\lambda}t}$. Cumulative distribution function of $t_{i,j}$ is $P(t_{i,j} < t) = \frac{1}{\lambda}e^{-\frac{1}{\lambda}t}$.

In Social-Chain, for each node, if it receives a new block, it will still wait for a fixed period of time $\theta$ to see if there are other blocks created. If there exists a block generated before the received block, then the probability that it can be received by the node during the waiting period is at least $P(t_{i,j} < \theta)$. Figure 4 shows that $P(t_{i,j} < \theta)$ goes up correspondingly with the increase of $\theta$.

We can see that when a node waits for more than $5\lambda$ of time, the probability of receiving the earliest generated block is 0.9933, which is very close to 1. Thus, we can safely conclude that the earliest generated block can be captured by most nodes with a high probability, as long as the waiting time is long enough.

Based on Algorithm 2, the miners will choose a unique block by checking the timestamp of the block, its creator's trust value, and so on. As analyzed already, timestamp is hard to be forged, so the block with an earlier timestamp and a higher creator's trust value will be selected as the valid new block. To summarize, this unique block will be received and selected by an expected number of nodes with high probability within a predefined certain $\theta$. Therefore, consistency can be achieved within a bounded time period.

## 5.5 Fault Tolerance

*Definition 3 - Fault Tolerance.* For each epoch, all honest miners will accept a common block even when there exists a certain proportion of malicious miners.

Theorem 3. *For each time epoch, all the honest miners agree on a common block even with the presence of malicious miners.*

Table 3. Fault Tolerance with Different $\sigma_{TV}$

| $\sigma_{TV}$ | 0 | 0.02 | 0.05 | 0.1 | 0.15 | 0.166 |
|---|---|---|---|---|---|---|
| Fault Tolerance | 50% | 48% | 45% | 41% | 37% | 36% |

Proof. Herein, we assume all malicious miners are rational and attempt to gain as more as possible benefits. They could approve an illegal block and refuse to sign on a legal block. But it is not possible for them to cut off the process of block propagation. In Algorithm 3, a new block wins when enough nodes that satisfy both $Thr_M$ and $Thr_T$ in the meanwhile have approved this block.

In practice, it is almost impossible for an adversary to control all honest nodes. Therefore, it is profitable for the adversary to compromise as more as possible high-trust nodes to control the blockchain in the real world. Herein, we consider the worst situation, where the adversary compromises the nodes with the highest trust values. In this case, we can obtain the lowest bound of fault tolerance. In our analysis, we calculate the number of nodes that the adversary needs to control to meet the thresholds $Thr_M$ and $Thr_T$ under different settings of $TV_{ave}$ and $\sigma_{TV}$. In particular, we set $TV_{ave}$ as 0.5 to ensure that the whole distribution of trust values falls into [0,1]. Accordingly, $\sigma_{TV}$ ranges from 0.02 to 0.166. The main reason to set $\sigma \leq 0.166$ is that we expect the distribution of trust values mainly falls into the range of [0,1]. When $\sigma \leq 0.166$, the probability that a trust value falls outside [0,1] is less than 0.003, which can be ignored. In this way, we can safely use a normal distribution to model the distribution of trust. If $\sigma > 0.166$, the the probability that a trust value falls outside the range of [0,1] cannot be ignored, which cannot be modelled by the normal distribution. The calculated results of fault tolerance with different $\sigma_{TV}$ when $TV_{ave} = 0.5$ are summarized in Table 3. Obviously, the fault tolerance decreases with the increase of $\sigma_{TV}$. We observe from Table 3 that Social-Chain achieves at least 36% fault tolerance. Actually, it is hard for the adversary to control all the nodes with top trust values, thus resulting in higher fault tolerance than 36%. Therefore, 36% is the lowest bound of fault tolerance of Social-Chain.

## 5.6 Decentralization

*Definition 4 - Decentralization.* Decentralization refers to that even a miner has much more powerful ability than other miners, the miner cannot control the Social-Chain.

Theorem 4. *Supposing there exists a powerful miner with much higher block generation efficiency than others, the miner still cannot control the blockchain by generating most of the valid blocks in the Social-Chain.*

Proof. We employ a state machine to prove decentralization. In particular, we assume that there exists a powerful node with much higher possibility than others to generate a new block. Then, we get the state transition probability of the system and further obtain the probability of each state that the system can reach. We calculate the probability that the powerful miner successfully creates a new block. Finally, based on the analysis, we demonstrate that the probability can be reduced dramatically to a low level by adjusting block sequence window size $w$.

