Future Generation Computer Systems 95 (2019) 649–666

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

MCS-Chain: Decentralized and trustworthy mobile crowdsourcing based on blockchain

Wei Feng [a,b,c], Zheng Yan [a,c,*]

[a] State Key Laboratory of ISN, School of Cyber Engineering, Xidian University, Xi'an, China
[b] Key Lab of Information Network Security, Ministry of Public Security, Shanghai, China
[c] Department of Communications and Networking, Aalto University, Espoo, Finland

HIGHLIGHTS

- MCS-Chain is a novel blockchain-based mobile crowdsourcing system.
- A consensus mechanism is designed for new block generation to avoid forks.
- A trust evaluation mechanism is applied for worker selection.
- MCS-Chain is secure w.r.t. liveness, decentralization, and fault tolerance.
- MCS-Chain is effective and efficient compared with Bitcoin and Ethereum.

ARTICLE INFO

Article history:
Received 30 October 2018
Received in revised form 30 December 2018
Accepted 13 January 2019
Available online 2 February 2019

Keywords:
Blockchain
Mobile Crowdsourcing (MCS)
Trust
Decentralization

ABSTRACT

Mobile Crowdsourcing (MCS) is an effective and novel method of data collection and processing. Current MCS generally adopts a centralized architecture by depending on an assumed trusted party. This design easily suffers from single-point failure and cannot be realized in practice since a trusted service provider does not really exist. More dangerously, the centralized party may perform dishonestly and thus harms the benefit and privacy of MCS users. To tackle these problems, we propose a novel blockchain-based MCS system, named MCS-Chain, to realize fully distributed and decentralized trust management in MCS. Aiming at improving the poor efficiency of traditional blockchain technology, we propose a novel consensus mechanism for block generation, which greatly reduces computational overhead. The proposed MCS-Chain system also solves the fork issue and centralization problem suffered by most existing blockchain-based systems. Serious security analysis and experimental evaluation further illustrate the security and efficiency of our system.

1. Introduction

Smart devices, like smart phones and wearable devices, are widely equipped. As a result, mobile crowdsourcing (MCS) has emerged as an effective method for data collection and processing [1]. It leverages existing mobile devices as sensors to collect various data about temperature, weather, crowd density, voice/video/image, etc. MCS can also finish some tasks that remain difficult for computers but easy for human beings. Nowadays, applications based on MCS are developed in various fields, such as smart transportation [2,3], public safety [4], environmental monitoring [5], and infrastructure monitoring. We can find many MCS based applications have been in use in industry, examples are Waze [6], Moovit [6], Weathermob [7], Minutely [7], WeatherSignal [8], OpenSignal [8], etc.

A typical MCS system is composed of three types of entities: end user, worker, and service provider. End users issue tasks to a service provider, which will publish the tasks to recruit workers for completion. The service provider acts as an MCS platform to receive and publish MCS tasks to the workers that accept the tasks, perform the tasks, and return the results of task execution to the MCS service provider. Generally, the MCS service provider is also responsible for worker recruitment and task assignment, data collection, data processing, and evaluating trust of other system entities. It returns the final result of task execution to the end user. MCS owns several advantages over traditional wireless sensor networking (WSN) in terms of availability and flexibility because it leverages mobile devices as sensors and thus is free from high sensor deployment expense.

Despite the advantage of MCS, it still confronts a number of challenges related to security and privacy. A traditional MCS system adopts a centralized architecture and assumes that there exists a trustworthy and centralized service provider. This design leads to

* Corresponding author at: State Key Laboratory of ISN, School of Cyber Engineering, Xidian University, Xi'an, China.
E-mail addresses: zyan@xidian.edu.cn, zheng.yan@aalto.fi (Z. Yan).

https://doi.org/10.1016/j.future.2019.01.036
0167-739X/

several shortcomings. First, it is easy to suffer from single point of failure because the system relies on the security and trust of the centralized service provider. Once it is intruded and broken, the whole system will crash. Second, potential privacy leakage is high. The service provider normally keeps sensitive information of both end users and workers, e.g., identities, task information, location information, etc. The compromise of the service provider will lead to privacy disclosure. Since the service provider may be curious about the privacy of both end users and workers, it may not follow the pre-defined protocols in order to gain more benefits, hence harm the privacy and profits of both end users and workers. It may also analyze its available information to infer additional privacy of end users and workers.

There have been many works to deal with security, privacy, and trust issues in MCS. Homomorphic encryption and pseudonym mechanism have been widely leveraged to solve data privacy and identity privacy problems, respectively [9,10]. Homomorphic encryption enables data processing with privacy preservation. Some schemes employ differential privacy to preserve location privacy of MCS workers [11]. To address false data uploading attack by workers, several solutions were proposed. Prandi et al. proposed to evaluate data trustworthiness by comparing the collected data with a gold data set in which the data is authorized and correct [12]; Zhang et al. employs K-means to find the truth from unreliable data and resist the negative impact caused by false data [13]. Other schemes aim at resisting the dishonest behaviors of the MCS service provider. For example, some incentive schemes provide a verification mechanism for workers to verify whether the MCS service provider performs according to a predefined protocol. Nevertheless, the aforementioned methods still adopt a centralized MCS architecture. They usually fail to resist dishonest behaviors of the centralized service provider. Neither can they solve the single-point failure. Although some schemes embed a verification mechanism into an incentive mechanism so that workers and end users can verify the correctness of the payment calculated by the service provider, they cannot guarantee the trust of other procedures in MCS. We can see that the traditional MCS system design based on centralized trust is not proper and impractical. MCS is a distributed system. Decentralized trust management is highly expected in MCS.

Blockchain, which is the backbone of the Bitcoin system [14], is a popular technique to support decentralization in distributed systems. Due to its properties of tamper-proof, transparency and decentralization support, it is widely paid special attention in recent years in both industry and academia. In spite of its popularity, the underlying consensus algorithm in Bitcoin suffers from low-efficiency and high resource consumption. Several improved consensus algorithms have been developed, such as Proof of Stake (PoS) [15], Delegated Proof of Stake (DPoS) [16], Bitcoin the next generation (Bitcoin-NG) [17], etc., but none of them avoid the occurrence of chain fork. To be specific, most of them requires waiting several blocks to check whether current chain is the longest chain, and thus suffers from high latency and is not efficient at all. Apart from the fork, most public blockchain faces the problem to ensure decentralization. A powerful miner or mining pool may control the blockchain by generating most blocks, and thus harms the trust of blockchain [18]. Therefore, it is necessary to solve the problems of fork and centralization issue of blockchain.

For decentralizing trust management in MCS and overcoming the weakness of existing blockchain techniques, we propose a novel MCS system, named MCS-Chain, by innovating a newly designed blockchain to realize fully distributed and decentralized trust management in MCS. Aiming at improving the poor efficiency of traditional blockchain technology, we propose a novel blockchain that greatly reduces computational overhead. The MCS-Chain also solves the fork issue and centralization problem suffered by most existing blockchain-based systems. MCS-Chain is suitable to be applied into the MCS where there exists no centralized trustworthy party. Specially, the main contributions of the paper can be summarized as follows:

- We design MCS-Chain, a novel blockchain-based MCS system, where the generation of a new block is determined by the total amount of payment records waiting to be stored in the next block. We design a new consensus algorithm for new block confirmation. It guarantees that a unique block can be determined even if several generated blocks appear simultaneously.
- Under the condition of no trusted party existing in the system, we propose a trust evaluation mechanism for MCS-Chain, with which end users can choose reliable workers.
- We analyze the security and availability of the system. To be specific, we theoretically prove safety, liveness, decentralization, and fault tolerance of the proposed MCS-Chain system, thus demonstrate the security of MCS-Chain.
- We implement MCS-Chain in both Windows and Android devices and conduct several experiments based on the implementation in order to evaluate MCS-Chain performance. Experimental results show the effectiveness and efficiency of our proposed system.

The remainder of the paper is organized as follows. We give a brief review on related work in Section 2. In Section 3, we present problem statement by describing a system and security model with our research assumptions. The detailed design of the proposed MCS-Chain system is presented in Section 4, followed by security analysis and performance evaluation in Section 5. Finally, we conclude the paper in the last section.

2. Related work

2.1. Mobile crowdsourcing

Despite the advantages of MCS, it faces several challenges in security, privacy, and trust. First, data collected by MCS is probably sensitive to worker privacy. Second, the content of a task could reveal the sensitive information of MCS users. Besides, data collected by MCS workers vary in quality because of imparity in workers' abilities and trust. Presently, numerous works were proposed to deal with these issues. Current works mainly concentrate on three aspects, i.e., incentive mechanism [19,20], trust evaluation [21,22], and truth discovery [13,23]. Among them, an incentive mechanism aims at selecting a group of reliable and trustworthy workers to conduct a task [19,20]; trust evaluation is performed for the purpose of evaluating the trust of workers [21,22]; truth discovery attempts to find the truth from collected distrusted or noisy data [13,23]. For example, some researchers built an MCS incentive mechanism based on game theory. By maximizing social welfare, these incentive mechanisms guarantee that the workers cannot get more benefits than behaving honestly if they bid tasks with false attributes. As a result, they successfully guarantee the trust of workers; however, they fail to prevent dishonest behaviors of a centralized service provider. To resist dishonest behaviors of the service provider, some incentive mechanisms offer a verification mechanism, with which workers can verify whether the centralized service provider follow the incentive mechanism and honestly calculate the possible payments for the workers. Effective as they are, the verification process requires the participation of all workers. Therefore, they are not practical and cannot resist a collusion attack between service provider and workers. Some works focus on truth discovery, of which the goal is to process raw data and exclude the negative impact caused by unreliable or fake data.

All the schemes mentioned above adopt a centralized architecture, while a centralized service provider is rational and probably behaves dishonestly out of their own benefit. Besides, the centralized MCS architecture is vulnerable to single point of failure. For solving these problems, some schemes were developed to alleviate the negative impact of dishonest behaviors [9,10,12,24–27]. For example, Zhang et al. [9] designed a secure and dependable incentive mechanism, in which the crowds of workers are randomly divided into two groups with different sizes. The service provider estimates the unit payment with the smaller group and a limited budget. After the service provider selects the workers and decides corresponding payments, all the workers can verify the correctness of results based on public information. However, it requires the participation of all the workers in the small groups, and thereby is not practical. Besides, the verification may fail if the service provider colludes with some workers. To resist single point of failure, some workers build MCS database in a distributed way [26]. Data are stored in several cloud databases rather than a single one. Data owners can decide data access policy by themselves. In this way, even some of the databases are compromised, attackers cannot access all the data. However, this existing work only builds a distributed storage mechanism, which cannot be applied to solve other issues of MCS, such as truth discovery and incentive mechanism. To conclude, how to overcome the security, privacy, and trust problems of MCS caused by a distrusted centralized service provider still remains an open issue.