In Social-Chain, when a miner generates a new block, if it has successfully created a block within previous $w$ block sequences, all honest miners will treat this block as invalid and reject it directly. This design disallows a single node to generate a mass number of blocks in a short period. We assume a powerful miner has a probability of $\alpha$ to create a new block ahead of other miners. We then prove that with a proper selection of $w$, Social-Chain can effectively guarantee decentralization.

Each block generation procedure in every epoch can be abstracted as a state machine transition process. We define the system state set $S = \{S_0, S_1, \ldots, S_w\}$, where $S_0$ represents the state that none

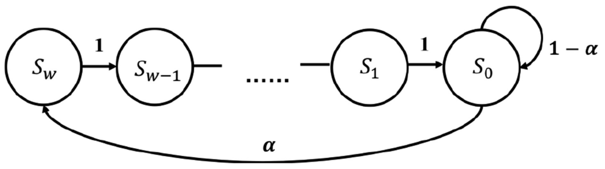


Fig. 5. State machine with transition frequencies.

of the previous $w$ blocks is generated by the powerful miner, and $S_i$ denotes the $i$th block in current sequence window is generated by the powerful node. Figure 5 depicts a state transfer graph. Since the powerful node is not allowed to generate more than one block within the time window, the probability from state $S_{i-1}$ to state $S_{i-1}$ is 1, where $i \neq 0$. When the system reaches state $S_0$, it means the powerful node can create a valid block with probability $\alpha$.

$P(S_i)$ denotes the probability that Social-Chain reaches state $S_i$. We can obtain the following equations based on Figure 5:

$$\begin{aligned} &P(S_0) = P(S_0) * (1-\alpha) + P(S_1) * 1, \\ &P(S_w) = P(S_0) * \alpha, \\ &P(S_i) = P(S_{i+1}) \; for \; 0 < i < w, \\ &\sum_{i=0}^{w} P(S_i) = 1. \end{aligned} \tag{11}$$

With simple algebra, we can easily obtain the probability for the Social-Chain system to reach state $S_0$ as follows:

$$P(S_0) = \frac{1}{1+\alpha w}. \tag{12}$$

Each time the system reaches $S_0$ means that the powerful miner generates a new block that is accepted by all honest miners. Therefore, $P(S_0)$ is also the probability that the miner generates a valid block in Social-Chain. In the worst case, where $\alpha$ is quite near to 1, the powerful miner still cannot gain any advantage over others if we set the size of the sequence window $w$ large enough. We can draw the conclusion that decentralization can be ensured by setting a proper sequence window.

## 5.7 Discussion on Potential Attacks

In this part, we discuss Social-Chain's resistance to various attacks.

**Collusion.** Dishonest nodes may collude with each other by pretending to have social activities for gaining high trust values. However, recording trust evidence in blockchain or storing it somewhere is costly. This helps prevent this kind of collusion attack. Besides, the trust evaluation algorithm can resist collusion attack as well (refer to Section 6). In our opinion, a machine learning–based trust evaluation method can effectively detect such malicious behaviors, which leaves our future work.

**Node Control.** In a permissionless blockchain, the adversary could control multiple mining nodes to violate decentralization. In Social-Chain, the nodes with a low trust value will neither allow to access trust evidence, which makes distrusted miners have little possibility to generate a new block. Besides, we can balance decentralization and Social-Chain maintenance by adjusting $w$. In this way, even though the adversary controls multiple nodes, the number of blocks generated by them can only occupy a small proportion of all blocks.

Table 4. Communication and Computation Complexity

| | Block generation | | | Mining winner selection | |
|---|---|---|---|---|---|
| | Block broadcasting | Signature verification | Trust value evaluation | Timestamp verification | Trust value verification |
| Communication complexity | $O(N)$ | $O(1)$ | $O(1)$ | $O(M_s) + O(N)$ | $O(1)$ |
| Computation complexity | – | $O(N_e)$ | $O(N_e)$ | $O(1)$ | $O(N_e)$ |

**Sybil Attack.** To prevent Sybil attack, Social-Chain assigns a newly joined node with zero trust and requires that a PoW puzzle must be solved before it works as a valid miner. The impact of zero trust will be discussed later in the experiment part. The idea of PoW puzzle is similar to Bitcoin and was also adopted by some other works [5, 25] to mitigate the threat of Sybil attack. The difficulty of PoW puzzle is balanced between efficiency and security. When the node registers itself into Social-Chain, it sends a registration request including the solution to the puzzle, and the blockchain verifies the correctness of the solution. Only if it passes the verification can the node become a valid miner.