2.2. Blockchain

Blockchain is the backbone technique of Bitcoin [14] and Ethereum [15] and was first applied to build distributed cryptocurrency and smart contract platforms. It allows all nodes to verify the correctness of the content in the blockchain and has such intrinsic advantages as tamper proof, transparency, decentralization support, and consistency. Currently, blockchain has attracted vast attention in both academia and industry. There are numerous works to address its applications in various areas, such as smart contracts, smart transportation [28], supply chain [29,30], data management [31], trust evaluation [32], etc. Popular as it is, blockchain still faces many challenges in terms of security, privacy, efficiency, and scalability. First, it faces a number of security threats like selfish mining [17], 51% attack, double spending [14], eclipse attack [33], etc. Second, although blockchain realizes anonymity, it fails to achieve transaction unlinkability. Besides, it cannot preserve transaction privacy since all transaction information is public on the chain [34]. Third, most blockchain based systems suffers from low throughput and high resource consumption [35]. Furthermore, fork exists in most public blockchain designs, which makes miners have to wait for several blocks to achieve eventual consistency, which leads to high consistency delay and long transaction confirmation time. Even worse, some blockchain systems, like Bitcoin and Ethereum, face a centralization problem [18] since block generation could be controlled by a limited small number of miners.

There are already some attempts to address the above problems. Heilman et al. leveraged blind signature to enhance transaction privacy [36]. Eyal et al. proposed a simple and backwards-compatible change to the Bitcoin protocol to resist selfish mining. When a miner finds out competing branches with the same length, it chooses to mine one of these branches randomly, and propagates all these branches to other miners [17]. People have also proposed a number of improved blockchain consensus protocols to reduce computation resource and achieve better scalability. Eyal et al. proposed a scalable Bitcoin protocol called Bitcoin-NG. Bitcoin-NG consists of two types of blocks, i.e., the block for leader election and the micro block containing ledger entries [16]. For miners, it is easier to create a micro block than a block for leader election. This increases block generation frequency and hence transactions can be recorded in blockchain quickly. In this way, bitcoin-NG achieves better scalability than Bitcoin. Ethereum leverages Proof-of-Stake (PoS) instead of Proof-of-Work (PoW) as its underlying consensus mechanism [15]. In this way, the computation overhead of Ethereum is greatly reduced. Besides, it adopts the concept of uncle block to alleviate the impact of fork [15]. Motivated by PoS, Delegated Proof of Stake (DPoS) was proposed [16]. In DPoS, a subset of miners is elected as block producers to validate the chain of blocks. DPoS divides time into a series of time slots, and a block producer is responsible for block generation for an assigned block. As a result, DPoS is able to generate new blocks with high efficiency and can achieve much better throughput. Nevertheless, this design sacrifices the property of decentralization. In summary, due to the current drawbacks of blockchain, this technique cannot be directly applied into MCS to realize decentralized trust management.

The development of blockchain motivates researchers to explore the application of blockchain in MCS. Thus far, there are already some works concerning blockchain based or assisted MCS systems. Some researchers directly apply blockchain into MCS to build a secure MCS system [28,37–43]. For example, Yuan and Wang proposed a blockchain based intelligent transportation system for transportation information collection [28]; Li et al. designed a blockchain based decentralized framework for crowdsourcing called CrowdBC, which utilizes smart contract to manage MCS task execution [37,42]. Similarly, Wang et al. employs blockchain in crowdsourced energy systems [38]; Pinto et al. employs blockchain to build a decentralized public key infrastructure (PKI) for crowdsourced Internet of Things (IoT) [39]; in [41], a blockchain based system for real-estate crowdsourcing was proposed, where blockchain works as a ledger to record agreements so that they cannot be modified; Hu et al. leverages blockchain to build a reputation based decentralized knowledge sharing system, and employs a trusted storage server to reduce the burden of miners [43]. These schemes can effectively overcome the negative effect of a dishonest centralized service provider. Nonetheless, they all use existing blockchain schemes. As a result, they inherit the shortcomings of existing blockchain techniques and cannot overcome their weakness, such as fork, poor scalability, etc.

To tackle these problems, some people improved the blockchain to adjust it to MCS scenarios. For example, Bhatia et al. proposed a decentralized MCS based on Ethereum [44]. When posting a transaction in blockchain, the poster is required to mine the blockchain. In this way, it can provide more miners to working on maintaining the blockchain when there are amounts of transactions. Although the increase of miner number helps reduce the cost of transaction verification, it is not the only reason that constrains the throughput of blockchain. Besides, it still cannot solve the fork issue and centralization in Ethereum. Fujihara proposed a blockchain incentivized transportation information gathering system [45]. In this scheme, traffic information is recorded in blockchain, and blockchain based cryptocurrency is employed to incentivize cars to upload traffic information. To achieve satisfactory performance, an area is divided into many segments, and each of them maintains a blockchain. When an accident happens, the blockchain in this segment is forked into two separate blockchains because the road becomes impassable. Considering this, the scheme takes advantage of blockchain fork to detect road accident, which is very novel and effective. Nonetheless, its application is limited to some certain crowdsourcing scenarios. In real world, when an end user submits a task to blockchain, it is not practical for the user to request several miners for information in different blockchains. Besides, different from transportation, most MCS scenarios are dynamic and lack fix infrastructure as roadside unit (RSU), and thus by dividing an area into several segments and maintaining several

blockchains is not suitable for common MCS tasks. In [46], Zou et al. builds a blockchain based MCS system, which applies a consensus mechanism improved with trust to enhance scalability. The system adopts a hybrid architecture, where several stakeholders compose a permissioned consortium blockchain for accountability. In spite its improvements on crowdsourcing accountability, it sacrifices decentralization to some extend. The scheme cannot totally eliminate the risk of single point of failure. Buccafurri et al. proposed an alternative to blockchain called Tweetchain based on social networking to realize decentralized MCS [47]. It takes public posts transmitted through social networks to build a meshed chain. The main idea of Tweetchain is leveraging existing social networks (for example, Twitter) to publish transactions. Each transaction consists of a signature and a hash value from a hash chain, thus the service provider of social network cannot forge, delete, or alter transactions. Tweetchain has better scalability than blockchain since it employs centralized service providers of social networks to publish transactions. However, it faces single point of failure since it introduces a centralized entity. If the centralized entity is compromised or maliciously deny the publication of transactions, the availability of Tweetchain will be destroyed. To conclude our related work review, how to reform blockchain design to make it effective for MCS is still an open issue.

3. Problem statement

In this section, we introduce MCS-Chain system model and security model, as well as our research assumptions and design goals. For easy presentation, we also describe the notations used in this paper at the end of this section.

3.1. System model

Fig. 1 illustrates the system model of MCS-Chain. MCS-Chain contains a number of nodes (including various mobile devices) connected with each other through various networks like cellular networks, mobile ad-hoc networks, Wi-Fi, Bluetooth, and so on. MCS nodes can be classified into three types: end user, worker, and miner, and each node can act as either an end user, a worker, or a miner. Among them, the miners cooperatively maintain and manage the blockchain of MCS-Chain innovated for mobile crowdsourcing. The blockchain works as an MCS platform, records MCS procedures, and evaluates trust of all system entities. In MCS-Chain, each miner keeps a copy of the blockchain and can access the data stored off the blockchain. An end user can be an individual or organization that lacks the ability to perform a certain task, e.g., data collection and processing. It requests task fulfillment by offering proper payment to task executors. Besides, it also provides a certain amount of service fee to the miners to motivate them to honestly record and verify the related information of task execution. MCS workers are the nodes that participate in crowdsourcing and perform the assigned tasks based on agreement. There are mainly three kinds of workers, i.e., sensing workers, computing workers, and storage workers. The difference between them lies in the different tasks they conduct. To be specific, the sensing workers leverage mobile devices as sensors to collect environmental data, like images, voice, temperature, etc., or collect opinions or personal data from device holders; the computing workers perform computing tasks and submit computing results to the end user. The storage workers offer data storage services with secure data access control.

Each node contains a number of basic functional modules as shown in Fig. 2 and as described below. An MCS-Chain App is deployed to perform the basic functions of MCS, e.g., task request, task bidding, task assignment, payment, and performance feedback. Blockchain UI displays the contents of MCS-Chain blockchain.

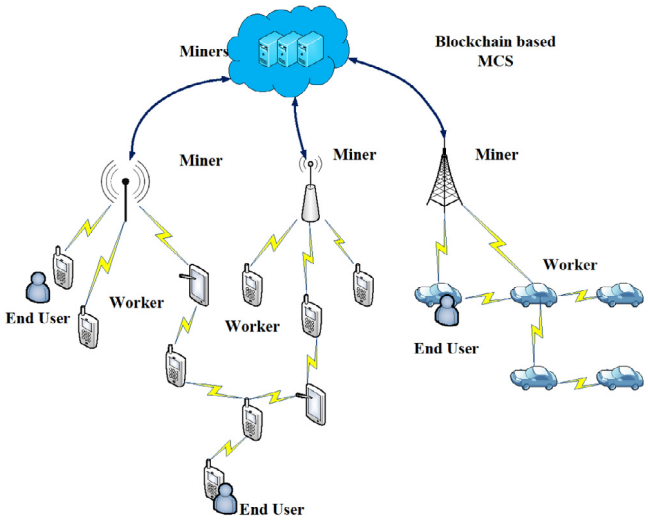

Fig. 1. MCS-Chain system model.

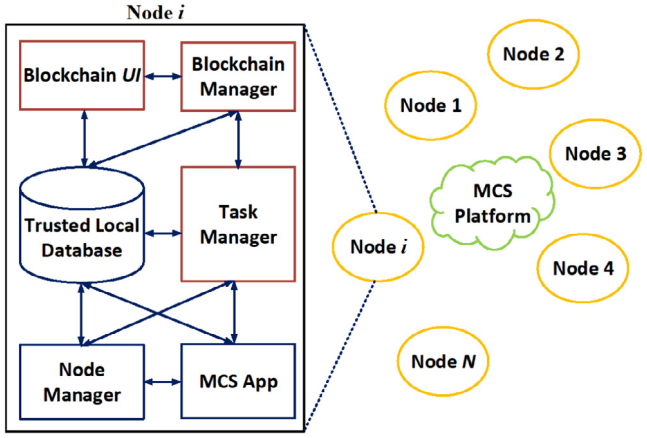

Fig. 2. MCS node structure.