**Message Spoofing Attack.** A malicious node may broadcast numerous fake messages, which might result in severe traffic congestion. In Social-Chain, such an abnormal message flooding behavior will lead to trust value deduction indicated by a node local trust evaluation algorithm. A node with a low trust value has trivial impact on block consensus, which reversely harms its own benefits. Moreover, as the number of malicious nodes is limited, such a behavior can be easily detected by deploying probes in the system to disable or block its source.

**Compromised Miner.** The miner in Social-Chain could be vulnerable and easily corrupted. Once a miner is controlled or corrupted by an adversary, the data stored in its local database might be modified, which results in a difference between the views of the miner and other nodes. Thanks to the data consistency of blockchain, the compromised miner can easily detect its difference from others. The reason is that any change of locally stored data will inevitably change the hash value of the latest block, which is different from the hash of the previous block stored in the newly coming block generated by benign miners.

## 6 PERFORMANCE EVALUATION

We implemented Social-Chain in both Windows and Android platforms. Based on the implementation, we tested Social-Chain performance.

### 6.1 Communication and Computation Complexity

The communication and computation complexity of our scheme is presented in Table 4, where $N$ represents the total number of nodes in the system, $N_e$ denotes the number of trust evidence pieces received by the node, and $M_s$ is the group size in the timestamp verification. Furthermore, we conduct the experiments based on the following stated evaluation metrics and testing environment.

### 6.2 Evaluation Metrics

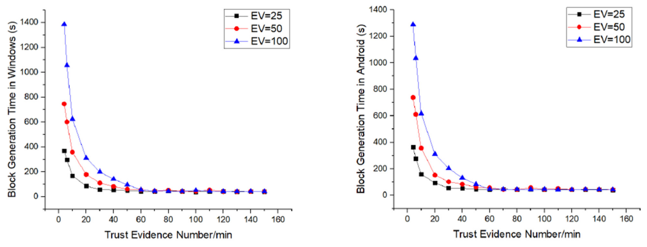
We consider the following metrics to evaluate the performance of Social-Chain:

*(a)* *Block Generation Time*: The block generation time is measured by the average generation time of a single block.

*(b)* *Computational Overhead*: The computational overhead is measured by the CPU utilization rate and the memory usage of a miner device. We observe the CPU utilization rate and memory usage of both desktop and mobile phone under different trust evidence generation rates.

Table 5. System Settings

| Parameters | $\sigma$ | $\tau$ | $EV$ | $v_{evi}$ | $TV_{ave}$ | $N$ |
|---|---|---|---|---|---|---|
| Values | 5 | 10 | 50 | 60 | 0.5 | 100 |



(a) (b)

Fig. 6. (a) Block generation time in Windows; (b) Block generation time in Android.

*(c) Latency*: Latency refers to the time delay between the time trust evidence is sent and the time trust values and evidence are recorded in the blockchain.

*(d) Throughput*: The throughput is measured by the maximum number of the pieces of evidence successfully recorded in the blockchain per second.

*(e) Accuracy of Trust Evaluation*: The accuracy of trust evaluation is measured by the average deviation between the evaluated trust values and the real trust values of nodes.

## 6.3 Experimental Settings

**Implementation Environment:** We implemented Social-Chain with Java language in both Windows platform and Android platform. The Social-Chain node was implemented in a desktop running 64-bit Windows 10 with 2.5 GHz Intel Core i7Quad-CPU and 8 G RAM, and also in an Android phone running 64-bit Android 7.1.0 with 2.2 GHz Snapdragon 820 Quad-CPU and 6 G RAM.

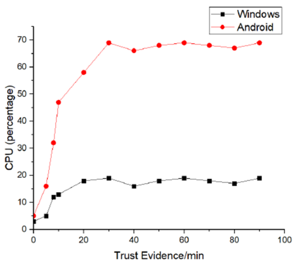
**Node Setup:** In our experiments, we emulated N = 100 nodes in the desktop to form a pervasive social network. Besides, an Android phone is also connected to it. All these nodes act as both miners and PSN users in the emulated network.

**Network Setup:** To better evaluate the performance of our system, we emulated a real-world overlay network by adding a random latency between node communications, so the time of receiving a block announced by a miner follows the exponential distribution as described in Reference [10].

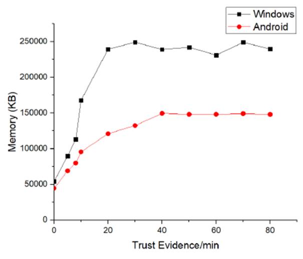
**Social-Chain Setup:** Several parameters were set as fixed values in our implementation, as shown in Table 5. For example, we set $\sigma = 5$, $\tau = 10$, where $\sigma$ is used to scale the impact of $IN_{i,j}$ and $CV_{i,j}$ on $TE_{i,j}$, respectively, and $\tau$ is applied to control time decaying to make a later previous trust value to contribute more in the evaluation of trust. Furthermore, we set trust evidence threshold $EV = 50$, average trust value of all nodes $TV_{ave} = 0.5$ and trust evidence sending rate $v_{evi} = 60$ pieces/min, if we do not mention it explicitly. We observed the performance of Social-Chain under different trust evidence generation rates.

## 6.4 Experimental Results

*6.4.1 Block Generation Time.* In this experiment, we tested block generation time with different trust evidence generation rates in both Windows and Android, as shown in Figure 6. From the figure, we can see that when trust evidence generation rate is low (10 pieces/min), the block

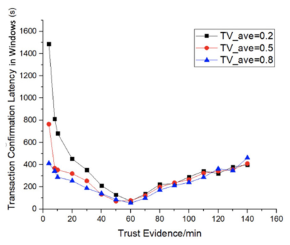


(a)

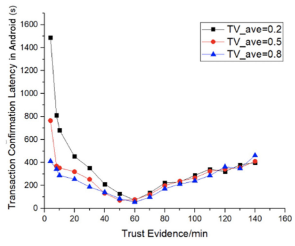


(b)

Fig. 7. (a) CPU utilization with regard to trust evidence creation rate; (b) Memory consumption with regard to trust evidence generation rate.



(a)



(b)

Fig. 8. (a) Confirmation latency in Windows; (b) Confirmation latency in Android.

generation time is long (about 400 s when $EV = 50$). Then, the block generation time drops sharply with the increase of trust evidence generation rate. This is because in Social-Chain, a miner creates a block when the size of received trust evidence reaches the threshold value EV (which equals 25, 50, 100, respectively, in this experiment). When the evidence generation rate increases to 60 pieces per minute, the block generation time reaches its minimum in both Windows and Android, and almost no longer varies as the generation rate increases. The minimum is gained, because the system has reached its maximum processing rate. Therefore, even though more evidence pieces are generated, the system processing rate cannot catch up and the time of new block generation reaches its lowest bound 44 s.

*6.4.2 Computational Overhead.* We tested Social-Chain computational overhead based on the settings listed in Table 5 in both Windows and Android. From Figure 7, we can see that both memory consumption and CPU utilization rise rapidly with the increase of trust evidence generation rate at the beginning and then stabilize at a bounded top. As shown in Figure 7(b), when the system reaches its maximum throughput, memory consumption is 245 MB in Windows and 140 MB in Android. The reason for higher memory consumption in Windows is mainly because multiple nodes are simulated at the same time. CPU utilization is 25% in Windows and 65% in Android due to the limited CPU computing resource in Android. Computational overhead meets an acceptable upper limit, which implies that Social-Chain is feasible enough to be run in a mobile device.

*6.4.3 Latency and Throughput.* We tested the latency of Social-Chain by simulating different trust evidence generation rates and various $TV_{ave}$, which leads to different $Thr_M$ and $Thr_T$. Figure 8 shows the simulation results performed in Windows. Along with the growth of evidence generation rate, trust evaluation confirmation latency drops dramatically at the beginning mainly because

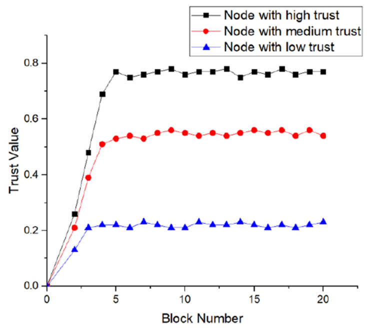


Fig. 9. Evolution of node trust.

*EV* can be satisfied more quickly. In this situation, the confirmation latency is mainly caused by the block generation time, since only when a block is created can trust evaluation results be involved in the block and further reach consensus. However, when the trust evidence generation rate exceeds 60 pieces/min, we can see that the latency slowly increases with the increase of evidence generation rate. This is mainly because the high-speed evidence transmission makes the processing speed of a PSN node slow down. In this experiment, we got the throughput of Social-Chain as 60 pieces of evidence per minute (1 evidence per second), while corresponding confirmation latency is 64 s in both Windows and Android.