Blockchain Manager is responsible for performing the tasks that should be done by a miner, e.g., block generation and verification, a personal node key pair generation, hashing data, checking data integrity, and signing/verifying signatures. Task Manager is applied to fulfill tasks being assigned and agreed. All information related to the above functional modules is stored at Trusted Local Database, e.g., latest blockchain if need to keep a copy locally, local data of MCS, public/private key pair, etc. Note that local credentials can be saved in a more secure place, other than Trusted Local Database.

3.2. Security model and research assumptions

3.2.1. Security model

Most current MCS systems are based on a centralized architecture, which consists of a trusted centralized service provider. As aforementioned, the centralized service provider is rational and may not always behave honestly, which leads to several security problems, such as false worker selection and selfish payment calculation, inaccurate data processing result, and privacy disclosure. Besides, it is also vulnerable to single point failure. MCS-Chain is a fully distributed system based on blockchain. There is no any centralized and trusted party to depend on. Instead, all nodes, including miners, workers, and users, are rational and profit-driven. They cannot be fully trusted. Moreover, the nodes in MCS-Chain do not trust with each other, and all the nodes behave rationally

and make decisions based on the information recorded in the blockchain, which is publicly proved and verified.

3.2.2. Research assumptions

We hold the following assumptions in the design of MCS-Chain with justification based on the above system model and security model, as well as previous work.

1. The system comprises no any centralized party responsible for identity and key management, trust management, or serving as a centralized MCS service provider.

2. We assume that each node can obtain a synchronized time stamp. This can be achieved with public GPS signals or based on a public time blockchain [48,49]. Besides, MCS nodes cannot forge time stamp. This assumption is reasonable since all messages transmitted through MCS are monitored by its neighbors and network infrastructures like base stations.

3. All nodes generate and store their public/private key pairs in a secure way. We also assume that the Trusted Local Database is well protected and safe, e.g., based on trusted computing technologies. Non-authorized parties cannot access it.

4. In case the limitation of local storage, some contents of the blockchain can be stored in another place, e.g., cloud with essential protection like encryption if needed. While a link and a key directing to the remotely stored contents are kept locally in a secure way.

5. Since communication security is not the focus of this paper, we assume that MCS nodes can communicate with each other through a secure communication channel.

3.3. MCS-Chain design goals

MCS-Chain aims to employ blockchain to build a decentralized and trustworthy MCS, where no entity is fully trusted. Nonetheless, as aforementioned, existing blockchain designs confront a series of problems in terms of security and availability. First, most existing blockchain designs face centralization problems. Second, it takes longer time to record a transaction in blockchain because of temporary forks in most blockchain designs, which also causes some security issues. Third, the computation overhead of some consensus mechanisms, like PoW, is extremely high, which leads to low efficiency. Fourth, long transaction confirmation time and high computation overhead limit the maximum of throughput of blockchain. Current solutions cannot solve all these problems. The main goal of MCS-Chain is to solve or improve the above problems in order to build a decentralized and trustworthy MCS system with high availability.

1. MCS-Chain aims at solving centralization problem in blockchain to improve its security. As a result, the blockchain cannot be controlled by a centralized party. Besides, MCS-Chain also tries to solve the temporary fork issue. To be specific, in each epoch, a unique block can be generated without waiting to check which branch is the longest.
2. To improve the availability of blockchain, MCS-Chain tries to avoid time-consuming mining happened in PoW and explores a more efficient consensus mechanism to achieve faster block generation. Besides, MCS-Chain also aims at reducing transaction confirmation time by solving temporary fork issue and reduce block generation time. Another goal of MCS-Chain is to improve maximum throughput of blockchain to support many MCS tasks.
3. MCS-Chain intends to provide accurate trust evaluation, which can offer MCS participants instruction on the trust levels of other participants, thus support trustworthy mobile crowdsourcing.

Table 1
Notations and descriptions.

Notations	Descriptions
N_i	The node i
PK_i, SK_i	The public and private key 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 time stamp of block k
B_k	The block k
B_{ID_k}	The ID of block k
$TV_{i,k,r}$	The trust value of node i in block k with role r, r can be end user (u), worker (w), or miner (m)
$TE_{i\to j,r,id}$	The feedback from node i on node j for its role r w.r.t. task id
CB_k	The content of B_k
TR_i	The task request of N_i
$STR_{s,i}$	The sub-task s of TR_i
$Des_{STR_{s,i}}$	The task description of $STR_{s,i}$
$Req_{STR_{s,i}}$	The task requirements of $STR_{s,i}$
ID_{TR_i}	The unique identifier of TR_i
$ID_{STR_{s,i}}$	The unique identifier of $STR_{s,i}$
$TB_{STR_{s,i,j}}$	The bidding on sub-task $STR_{s,i}$ from N_j
$TA_{STR_{s,i,j}}$	The assignment on sub-task $STR_{s,i}$ on N_j
$TC_{STR_{s,i,j}}$	The assignment confirmation on sub-task $STR_{s,i}$ from N_j
BP_{TR_i}	The bidding period of TR_i
$FD(r, id, PK_i)$	The feedback on node N_i regarding role r and task id
$KEY_{STR_{s,i}}$	The key that is applied to protect the result of $STR_{s,i}$
$ADR_{STR_{s,i}}$	The storage address that is applied to store the result of $STR_{s,i}$; it can be a storage node
$WPay_{STR_{s,i,j}}$	The payment confirmation message of $STR_{s,i}$ from N_i including actual payment for worker N_j
$ENC(m, DEK)$	The data encryption function that uses DEK to encrypt m
$ET_{STR_{s,i}}$	The execution time of $STR_{s,i}$
δ	The percentage of blockchain management service fee
$QUA_{STR_{s,i}}$	The quality requirement of $STR_{s,i}$
PAY	The threshold sum of payment that triggers new block generation
$TER_{STR_{s,i,j}}$	The task execution message containing the execution result of $STR_{s,i}$
$R(ID_{TR_i})$	The record of TR_i and its execution
$Pay_{STR_{s,i}}$	The payment budget of user N_i for subtask $STR_{s,i}$
$Pay'_{STR_{s,i,j}}$	The expected payment amount from worker j for subtask s generated by user N_i
$Pay''_{STR_{s,i,j}}$	The actual payment paid by user N_i to worker N_j for its contribution in subtask s
$D_{s,i,j}$	The execution result generated by worker N_{jh} for subtask $STR_{s,i}$

3.4. Notations and definitions

For easy presentation, Table 1 summarizes all the notations used in the MCS-Chain design. For some important notations, we give detailed definitions as below:

1. TR_i is the task request of N_i and is composed of several sub-tasks. Each sub-task consists of following parts: subtask id $ID_{STR_{s,i}}$, the description of the sub-task $Des_{STR_{s,i}}$, the requirements the workers should satisfy $Req_{STR_{s,i}}$, the payment budget $Pay_{STR_{s,i}}$, N_i's public key PK_i, and bidding period of TR_i. To be specific, $TR_i = \{ID_{STR_{s,i}}, Des_{STR_{s,i}}, Req_{STR_{s,i}}, Pay_{STR_{s,i}}, PK_i, BP_{TR_i}\}, s = 1, \ldots \ldots S$;
2. ID_{TR_i} is the identifier of the task, which is generated by calculating the hash code of the task request, namely $ID_{TR_i} = H(TR_i)$
3. $ID_{STR_{s,i}}$ is the identifier of the subtask, and is generated by calculating the hash code of the subtask description, subtask requirements, and its payment budget, namely, $ID_{STR_{s,i}} = H(Des_{STR_{s,i}}, Req_{STR_{s,i}}, Pay_{STR_{s,i}})$
4. $Req_{STR_{s,i}} = \left(Cert_{s,i,j}, KEY_{STR_{s,i}}, QUA_{STR_{s,i}}, ADR_{STR_{s,i}}, ET_{STR_{s,i}}\right)$ is the subtask requirement, which indicates the expected certificates of the workers for fulfilling the task, the encryption key used to protect task data, the quality requirement of

task execution and the location to upload/store the task execution result, as well execution time deadline.

5. $TB_{STR_{s,i,j}} = \left\{\left(ID_{STR_{s,i}}, Cert_{s,i,j}, Pay'_{STR_{s,i,j}}\right)\right\}$ is node N_j's bidding on subtask $ID_{STR_{s,i}}$, which includes required certificate $Cert_j$ and a bidding price $Pay'_{STR_{s,i,j}}$. The bidding is signed by N_j's private key as $SIG\left(H\left(TB_{STR_{s,i,j}}\right), SK_j\right)$;
6. $TA_{STR_{s,i,j}} = \left\{\left(ID_{STR_{s,i,j}}, Pay'_{STR_{s,i,j}}\right)\right\}, s = 1, \ldots, S$ is the task assignment result;
7. $TC_{STR_{s,i,j}} = SIG\left(H\left(TA_{STR_{s,i,j}}\right), SK_j\right), s = 1, \ldots \ldots, S$, is task confirmation generated by selected worker N_j;
8. $TER_{STR_{s,i,j}} = \{ID_{STR_{s,i}}, E\left(D_{s,i,j}, Key_{STR_{s,i}}\right), ADR_{STR_{s,i}}\}$ is the task execution reply for $STR_{s,i}$ from N_j to N_i, where $D_{s,i,j}$ is the task execution result of N_j for subtask $STK_{s,i}$.
9. $WPay_{STR_{s,i,j}} = SIG((Pay''_{STR_{s,i,j}}, PK_j), SK_i), SIG((\delta * Pay''_{STR_{s,i,j}}), SK_i)$ is the payment confirmation message including the actual payment of $STR_{s,i}$ from $N_i$to N_j considering trust values;
10. $$R\left(ID_{TR_i}\right) = \begin{cases} TR_i, SIG(H\left(TR_i\right), SK_i) \\ TA_{STR_{s,i,j}}, SIG(H\left(TA_{STR_{s,i,j}}\right), SK_i) \\ TC_{STR_{s,i,j}} \\ TER_{STR_{s,i,j}}, SIG(H(TER_{STR_{s,i,j}}), SK_j) \\ WPay_{STR_{s,i,j}} \\ TE_{x\to y,r,id}, SIG(H\left(TE_{x\to y,r,id}, ID_{STR_{s,i}}\right), SK_x) \end{cases},$$

where $TE_{x\to y,r,id}, SIG(H\left(TE_{x\to y,r,id}, ID_{STR_{s,i}}\right), SK_x)$ is node x's feedback on node y with regard to role r played by y and task $ID_{STR_{s,i}}$. $R(ID_{TR_i})$ is a record in the blockchain, and each record is attached with a signature by its issuer. The complete content of $R(ID_{TR_i})$ can be saved off the blockchain, while the hash value of $R(ID_{TR_i})$ can be saved in the blockchain.

4. MCS-Chain design

This section provides the details of MCS-Design. We first overview the whole system design and introduce the structure of MCS-Chain block. Then, we describe the task management procedure in the MCS-Chain and design its trust evaluation algorithm. Finally, we specify the consensus mechanism and incentive mechanism of MCS-Chain.

4.1. System overview

We establish a fully distributed MCS architecture based on the proposed MCS-Chain. MCS-Chain achieves security, trust, and efficiency with economic computing resource consumption. It enables miners to record messages and to generate blocks efficiently, which reduces processing latency and improves system throughput. We design a novel consensus mechanism for MCS-Chain, which avoids time-consuming computation to generate a new block. In MCS-Chain, a miner creates a new block when the accumulated payment amount waiting to be recorded in the next block exceeds a pre-defined threshold. Therefore, it greatly reduces the computation overhead. To avoid forks that occur in most current blockchain systems, we design a block selection algorithm that enables the miners to uniquely determine which block should be chosen and confirmed when they receive multiple newly announced blocks. Based on the proposed MCS-Chain, we build a fully distributed MCS system.

The blockchain applied in the MCS-Chain records all the task related information as well as trust information of nodes. The data stored by the miners can be categorized into two types: online data and offline data. Online data are stored in the blockchain. When generating a new block, the content of the data is recorded in the blockchain. Differently, for offline data, the miners only record the hash codes of them in the blockchain. In this way, the miners can remove the outdated data in time to reduce memory space occupation. Once data are recorded in the blockchain, it means that most miners have verified the validity of the data. Once a block is accepted by most of miners and satisfied with the consensus mechanism, we call the block is validated and the information recorded in this block is activated as well. Similarly, if a task execution result is recorded in the blockchain, this means that the correctness and validity of the execution result has been verified and confirmed. The information in the blockchain instructs its maintenance and the execution of MCS tasks. For example, the end user can refer to the trust information of nodes in the blockchain to select trustworthy workers.

MCS-Chain mainly implements four functions, namely, blockchain management, task management, trust evaluation, and incentive mechanism. Among them, the blockchain of MCS-Chain acts as a distributed database recording key information regarding task execution and management, trust evaluation and management, and incentive mechanism. The task management includes task request, bid collection, task assignment and confirmation, and task execution (including data collection, data storage, and data processing). Task management stores collected data and processing results in a distributed way by recruiting storage nodes, and keeps the digest of task execution in the blockchain. Besides, all the data about task request, task bid, task assignment, and task confirmation are stored in the blockchain as offline data. The trust of miners, workers, and end users are evaluated by the miners. The results of evaluation are stored in the blockchain. The incentive mechanism includes the incentive to miners and the incentive to workers. It decides the amount of payments that the miners and the workers can obtain based on their behaviors and performances in task fulfillment and their trust. As a result, it motivates mobile users to act as either workers or miners and encourages their honest behaviors.

4.2. Block structure

The structure of block k is designed and shown in Fig. 3. It contains its previous block's ID, $B_{ID_{k-1}}$, which is the hash value of the content CB_{k-1} of block $k-1$, i.e., $B_{ID_k} = H(CB_{k-1})$; the time stamp of block k's generation T_k; the ID of block creator ID_{N_x}; a series of records that record a number of MCS tasks' execution. Each task record contains the data about task request, task assignment, task assignment confirmation, task execution results, and task payment (the payments to both the miners and the workers), as well as the feedback related to the task. What is also recorded in the block is a trust value list that records the trust value of all nodes in different roles with newly updates in this block. The part of feedback can be empty for some nodes in terms of some roles in case that there are no MCS interactions happening with the roles after the generation of previous block and before the new block is generated.

4.3. Task management

MCS-Chain has no centralized service provider responsible for worker selection, data collection, and trust evaluation. Instead, task management is performed in a fully distributed way with the assistance of the blockchain. First, an end user sends a task request with signature to a miner. The miner that receives the request will verify the validity of signature and then broadcast it in the network of MCS-Chain. In MCS-Chain, we call a message is validated if and only if it has been recorded in a created block and the block is accepted by the miners. If the task request is validated, all the workers can submit bidding messages to provide

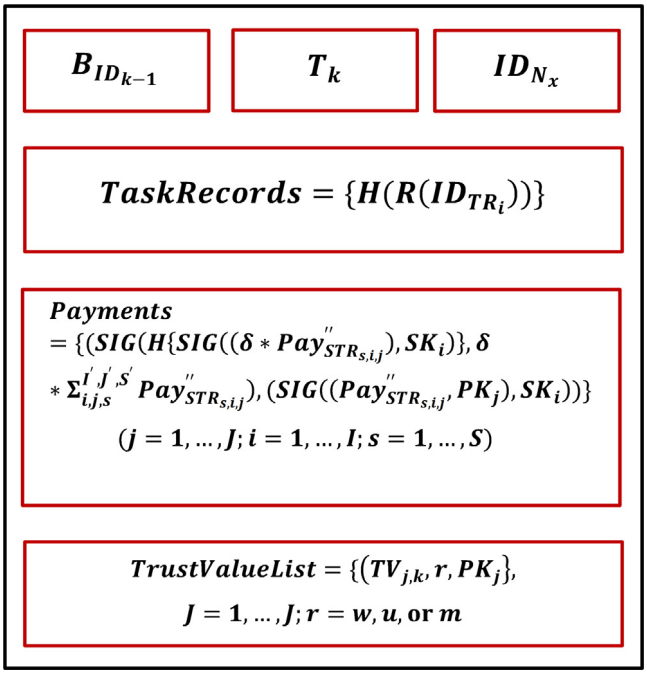

Fig. 3. Block structure.

their attributes and expected payment within an expiry period to a miner in order to apply for the task. With several rounds of bargain, the end user and the worker reach an agreement on the payment, and the decision will be recorded in the blockchain. After the agreement is validated, the workers begin to perform the confirmed tasks. Typically, sensing workers sense data, upload data to the indicated storage workers. The data should be protected with encryption. Our scheme supports various encryption algorithms, such as symmetric encryption, asymmetric encryption, attribute-based encryption (ABE), homomorphic encryption, homomorphic re-encryption, etc., based on the requirements of the end users. Computing workers can access the data and perform requested operations. Finally, the computing workers transfer processing results to the end users in a secure way. After the task is completed, all the participants involved in the task generate feedback on others based on their behaviors. The miners further aggregate the feedback and evaluate the trust of them. The miners also decide the final payment to themselves and the workers in this step. Concretely, the detailed procedure is illustrated in Fig. 4 and presented as below.

Step 1. End user i announces its task request TR_i by sending TR_i with $SIG(H(TR_i), SK_i)$ and PK_i in the MCS network. The announced task contains a number of subtasks $STR_{s,i}$ $(s = 1, \ldots, S)$ with detailed subtask description $Des_{STR_{s,i}}$; the subtask requirement $Req_{STR_{s,i}}$ that indicates the expected certificates of the workers for fulfilling the task, the encryption key used to protect task data $KEY_{STR_{s,i}}$ and the location to upload/store the task execution result, quality requirements $QUA_{STR_{s,i}}$, subtask execution time deadline $ET_{STR_{s,i}}$, and the minimum payment for subtask fulfillment $Pay_{STR_{s,i}}$. Note that the certificates can be stored in some existing blockchains and verified in a transparent and open measure. The miners verify the validity of task request and announce validation result in the MCS network.

Step 2: Workers bid the task or the subtasks $TB_{STR_{s,i,j}} = \{ID_{STR_{s,i}}, Cert_{s,i,j}, Pay'_{STR_{s,i,j}}\}$ by providing required certificate $Cert_{s,i,j}$ and indicating a bidding price $Pay'_{STR_{s,i,j}}$. The decision of worker selection is made by the end user by considering the bidding price and worker trust.

Step 3: End user i verifies $SIG(H(TB_{STR_{s,i,j}}), SK_j)$ and checks $Cert_{s,i,j}$. It decides task or subtask workers based on trust values of workers $TV_{i,k,r}$ in the role of worker and their bidding price $Pay'_{STR_{s,i,j}}$.

Step 4: End user i announces its task assignment $TA_{STR_{s,i,j}}$ with $SIG(H(TA_{STR_{s,i,j}}), SK_i)$. The miners verify the validity of task assignment and announce validation result in the MCS network.

Step 5: The workers provide their confirmation on the task assignment $TC_{STR_{s,i,j}}$. Note that Step 2–5 could perform several rounds in order to reach a task contract between the end user and the workers. Negotiation could be conducted before real task assignment and confirmation. The miners verify the validity of task assignment confirmation and announce validation result in the MCS network.

Step 6: The workers announce task fulfillment before execution deadline by announcing $TER_{STR_{s,i,j}}$ with $SIG(H(TER_{STR_{s,i,j}}), SK_j)$. The task execution result is stored in the indicated address $ADR_{STR_{s,i}}$ and protected by indicated key$KEY_{STR_{s,i}}$.

Step 7: End user i checks the quality of the work, if the quality can satisfy $QUA_{STR_{s,i}}$, it announces $WPay_{STR_{s,i,j}}$, which includes the actual payment to node j, and also the service fee for miners that is an specified percentage of $Pay''_{STR_{s,i,j}}$. If the quality cannot satisfy $QUA_{STR_{s,i}}$, node i may reject the payment or request task re-do. The quality can be proved by the miner group that judges the correctness of execution behaviors. The miners verify the validity of task fulfillment and announce validation result in the MCS network.

Step 8: Feedback is announced by the end user, the miners and the workers $TE_{x\to y,r,id}$ with regard to different roles in the fulfillment of TR_i or its subtasks $STR_{s,i,j}$.

Step 9: The miners collect feedback within an expected and specified period. When the feedback collection period is expired, they perform trust evaluation and block generation in order to confirm task execution and payment.

Note that all related parties in the system (end users, miners and workers) can feedback with each other after the task is executed during the feedback period. The feedback is used to evaluate the trust of different nodes with different roles in order to decide the priority of block generation.