*6.4.4 Accuracy of Trust Evaluation.* In this experiment, we set up 100 nodes with initial trust values as 0. Among them, the first 25 nodes did not communicate with other nodes frequently, whose social interaction frequencies were set as 1 time per 3 minutes. The last 25 nodes behaved very actively in social interactions, whose social interaction frequencies were set as 3 times every minute, while the social interaction frequencies of the rest nodes were set as 1 time per minute. The volume of social communication size was set the same for all nodes and $LTV_{i,j} = 0.5$ for $i = 1, \ldots, I$; $j = 1, \ldots, J$. We simulated social interactions according to the above settings. Figure 9 shows the change of average trust values of the three types of nodes, which illustrates their evolution. From Figure 9, we can see that after 10 and 20 epochs, the difference of node trust values with different social interactions is obvious and the trust value distribution of the nodes is consistent with our expectation. The trust of the first 25 nodes is low, the trust of the last 25 nodes is high, and the rest of the nodes hold medium-level trust values. This result demonstrates the accuracy of our evaluation method.

It is obvious that setting the initial trust value as zero can protect the Social-Chain system from Sybil attack, because fully distrusted nodes cannot impact block approval at all. About the protection from bad-mouthing attack, we further conducted an experiment in which malicious nodes can deliberately give negative feedback on a victim node or provide positive feedback on other malicious nodes. Concretely, we randomly selected 10 malicious nodes from the simulated 100 nodes to communicate with other nodes. These malicious nodes were set to generate incorrect trust evidence as $LTV_{i,j} = 1 - TV_{i,k}$ when they interact with other nodes, which means they give opposite trust evidence to disrupt normal trust evaluation. Other setting remains the same as before. Figures 10(a) and (b) show the trust value distribution of the nodes after 10 epochs and 20 epochs of block generation, respectively. We can see the distribution of trust of these nodes are slightly different, and the average deviations of evaluated trust values from normal cases are 3.6% and 5.3%, respectively. Therefore, Social-Chain can alleviate the impact of bad-mouthing attack by applying Equation (2).

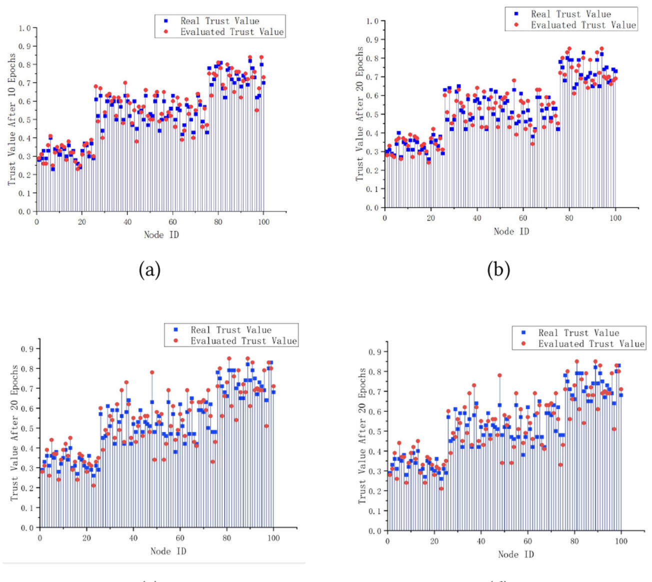


Fig. 10. (a) Real trust values and evaluated trust values after 10 epochs in Social-Chain; (b) Real trust values and evaluated trust values after 20 epochs in Social-Chain; (c) Real trust values and evaluated trust values after 10 epochs in Reference [51]; (d) Real trust values and evaluated trust values after 20 epochs in Reference [51].

In Reference [51], a blockchain-based trust evaluation algorithm was proposed based on Bayesian inference. However, it is vulnerable to the bad-mouthing attack, although it claims that unfair ratings can hardly change the aggregated trust values due to the limited number of attackers. To show this, we conducted an experiment to compare it with the trust evaluation of Social-Chain. The results shown in Figures 10(c) and (d) demonstrate that the average deviations of evaluated trust values from normal cases are 5.7% at the 10th epoch and 8.6% at the 20th epoch, which is higher than ours: 3.6% at the 10th epoch and 5.3% at the 20th epoch. The result indicates that Social-Chain outperforms this work with regard to mitigating bad-mouthing attack.

## 6.5 Comparison

To demonstrate the superiority of Social-Chain over other existing blockchain systems, we compare Social-Chain with several public blockchain systems in terms of security and performance, as shown in Table 6.