4.4. Trust evaluation

This subsection presents the way of trust evaluation in MCS-Chain. In order to overcome attacks in trust evaluation (e.g., unfair rating attack and bad/good mouthing attack), we apply deviation between personal feedback and average feedback as well as past trust value to tailor the contribution of individual feedback $TE_{i\to j,r,id}$ to the trust value calculation for generating a new block. The trust evaluation on $N_j(j = 1, \ldots, J)$ is performed by the miners during the process to create a new block based on the following formula:

$$TV_{j,r,k} = \frac{1}{\left(e^{-\frac{|k-k_i|}{\tau}} + 1\right) * O} \Sigma^{I,S}_{i,S} w_{i\to j,r,id} * TE_{i\to j,r,id} * \left(1 - dv_{i,j,r}\right) + \frac{e^{-\frac{|k-k_i|}{\tau}}}{e^{-\frac{|k-k_i|}{\tau}} + 1} TV_{j,r,k_j} * e^{-\frac{|k-k_i|}{\tau}},$$

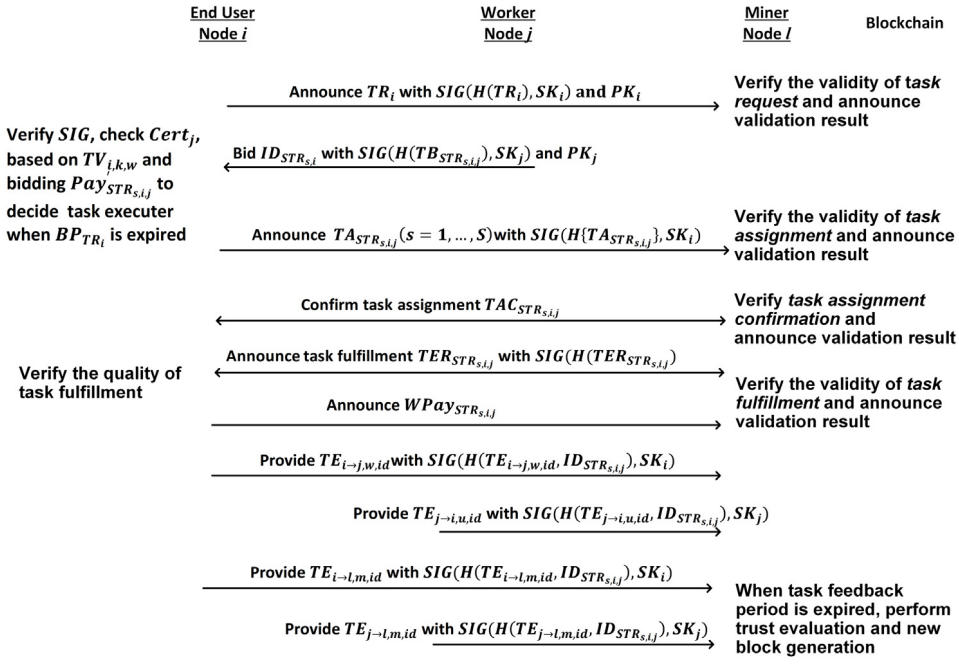

Fig. 4. MCS-Chain task management procedures.

where $dv_{i,j,r} = |TE_{i\to j,r,id} - \frac{1}{\Sigma_{i,S}^{I,S} w_{i\to j,r,id}} \Sigma_{i,S}^{I,S} w_{i\to j,r,id} * TE_{i\to j,r,id}|$ is the trust evidence deviation, $O = \Sigma_{i,S}^{I,S} w_{i\to j,r,id} \left(1 - dv_{i,j,r}\right)$, $w_{i\to j,r,id} = Pay''_{STR_{id,i,j}}$ is the weight calculated based on the volume of subtask payment. Obviously, the weight of a feedback is related to the payment of the feedback generator and the accuracy of the feedback. The main reason for it is that we assume all MCS nodes are rational. In this way, MCS nodes are motivated to generate feedback honestly, which will be analyzed in detail in the following part. In the evaluation, we also consider the effect of the previous value of trust. Parameter τ is applied to control time decaying that makes later trust value to contribute more in the evaluation. k_i is the block number of the latest TV_{j,rk_i} appeared in the blockchain. We use $(1 - dv_{i,j,r})$ to tailor $TE_{i\to j,r,id}$ in order to overcome the negative influence on the trust evaluation raised by bad mouthing attack or malicious/distrusted evidence providing nodes [50,51]. In the initial block, since there is no previous information about any nodes, all node trust values are set to zero, and the task record area is empty. We assume that the trust value is a real number in the scope of [0, 1], where 0 represents fully distrusted and 1 stands for fully trusted.

4.5. Consensus mechanism

For solving the fork problem in the existing consensus mechanisms, e.g., PoW, PoS and DPoS, we propose a novel consensus mechanism in MCS-Chain. PoW is the underlying consensus algorithm of Bitcoin. Although it can achieve eventual consistency, it suffers from high resource consumption, high latency, and low throughput. Presently, a number of improved schemes are developed, like PoS and DPoS. Although these algorithms well overcome the drawbacks of Bitcoin, they also face the fork problem, the same as PoW. When fork happens, it takes time to judge which one is the longest chain. In order to overcome this issue, we propose a novel consensus algorithm, where a new block is generated when the accumulated payment amount waiting to be recorded in the next block exceeds a pre-defined threshold. When finding a new block, the miner conduct trust evaluation and calculate the payment to the former block generator in order to fully make use of the computation resources of the miners. Algorithm 1 illustrates the details of block generation algorithm.

Note that in $SIG(H\left\{SIG\left(\left(\delta * Pay''_{STR_{s,i,j}}\right), SK_i\right)\right\}, SK_x)$, SK_x is block k's creator's private key. $\delta * \Sigma_{i,j,s}^{I',J',S'} Pay''_{STR_{s,i,j}}$ is the total service award that should be paid to the creator of block $k-1$. $\Sigma_{i,j,s}^{I',J',S'} Pay''_{STR_{s,i,j}}$ is the total value of payment of all the tasks that should be recorded in the new block. In case that multiple miners work out the new block at the same time, we apply Algorithm 2 to select the winner in order to avoid blockchain forks. First, the node that generates the block at the earliest time wins. Applying this rule aims to ensure the efficiency of blockchain generation. But if a node holds too many awards, we give priority to another miner in order to ensure decentralization and avoid such a situation that the blockchain is controlled by few nodes. Second, in case that two nodes generate the block at the same time, we give the node with higher trust a higher priority, since the node holding a higher trust value has more incentive to behave honestly for block generation. But if the reputable node holds too many awards, we give priority to another node in order to ensure decentralization and also avoid such a situation that the blockchain is controlled by few nodes. Third, in case that two nodes generate the blocks at the same time and have the same trust values, we give a higher priority to the node with less awards. This rule aims to avoid the situation that the blocks are generated by a small number of miners in order to ensure decentralization. In case tie happens again at this moment, we let the miner with a bigger (or smaller) public key wins. In case that multiple miners generate the new block at the same time, we select a winner by following the similar rules as above. To guarantee the next block can be decided within a limited time, the miner will wait for a limited period for more blocks when it receives a block of the next epoch for the first time and if the block is verified as valid. The length of the period, represented as θ, is predefined and can be adjusted according to the variation of the MCS network. Any blocks that are received after the expiry of the period will be rejected by the miner. In order to ensure decentralization, one miner should not always win, we limit the total number of wins for an individual miner in a specific time period based on the total number of miners (i.e., M), e.g., the total

Algorithm 1: Block Generation

Input: A set of $\{R(ID_{TR_i})\}$ waiting for inserting into the blockchain of MCS-Chain; B_l $(l = 0, \ldots\ldots, k-1)$ is the previous blocks.

Output: Block B_k, its creator's public key PK_x, and the creator's signature on CB_k.

When a number of tasks have completely executed and their feedback collection periods are expired, and $\Sigma\Sigma_{s=1}^{S} Pay''_{STR_{s,i,j}} \geq PAY$,

Do

Verify the correctness of all related signatures as valid in task execution procedure;

Verify the correctness of all payments by checking with the latest trust values based on roles that are used for payment calculation;

Verify the payment to the creator of block *k-1*;

For $j = 1, \ldots, J; i = 1, \ldots, I$ **Do**

For $r = w, u, or\ m$, respectively **Do**

Calculate $dv_{i,j,r}$ based on $TE_{i\to j,r,id}$ and $Pay''_{STR_{id,i,j}}$;

Calculate $TV_{j,r,k}$

Package B_k based on block structure by calculating $B_{ID_{k-1}}$, and inserting the payment into the block and issue awards to block $k-1$ creator. The award $\delta * \Sigma_{i,j,s}^{I',J',S'} Pa''_{STR_{s,i,j}}$ and the signature $SIG(H\{SIG((\delta * Pay''_{STR_{s,i,j}})\}, SK_i), SK_x)$ are put as the first part of payments.

(Note that T_k is the signing time of B_k's creator.)

number of wins can be set as $\frac{c*M}{M-1}$, where c is a parameter to control winning times. Algorithm 2 ensures that only one winner of a new block generation can be found, thus no blockchain fork could happen. In MCS-Chain, any mistake on block creation can be found and solved.

4.6. Incentive mechanism

Incentive mechanism decides the payment to nodes to encourage them to behave honestly and positively. The incentive mechanism of MCS-Chain is composed of the incentive mechanism to miners and the incentive mechanism to workers. To be specific, it determines the amount of payment the miners and the workers can obtain through participation in MCS tasks and based on their trust. The node trust is directly decided by node feedback regarding the execution of MCS tasks. The trust evaluation algorithm can effectively resist negative impact caused by inaccurate feedback, which is proved in our previous studies [50,51]. Thus, the node trust can well reveal the behavior of nodes. Based on the trust, we deign the incentive mechanism of MCS-Chain. As a result, it is able to motivate both miners and workers to behave honestly.

4.6.1. Incentive mechanism to workers

Payment is the direct motivation that encourages workers to accept a task. Obviously, it is effective to associate the payment of nodes with their trust in order to incent them to behave honestly. Based on this idea, we design the incentive mechanism to workers. To be specific, suppose $Pay''_{STR_{s,i,j}}$ denotes the actual payment paid to worker N_j for its contribution in subtask s, and $Pay''_{STR_{s,i,j}}$ can be calculated with the following formula:

$$Pay''_{STR_{s,i,j}} = \frac{(Pay_{STR_{s,i}} + Pay'_{STR_{s,i,j}})}{2} * (1 - \epsilon\left(TV_{i,u,k_i} - TV_{j,w,k_j}\right))$$

where parameter ϵ is a payment control index to adjust the payment scale impacted by the trust values of the end user and the worker. The higher trust value the worker has, the more payment the worker gains; the lower trust value the user has, the more payment the worker gains. With this way, both the user and the worker are encouraged to behave honestly in order to increase their income and reduce their expense, respectively.