We observe that all blockchain systems achieve liveness and safety. Nevertheless, only Social-Chain can effectively guarantee decentralization. By applying the block sequence window, Social-Chain can ensure that the probability that a powerful miner wins can be limited to a low level by simply adjusting the window size. In terms of fault tolerance, EOS performs worse than Social-Chain while others are better. But Social-Chain achieves fault tolerance of 36% only in the worst case, since it is almost impossible for the adversary to control all the most-trusted PSN nodes in practice. Based on the above analysis, Social-Chain can achieve sound security.

In terms of efficiency, Social-Chain does not require a lot of computation, as demonstrated by the experimental results. It can be deployed in personal desktops and even in mobile devices. In addition, with the lightweight PoT consensus mechanism, block generation time and latency are better than Bitcoin and similar to Ethereum. Social-Chain outperforms most of other blockchain

Table 6. Comparison with Other Blockchain Systems

| | Security Property | | | | Performance | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Metrics** | **Liveness** | **Safety** | **Decen-traliza-tion** | **Fault Tolerance** | **Block Generation Time** | **Latency** | **Compu-tational Overhead** | **Throughput** | **Commu-nication Complexity** |
| **Bitcoin [31]** | Yes | Yes | No | 51% | 600 s | 3,600 s | High | 7 transactions per second | $O(N)$ |
| **Ethereum [40]** | Yes | Yes | No | 51% | 12 s | 72 s | Medium | 30 transactions per second | $O(N)$ |
| **EOS [1]** | Yes | Yes | No | 33% (for delegated nodes) | 3 s | 45 s | Low | 3,500 transactions per second | $O(N)$ |
| **QTUM [2]** | Yes | Yes | No | 51% | 14 s | 72,000 s | Medium | 150 transactions per second | $O(N)$ |
| **Waves [3]** | Yes | Yes | No | 51% | 60 s | 6,000 s | Low | 100 transactions per second | $O(N)$ |
| **Algorand [18]** | Yes | Yes | Yes | 33% | - | 40 s | Low | 875 transactions per second | $O(N^2)$ |
| **Social-Chain** | Yes | Yes | Yes | 36% | 44 s | 64 s | Low | 1 evidence per second | $O(N)$ |

systems in terms of computational overhead, because it does not involve computationally expensive operations. In terms of communication complexity, trust evidence and new blocks are broadcasted to other nodes for just one time, and communication complexity is $O(N)$, the same as other systems. For the timestamp validation, the block needs to be broadcasted to all group members, and the group size is $M_s$. Therefore, the communication cost of timestamp validation is $O(M_s)$. As a result, the total communication cost is $O(M_s) + O(N)$. Since in practice, $M_s \leq N$, in the worst case where $M_s = N$, the communication cost is $O(N)$. Considering the fast development of wireless communication (e.g., 5G) that enables mobile devices to communicate with each other with a high speed and a low cost, Social-Chain achieves relatively low communication complexity (i.e., $O(N)$). Therefore, it is possible to deploy PoT in different types of mobile devices. However, the throughput of Social-Chain is not high, which is the direction of our future work.

## 7 CONCLUSION

In this article, we designed and developed Social-Chain, a novel decentralized trust evaluation system in PSN. We proposed the Proof-of-Trust consensus mechanism, which is lightweight, thus can be feasibly deployed in a mass of resource-limited PSN nodes. With extensive security analysis, we proved the security of Social-Chain, which overcomes the risk of centralization and fork issue appearing in many existing blockchain systems. The experimental results further show its effectiveness and efficiency. Comparison with several existing blockchain systems additionally demonstrates the superiority of Social-Chain.

## REFERENCES

[1] Ian Grigg. [n.d.]. EOS-An Introduction. Retrieved on Dec. 18, 2020 from https://www.iang.org/papers/EOS_An_Introduction-BLACK-EDITION.pdf.
[2] 2017. Qtum whitepaper. Retrieved from https://whitepaperdatabase.com/qtum-whitepaper/.
[3] 2016. Waves whitepaper. Retrieved from https://blog.wavesplatform.com/waves-whitepaper-164dd6ca6a23.
[4] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. 2018. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 913–930.
[5] Mohamed Baza, Mahmoud Nabil, Mohamed Mohamed Elsalih Abdelsalam Mahmoud, Niclas Bewermeier, Kemal Fidan, Waleed Alasmary, and Mohamed Abdallah. 2019. Detecting Sybil attacks using proofs of work and location in vanets. *IEEE Trans. Depend. Sec. Comput. arXiv:1904.05845v1.*

[6] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. 2016. Cryptocurrencies without proof of work. In *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, 142–157.

[7] Vitalik Buterin and Virgil Griffith. 2017. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437* (2017).

[8] Miguel Castro, Barbara Liskov, et al. 1999. Practical Byzantine fault tolerance. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI '99)*, Vol. 99. 173–186.

[9] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. 2016. On scaling decentralized blockchains. In *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, 106–125.

[10] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the bitcoin network. In *Proceedings of the IEEE P2P Conference*. IEEE, 1–10.

[11] Chrysanthos Dellarocas. 2000. Mechanisms for coping with unfair ratings and discriminatory behavior in online reputation reporting systems. In *Proceedings of the International Conference on Information Systems (ICIS'00)*. 52.

[12] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert Van Renesse. 2016. Bitcoin-NG: A scalable blockchain protocol. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*. 45–59.

[13] Ittay Eyal and Emin Sirer. 2018. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM* 61, 7 (2018), 95–102.

[14] Kai Fan, Yanhui Ren, Zheng Yan, Shangyang Wang, Hui Li, and Yintang Yang. 2018. Secure Time synchronization scheme in IoT based on blockchain. *IEEE Blockchain* (2018).

[15] Kai Fan, Shangyang Wang, Yanhui Ren, Kan Yang, Zheng Yan, Hui Li, and Yintang Yang. 2018. Blockchain-based secure time protection scheme in IoT. *IEEE Internet Things J.* (2018).

[16] Dongqi Fu and Liri Fang. 2016. Blockchain-based trusted computing in social network. In *Proceedings of the 2nd IEEE International Conference on Computer and Communications (ICCC'16)*. IEEE, 19–22.

[17] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert Van Renesse, and Emin Gün Sirer. 2018. Decentralization in Bitcoin and Ethereum networks. *arXiv preprint arXiv:1801.03998* (2018).

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

[19] Linke Guo, Chi Zhang, and Yuguang Fang. 2015. A trust-based privacy-preserving friend recommendation scheme for online social networks. *IEEE Trans. Depend. Sec. Comput.* 12, 4 (2015), 413–427.

[20] Timo Hanke, Mahnush Movahedi, and Dominic Williams. 2018. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548* (2018).

[21] Fei Hao, Geyong Min, Man Lin, Changqing Luo, and Laurence T. Yang. 2014. MobiFuzzyTrust: An efficient fuzzy trust inference mechanism in mobile social networks. *IEEE Trans. Parallel Distrib. Syst.* 25, 11 (2014), 2944–2955.

[22] Wenjun Jiang, Jie Wu, Feng Li, Guojun Wang, and Huanyang Zheng. 2016. Trust evaluation in online social networks using generalized network flow. *IEEE Trans. Comput.* 65, 3 (2016), 952–963.

[23] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. 2003. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International Conference on World Wide Web*. ACM, 640–651.

[24] Sunny King and Scott Nadal. 2012. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. *Self-published Paper, August* 19 (2012). Retrieved on December 18. 2020 from https://decred.org/research/king2012.pdf.

[25] Marek Klonowski and Micha B. Koza. 2013. Countermeasures against Sybil attacks in WSN based on proofs-of-work. In *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 179–184.

[26] Daniel Larimer. 2014. Delegated proof-of-stake (DPOS). *Bitshare Whitepaper* (2014). Retrieved on December 2020 from https://bitcointalk.org/index.php?topic=558316.msg6082114#msg6082114.

[27] Guanfeng Liu, Yan Wang, Mehmet A. Orgun, and Ee-Peng Lim. 2013. Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks. *IEEE Trans. Serv. Comput.* 6, 2 (2013), 152–167.

[28] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 17–30.

[29] Dianhui Mao, Fan Wang, Zhihao Hao, and Haisheng Li. 2018. Credit evaluation system based on blockchain for multiple stakeholders in the food supply chain. *Int. J. Environ. Res. Pub. Health* 15, 8 (2018), 1627.

[30] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The honey badger of BFT protocols. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 31–42.

[31] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008). Retrieved on December 18, 2020 from https://bitcoin.org/bitcoin.pdf.

[32] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. 2004. The Sybil attack in sensor networks: Analysis & defenses. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*. IEEE, 259–268.

[33] Daejun Park, Yi Zhang, Manasvi Saxena, Philip Daian, and Grigore Roşu. 2018. A formal verification tool for Ethereum VM Bytecode. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 912–915.