4.6.2. Incentive mechanism to miners

Just like the incentive mechanism in Bitcoin, our incentive mechanism offers payment to miners that successfully create a new block. Nevertheless, in our scheme, the reward for block creation is not cryptocurrency. Instead, MCS-Chain requests the end user to include the payment to miners in $Pay_{STR_{s,i,j}}$. The miners calculate the sum of $\delta * Pay''_{STR_{s,i,j}}$ by aggregating all the $SIG((\delta * Pay''_{STR_{s,i,j}}), SK_i)$ in $WPay_{STR_{s,i,j}}$ message as the reward to the previous block creator. We should note that δ can be dynamically adjusted based on the trust of the block miner, that is $\delta = \delta_0 * \left(1 + TV_{x,m,k_x} - \frac{1}{X}\Sigma_{x=1}^{X} TV_{x,m,k_x}\right)$, where the trust value of miner x below the average of miner trust value will cause lower service award and above the average of miner trust value will help gaining higher award than the offer. In order to motivate the miners to work honestly and efficiently, we give a more reputable miner that works out the new block a higher award. We also give the miner with higher miner trust value the higher priority for block generation, refer to Algorithm 2.

For encouraging the collaboration of the miners, the awards can be shared among the miners who contributed to the verification of each step of the task execution. Information announcement is valid if more than half of the miners acknowledge that the announcement is received and verified as valid. Some part of the award can be shared based on the number of announcement acknowledgments provided by the miners. Thus, even though some miners cannot be the winner of block creation, they can still benefit from the participation of the work of MCS.

5. Security analysis and performance evaluation

In this section, we prove the security of MCS-Chain and conduct a series of experiments to evaluate its performance. For the security analysis, we mainly prove four security properties of MCS-Chain, namely, liveness, safety, fault tolerance, and decentralization assurance [52,53]. For the performance evaluation, we conduct a number of experiments to test such quality attributes as block generation frequency, latency of transaction response, throughput, computational overhead, and trust evaluation accuracy, based on concrete MCS-Chain implementation in both Windows and Android platforms.

Algorithm 2: Block Miner Selection

Input: N_i and N_j, which announced creating a new block with timestamp T_{N_i} and T_{N_j} and both generated blocks correctly;

Output: the winner of block generation

Do

If $T_{N_i} \neq T_{N_j}$,

Do

Output the node that creates the valid next block at earlier time as the winner (except that this node has gained more than a threshold times of awards, in this case, we select the winner as the second earliest miner);

If $T_{N_i} = T_{N_j}$

If $TV_{i,m} \neq TV_{j,m}$,

Do

Output the node with a latest higher miner trust value as the winner (except that this node has gained more than a threshold times of awards);

Else If $TV_{i,m} = TV_{j,m}$

Output the node with less number of awards;

In case tie again in the above process, the node with a bigger or smaller public key wins.

5.1. Security analysis

In this subsection, we analyzed the security of the proposed system with four properties, i.e., liveness, safety, fault tolerance, and decentralization. In a blockchain system, liveness means that a new block can be always generated. Safety means that there will not exist a permanent fork among honest miners if they obey an underlying protocol. Liveness and safety are two basic security properties of a blockchain system. Among them, liveness guarantees the availability of blockchain based services. Safety ensures that the blockchain is trustful. That is, there will not exist more than one block that are accepted by honest miners at the same time. As a result, inconsistency among honest miners will not happen. Another important security property of a blockchain system is fault tolerance. The blockchain system works in a trustless environment. There surely exist dishonest miners that attempt to harm the security of the blockchain. Fault tolerance indicates how many dishonest miners can be resisted by the system. The last property of security is decentralization. As mentioned above, centralization phenomenon occurs in several existing blockchain based applications, like Bitcoin and Ethereum. Therefore, although blockchain claims to provide a perfect decentralized platform, it still confronts serious centralization problem. In our scheme, we provide an extra mechanism to ensure decentralization even there exist a powerful miner. To demonstrate its effectiveness, we prove the decentralization of our scheme in theory.

5.1.1. Model of analysis

We analyze the security of MCS-Chain based on the following model. We assume the presence of an adversary that controls a proportion $\mu \in (0, 1)$ of the whole set of miners, and thus a proportion of $1 - \mu$ behave honestly. Suppose all the miners have equal data collection abilities. Therefore, the number of miners controlled by the adversary determines its probability for block admission. We also take into consideration the impact of network model on the consistency of blockchain. We adopt the network model in [54], where the time difference between the time of block announcement and the time when a node receives the announcement follows an exponential distribution. To be specific, we denote the time difference as $t_{i,j}$, where i is the block creator and j is the miner receiving the block creation announcement. Then $t_{i,j}$ follows the exponential distribution as below.

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

where λ is the average time latency for a miner receiving the block announcement. In Bitcoin, the average time latency is 12.6 s, namely, $\lambda = 12.6$ s. In our analysis, we assume the average time required to find a new block is γ. In a certain time epoch, if an honest miner receives a block with time stamp TS_i for the next time epoch, it should first verify the validity of the block. If the block passes the verification, the miner will stop mining and continue to listen to receive more blocks for the next epoch until it meets the expiry time $TS_i+\theta$, where θ is the waiting period of receiving newly generated blocks.

Similar to other blockchain systems, time is divided into a series of epochs. The creation of the next block indicates the end of the previous epoch and the start of the next epoch. As a result, each epoch corresponds to a block. Suppose $\{B_k\}_{k=0,1,2,\ldots\ldots,N_{Block}}$ is the created blocks that have been admitted by all the miners, where N_{Block} is the number of blocks that have achieved consistency. Let $B_{N_{Block}+1,i}$be the next block created by miner M_i and $M_i \in M$, where M is the set of miners. Suppose TS_i is the time stamp that indicates when the block $B_{N_{Block}+1,i}$ is created.

5.1.2. Liveness

Definition 1 (*Liveness*)**.** For any epoch after the blockchain started, liveness means that at least a new block will be created after a sufficiently long but bounded period of time.

Theorem 1. *Suppose* $\{B_k\}_{k=0,1,2,\ldots\ldots N_{Block}}$ *is the created blocks that have been accepted by the miners, at least one block* $B_{N_{BLock}+1}$ *will be generated by a miner.*

Proof. In our design, a miner generates a new block when it records enough payment confirmation messages, of which the accumulated payment amount reaches the predefined threshold. Therefore, as long as MCS-Chain is operating, there will always be a new task request reaching miners. Suppose there are at least n payment confirmation messages per unit time, and the average payment amount for each transaction is α, and the threshold for block generation is β, then the waiting time for generating the next block should be at most $\frac{\beta}{n\alpha}$. Therefore, miners can always generate a new block as long as the system is operating.

5.1.3. Safety

Definition 2 (*Safety*)**.** Safety means that there exist no forks even multiple blocks are created by different miners.

Theorem 2. *Suppose* $\{B_k\}_{k=0,1,2,\ldots\ldots N_{Block}}$ *is the created blocks that have been accepted by the miners, where* N_{Block} *is the number of blocks that have achieved consistency, then after a sufficiently long but bounded period of time, all honest miners will accept a common block* $B_{N_{Block}+1}$ *as the next block of current blockchain.*

Proof. The proof of safety is based on the security model where no entity in MCS-Chain is fully trusted and all entities are rational and profit driven. First, we prove that in our design, the probability for a miner to receive a valid block with the earliest timestamp is at least $1 - e^{-\frac{\theta}{\lambda}}$. Then we show that within a certain θ, this block can be received and accepted by all the miners with a high probability. Finally, we can conclude that a unique block can be selected as the next block.

Based on Theorem 1, if the block set of current chain is $\{B_k\}_{k=0,1,2,\ldots\ldots N_{Block}}$, at least one $B_{N_{BLock}+1}$ block will be generated by a miner after a sufficiently long but bounded period of time. Here, we represent the upper bound of the time as γ, then within γ, there is at least one block generated by a miner. Based on the network model, once a block is created and broadcast, it can arrive in other nodes within limited time. Therefore, suppose a honest miner M_j receives a block $B_{N_{Block+1},i_0}$ and as aforementioned, it will create a block set $\{B_{N_{Block+1},i}\}$, insert the received block into the set, and continue to listen to the network for more blocks before the expiry time $TS_{i_0} + \theta$. Once it meets the expiry time, M_j stops listening and invokes Algorithm 2 to select a unique block represented by $B^j_{N_{Block}+1}$. Based on the Algorithm 2, we could safely draw the conclusion that $B^j_{N_{Block}+1}$ has the earliest time stamp (with selection priority if its generator does not win over a threshold times) in set $\{B_{N_{Block+1},i}\}$.

At this point, all the honest miners in the blockchain network has their own $B^j_{N_{Block}+1}$. Suppose these blocks form the block set $\{B^j_{N_{Block}+1}\}$, and suppose $B_{N_{Block}+1}$ is the output of Algorithm 2 taking $\{B^j_{N_{Block}+1}\}$ as input. Therefore, $B_{N_{Block}+1}$ has the earliest time stamp, which stands for that it is the earliest block for epoch $N_{Block} + 1$. Considering that $B_{N_{Block}+1}$ is created at $TS_{N_{Block}+1}$, and that the time difference between the time when the block is created and the time a miner receives the block follows exponential distribution, if a miner M_j receives the block at time t_j, then t_j should follows the following distribution:

$$P\left(t_j = t\right) = \frac{1}{\lambda} e^{-\frac{1}{\lambda}(t - TS_{N_{Block}+1})}, \quad (t_j > TS_{N_{Block}+1})$$

Cumulative distribution function of t_j is that:

$$P\left(t_j < t\right) = 1 - e^{-\frac{1}{\lambda}\left(t - TS_{N_{Block}+1}\right)}, \quad (t_j > TS_{N_{Block}+1})$$

Therefore, the probability of the block $B_{N_{Block}+1}$ is received by a miner before the expiry time $T_{N_{Block}+1} + \theta$ is:

$$P\left(t_j < TS_{N_{Block}+1} + \theta\right) = 1 - e^{-\frac{\theta}{\lambda}}, \quad (t_j > TS_{N_{Block}+1})$$

Since $B_{N_{Block}+1}$ has the earliest time stamp, the miners that first received $B_{N_{Block}+1}$ will ends listening period ahead of other miners. Hence for any miner M_j, its expiry time $TS^j_{N_{Block}+1} + \theta$ satisfies $TS^j_{N_{Block}+1} + \theta \geq TS_{N_{Block}+1} + \theta$, and its probability of receiving $B_{N_{Block}+1}$ before its expiry time satisfies:

$$P\left(t_j < TS^j_{N_{Block}+1} + \theta\right) > P\left(t_j < TS_{N_{Block}+1} + \theta\right) = 1 - e^{-\frac{\theta}{\lambda}}$$

Therefore, for any miner in the network, its probability of receiving $B_{N_{Block}+1}$ is at least $1 - e^{-\frac{\theta}{\lambda}}$. Therefore, if the miner obeys the predefined protocol, when expiry time ends, at least $1 - e^{-\frac{\theta}{\lambda}}$ percent of miners have received the block. θ is a threshold value defined ahead of time, and can be adjusted based on a network status. The aforementioned probability varies with different values of θ. We calculate the probability with different values of θ as below, from which we can see that more miners can receive the block with a longer θ. A proportion 99.995% of all the miners can receive $B_{N_{Block}+1}$ when we set the value θ as 10λ. Therefore, considering the time consumed for block creation, the block with earliest timestamp can arrive at more than 99.995% of all the miners within $\gamma + \theta$ with a high probability when $\theta > 10\lambda$.

When $\theta = \lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-1} = 0.6321$;

When $\theta = 2\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-2} = 0.8647$;

When $\theta = 3\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-3} = 0.9502$;

When $\theta = 4\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-4} = 0.9816$;

When $\theta = 5\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-5} = 0.9933$;

When $\theta = 6\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-6} = 0.9975$;

When $\theta = 7\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-7} = 0.9991$;

When $\theta = 8\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-8} = 0.9996$;

When $\theta = 9\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-9} = 0.9998$;

When $\theta = 10\lambda$, $P\left(t_j < T_{N_{Block}+1} + \theta\right) = 1 - e^{-10} = 0.99995$.

When there exist multiple blocks with the earliest timestamp, we can leverage Algorithm 2 to decide which block to select. Since each miner has its unique public key, even the block generators have the same timestamp, trust value, and number of blocks generated, a unique block can be decided by comparing their public keys and then the system reaches consistency.

5.1.4. Fault tolerance

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

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

Proof. We conduct our analysis with the assumption that all malicious miners are rational and attempt to gain the biggest benefits. Therefore, if a malicious miner finds a block ahead of others, the best choice is to publish it at once because with a proper θ, the block generated by the malicious miner will be accepted as the next block with a high probability.

If the malicious miner finds a block later than an honest miner, then it will lose the game based on Theorem 2 even it publishes the block at once, and all honest miners will reject this block. Nevertheless, since the adversary controls a proportion μ of the whole network miners, it can ask all malicious miners to work on the block it found. However, honest miners will not turn to work on the block created by the malicious miners since it has a later creation timestamp. Besides, since all the blocks contain the hash values of their previous blocks, all the blocks created by the malicious miners will be forbidden by all honest miners because they do not share a common previous block in the blockchain.

When a miner cannot figure out whether a block is generated by an honest miner or a malicious miner (e.g., the miner just joins in MCS-Chain), it can simply request the latest blocks from other miners, and choose a unique block from all received blocks with Algorithm 2. In this case, if there exists more than one honest

miner among these miners, a miner is able to receive the block accepted by honest miners. Therefore, the malicious miners succeed in cheating the miner to choose the invalid block only when all the n miners are dishonest. Suppose the malicious miners occupy a proportion μ of the whole nodes and the number of miners being requested is n. Then, the probability for malicious miners successfully cheat the honest miner is $P_{succ} = \mu^n$. In practice, we hope the probability to be as small as possible. For simplicity, if we hope P_{succ} is smaller than σ, then μ satisfies:

$$\mu < \sqrt[n]{\sigma}$$

Apparently, a large n can reduce the probability P_{succ}. Nonetheless, it will also raise higher communication cost and consensus delay. Therefore, we balance the efficiency and security, and set $n = 10$ and $\sigma = 0.00001$, then we calculate that $\mu < \sqrt[10]{0.00001} \approx 32\%$. Therefore, when we set n as 10, our scheme can resist misbehaviors of 32% of the whole miners. When n is 20, our scheme can resist misbehaviors of 56%.

5.1.5. Decentralization

Definition 4 (*Decentralization*)**.** Decentralization refers to that even a miner has much more powerful ability than other miners, the miner cannot control the blockchain.

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 blockchain.*

Proof. We employ state machine to prove the decentralization. In particular, we first model MCS-Chain consisting of a powerful miner as a state machine and define system states. Then, we get the state transition probability of the system and further obtain the probability of each state that the system reaches. We calculate the probability that the powerful miner successfully creates a new block. Finally, based on the calculated probability, we demonstrate that the probability can be reduced to a low level in MCS-Chain and prove its decentralization.

In our design, when a miner generates a new block, if the miner has successfully created more than n_{thr} blocks within the latest n_b blocks, all honest miners will consider the block as invalid and reject it directly. Obviously, n_b should be larger thann_{thr}. We assume there is a miner that is more powerful than other miners and is able to generate new blocks more efficiently. For simplicity, we assume this miner has a probability of ρ to create a new block ahead of other miners. We then prove that with a certain selection of n_b and n_{thr}, our scheme can effectively guarantee decentralization.

Herein, we consider a simple case, where $n_{thr} = 1$. In this case, once a miner generates a block and the block is accepted by other miners, any block generated by this miner will not be accepted in the following n_b epochs. In the $n_b + 1$ epoch, the miner competes with other miners. Fig. 5 depicts the process of the system as a state machine. We define the system state with a 0–1 sequence of length n_b. If the ith element of the sequence is 1, it means that the ith block of the previous n_b epochs is generated by the powerful miner, while 0 represents the opposite. For any n_b continuous epochs, the powerful miner can create at most one block. Therefore, the number of possible system states is $n_b + 1$. Suppose that current epoch is the kth epoch, we define the system state set $S = \{s_0, s_1, \ldots, s_{n_b}\}$, where s_0 represents the state that none of the previous n_b blocks is generated by the powerful miner and all of s_0's elements are 0; $s_i (0 < i \leq n_b)$ represents the ith block in the previous n_b block is generated by the powerful miner, and only the ith element is 1.

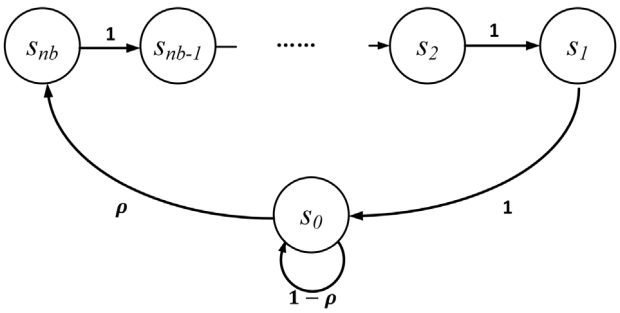

Fig. 5. State machine with transition frequencies.

$P(s_i)$ denotes the probability that MCS-chain reaches state s_i. From Fig. 5, we can obtain the following equations:

$$\begin{aligned} P(s_0) &= P(s_0) \cdot (1-\rho) + P(s_1) \cdot 1 \\ P\left(s_{n_b}\right) &= P(s_0) \cdot \rho \\ P(s_i) &= P(s_{i+1}) \text{ for } 0 < i < n_b \\ \Sigma_{i=0}^{n_b} P(s_i) &= 1 \end{aligned}$$

Through simple algebra, we can obtain the probability of each state as below:

$$P(s_0) = \frac{1}{1+\rho n_b}$$

$$P(s_i) = \frac{\rho}{1+\rho n_b} \quad (0 < i < n_b)$$

Since the power miner creates a block if and only if the system state transfers from s_0 to s_{n_b}. Therefore, the probability of the miner to generate a new block is $P(s_0) \cdot \rho = \frac{\rho}{1+\rho n_b}$.

In the worst case, ρ is equal to 1, which means that the powerful miner can always generate the next block ahead of others. With the proposed design, the probability that the powerful miner generates a new block is $\frac{1}{1+n_b}$, and a large enough n_b can effectively reduce the probability. Therefore, in practice, we can balance the availability and decentralization and select a proper value of n_b to guarantee the decentralization of MCS-Chain.

5.2. Performance evaluation

In this subsection, we first set up an evaluation metrics and then go ahead to test MCS-Chain performance.

5.2.1. Evaluation metrics

In our experiments, we tested the following five metrics of MCS-Chain, i.e., block generation time, latency, throughput, computational overhead, and accuracy of trust evaluation. Among them, block generation time is highly related to latency. In most blockchain systems, it is also an important factor affecting throughput. Latency is related to service quality. It defines how long a user needs to wait to confirm that the message he/she submitted to the blockchain has been processed and has achieved consistency among miners. Throughput reveals system's ability of message processing. A higher throughput means that the system can process more messages within a unit time period. We further tested computational overhead because some blockchain systems result in high computational overhead. Since there are numerous mobile devices in MCS, we hope MCS-Chain is so efficient that mobile devices can participate in mining process. In MCS-Chain, we introduce trust evaluation to enhance the trust of blockchain since node trust influences block selection as well as node payment. Thus, we need to evaluate the accuracy of trust evaluation to ensure its effectiveness.

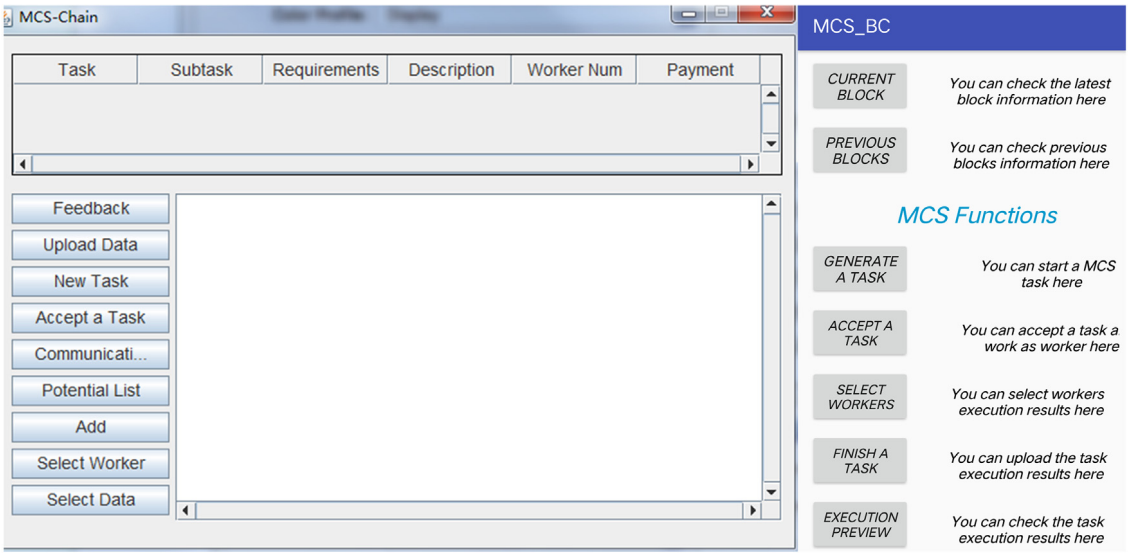

Fig. 6. MCS-Chain interface.