[34] Maxim Raya, Panagiotis Papadimitratos, Virgil D. Gligor, and J.-P. Hubaux. 2008. On data-centric trust establishment in ephemeral ad hoc networks. In *Proceedings of the IEEE 27th Conference on Computer Communications*. IEEE, 1238–1246.

[35] Paul Resnick and Richard Zeckhauser. 2002. Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. In *The Economics of the Internet and E-commerce*. Emerald Group Publishing Limited, 127–157. Retrieved on December 18, 2020 from http://presnick.people.si.umich.edu/papers/ebayNBER/RZNBERBodegaBay.pdf.

[36] Emre Sarigöl, Oriana Riva, Patrick Stuedi, and Gustavo Alonso. 2009. Enabling social networking in ad hoc networks of mobile phones. *Proc. VLDB Endow.* 2, 2 (2009), 1634–1637.

[37] Jian Shen, Tianqi Zhou, Chin-Feng Lai, Jiguo Li, and Xiong Li. 2017. Hierarchical trust level evaluation for pervasive social networking. *IEEE Access* 5 (2017), 1178–1187.

[38] Melanie Swan. 2015. *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc.

[39] Wenbo Wang, Dinh Thai Hoang, Zehui Xiong, Dusit Niyato, Ping Wang, Peizhao Hu, and Yonggang Wen. 2018. A survey on consensus mechanisms and mining management in blockchain networks. *arXiv preprint arXiv:1805.02707* (2018).

[40] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yell. Pap.* 151 (2014), 1–32.

[41] Ronghua Xu, Xuheng Lin, Qi Dong, and Yu Chen. 2018. Constructing trustworthy and safe communities on a blockchain-enabled social credits system. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 449–453.

[42] Zheng Yan, Yu Chen, and Yue Shen. 2013. A practical reputation system for pervasive social chatting. *J. Comput. Syst. Sci.* 79, 5 (2013), 556–572.

[43] Zheng Yan, Yu Chen, and Yue Shen. 2014. PerContRep: A practical reputation system for pervasive content services. *J. Supercomput.* 70, 3 (2014), 1051–1074.

[44] Zheng Yan, Wei Feng, and Pu Wang. 2015. Anonymous authentication for trustworthy pervasive social networking. *IEEE Trans. Comput. Soc. Syst.* 2, 3 (2015), 88–98.

[45] Zheng Yan, Xueyun Li, and Raimo Kantola. 2015. Controlling cloud data access based on reputation. *Mob. Netw. Applic.* 20, 6 (2015), 828–839.

[46] Zheng Yan, Xueyun Li, Mingjun Wang, and Athanasios V. Vasilakos. 2015. Flexible data access control based on trust and reputation in cloud computing. *IEEE Trans. Cloud Comput.* 5, 3 (2015), 485–498.

[47] Zheng Yan and Wangyang Shi. 2017. CloudFile: A cloud data access control system based on mobile social trust. *J. Netw. Comput. Applic.* 86 (2017), 46–58.

[48] Zheng Yan, Honggang Wang, Laurence T. Yang, and Valtteri Niemi. 2018. Special section editorial: Trust management in pervasive social networking (TruPSN). *IEEE Access* 6 (2018), 16851–16854.

[49] Zheng Yan and Mingjun Wang. 2017. Protect pervasive social networking based on two-dimensional trust levels. *IEEE Syst. J.* 11, 1 (2017), 207–218.

[50] Zheng Yan, Pu Wang, and Wei Feng. 2018. A novel scheme of anonymous authentication on trust in pervasive social networking. *Inf. Sci.* 445 (2018), 79–96.

[51] Zhe Yang, Kan Yang, Lei Lei, Kan Zheng, and Victor C. M. Leung. 2019. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet Things J.* 6, 2 (2019), 1495–1505.

[52] Zhe Yang, Kan Zheng, Kan Yang, and Victor C. M. Leung. 2017. A blockchain-based reputation system for data credibility assessment in vehicular networks. In *Proceedings of the IEEE 28th International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC'17)*. IEEE, 1–5.

[53] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. 2018. RapidChain: Scaling blockchain via full sharding. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 931–948.

[54] Bo Zhang, Ruihan Yong, Meizi Li, Jianguo Pan, and Jifeng Huang. 2017. A hybrid trust evaluation framework for e-commerce in online social network: A factor enrichment perspective. *IEEE Access* 5 (2017), 7080–7096.

Received April 2020; revised July 2020; accepted August 2020