(1) Block generation time: The block generation time is measured as the average generation time of a single block. In our test, we configure the workload by adjusting the task generation frequency of each user and then monitor the block generation time.
(2) Latency: The latency is measured as the response time per transaction. We test the confirmation time that the system spends to confirm a transaction with regard to different task generation frequency.
(3) Throughput: The throughput is measured as the maximum number of messages successfully recorded in blockchain per second.
(4) Computational overhead: The computational overhead is measured as the CPU utilization rate and the memory usage of devices. We observe the CPU utilization rate and memory usage of both desktop and mobile phone under different task generation frequency.
(5) Accuracy of trust evaluation: The accuracy of trust evaluation is measured as the deviation between trust evaluation results and the real trust of nodes.

5.2.2. Experimental setting

Implementation environment: We implemented MCS-Chain with Java language in both Windows platform and Android platform to evaluate its performance. It is implemented in a desktop running 64-bit Windows 7 with 3.2-GHz Intel Core i5 Quad-CPU and 8G RAM, and in an Android phone running 64-bit Android 7.1.2 with 1.95-GHz Snapdragon 653 Octa-CPU and 6G RAM. Fig. 6 depicts the interface of MCS-Chain in both Windows and Android, respectively.

Node setup: In the experiment, we emulated 110 nodes in the desktop, of which 50 nodes are miners, 50 nodes are workers, and ten nodes are users. Besides, the desktop connects to an Android phone that acts as a miner in the emulated network.

Network setup: According to [54], in the Bitcoin system, the time of other miners receives a block creation announcement follows an exponential distribution. Therefore, to better evaluate the performance of our system in a real world, we emulate the real-world overlay network by adding a random latency between node communications so that the time of receiving a block announcement by the miners follows the exponential distribution.

MCS task setup: The execution of a task includes a series of message exchanges and decides the transaction frequency and communication overhead. Besides, the number of tasks conducted by MCS-Chain is also related to the block generation time. In our experiments, all the worker will generate a bid message for an issued task, and 25 of them will be randomly selected by the task issuer to execute the task and the average payment for each worker is set as 5. We set different values for *PAY* in the following experiments, ranging from 2000 to 10000. The selected workers will submit a task execution result to the blockchain. After all the task execution results are collected, all participants of the task will provide feedback to others. We observe the performance of MCS-Chain under different task generation frequencies. For easier presentation, we summarize the experiment setting in Table 2.

Table 2
Experiment settings.

Experiment setting		
Node setting	Miner	50 desktop nodes and 1 android node
	Worker	50 desktop nodes
	End user	10 desktop nodes
Task setting	Bidding message	50/task
	Data message submitted	25/task
	Average payment for a Worker	5
	Task message generation rate	0∼150 tasks/min
Block setting	*PAY* (Threshold to generate a new block)	2000∼10 000

5.2.3. Experimental results

(1) Block generation time

In this experiment, we tested the block generation time with different task generation frequency, as shown in Fig. 7. From the figure, we can see that when task generation frequency is low (2 tasks/min), the block generation time is very long (2578 s). Then the block generation time drops sharply with the increase of task generation frequency. This is because in our scheme, a miner creates a block when it records enough payment confirmation messages of which accumulated payment reaches the threshold value (10 000 in our experiments). Since each task recruits the same number of workers and average payment for a worker is set as a fixed value, the block generation time varies inversely with task generation frequency. However, when the task generation frequency reaches 20 tasks per minute, the block generation reaches its minimum, and almost no longer varies as task

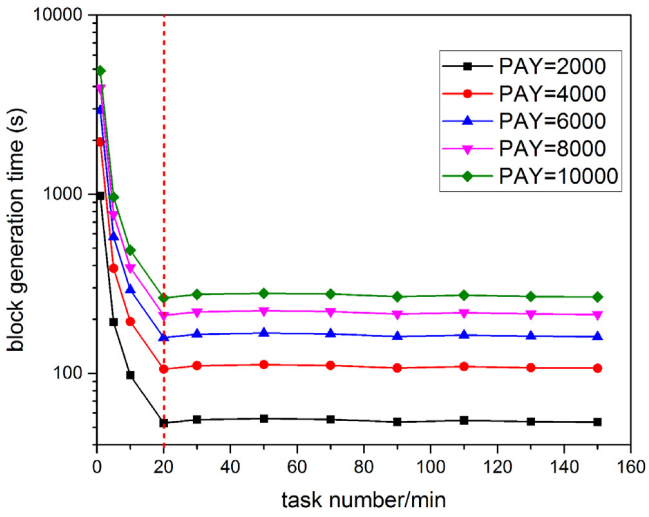

Fig. 7. Block generation time with regard to task generation rate.

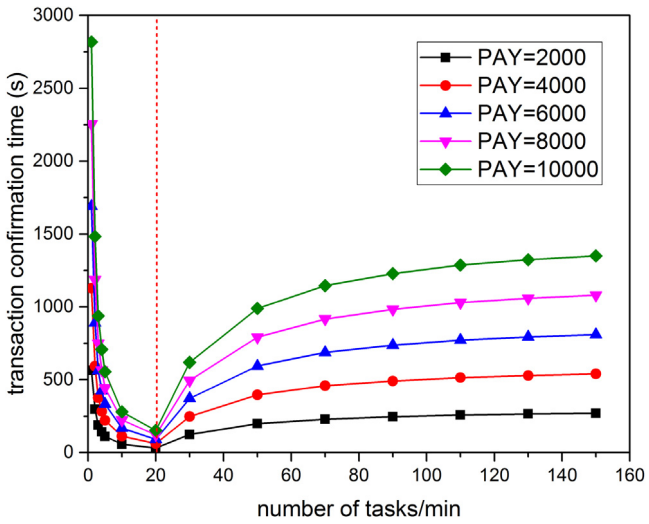

Fig. 8. Transaction confirmation time with regard to task generation rate.

generation frequency increases. This is because the system has reached its maximum processing rate. Therefore, though more tasks are generated, the system processing rate cannot catch with the task generation frequency. But the block generation time can be controlled by setting up a proper payment threshold *PAY*.

(2) Transaction confirmation latency

We tested transaction confirmation latency with different task generation frequency and observed the average latency from the first epoch to the tenth epoch. The experimental results are illustrated in Fig. 8. Similar to block generation time, transaction confirmation latency also drops sharply when task generation frequency is below 20 tasks per minute. When task generation frequency exceeds 20 tasks per minute, the average latency gradually increases with the increase of task generation frequency. This is mainly because when task generation frequency is below 20 tasks per minute, the process rate of MCS-Chain is higher than task generation frequency. Therefore, it is able to deal with all the messages received. In this case, the long confirmation time is mainly caused by the long block generation time, since only when a block is created can a message be involved in a block and further reach the consistency. However, when task generation frequency is above 20 tasks per minute, the system cannot process all the messages in time. As a result, MCS messages beyond the processing ability of MCS-Chain are blocked in sequence, which leads to the increase of latency. To further verify it, we performed an extra experiment, where we tested the transaction confirmation latency with regard to epochs under different task generation frequency. Fig. 9 presents the experimental results. In Fig. 9, when the task generation frequency is 1, 10, and 20, the message confirmation latency almost remains invariant. While task generation frequency is 30, 40, 50, 110, 150 tasks per minute, the message confirmation latency increases linearly with time. This is because task generation frequency exceeds the maximum processing rate of the system.

Based on the experimental results of experiment 1 and experiment 2, we can see when task generation frequency increases beyond 20 tasks per minute, the system cannot immediately process the related transactions. Since the execution of each task requires at least 102 messages including the execution of each task requires at least 102 messages, including 1 task request, 50 bidding messages, 1 worker selection confirmation message, 25 task execution results, we can safely conclude that the throughput of the system is 34 messages per second when *PAY* is 10 000.

(3) Computational overhead

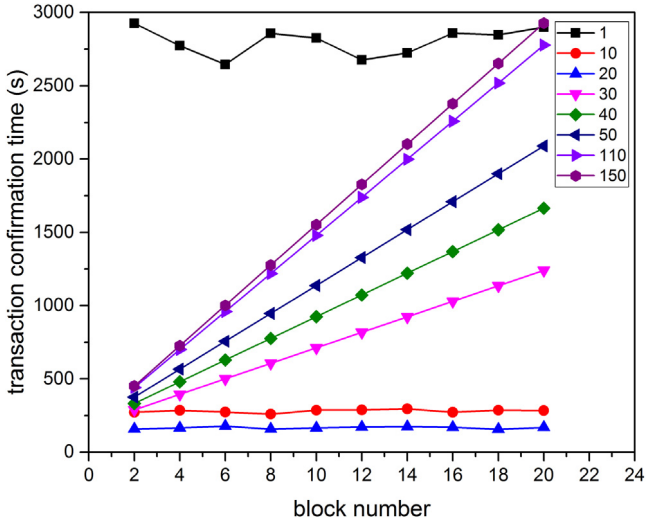

Fig. 9. Transaction confirmation time with regard to block number.

We evaluate the computing efficiency by testing the CPU utilization rate and memory usage in both desktop and mobile phone. Figs. 10 and 11 respectively depict the CPU utilization rate and memory usage with regard to task generation rate. As can be seen in the figures, when the system reaches its maximum throughput, the CPU utilization rates are 75% in mobile device and 25% in desktop, and the memory usage is 97 MB in mobile device and 460 MB in desktop. Therefore, MCS-Chain is efficient enough to run on a mobile device.

(4) Trust evaluation accuracy

We further evaluate the accuracy of trust evaluation algorithm. In these experiments, we set 100 nodes. Among them, the first 25 nodes are with low trust, the last 25 nodes are with high trust, and the remainders are with medium trust. All these nodes are assigned 0 as their initial trust. These nodes are involved in 100 tasks, and their trust values are updated after each task is finished. In our experiment, we assume the high-trust nodes generate feedback honestly and with high accuracy; the medium-trust nodes tend to generate feedback honestly and a little lower accuracy; the low-trust nodes tend to generate feedback dishonestly. The feedback was generated with a random function. If a node has a high payment, the average of its feedback will be closer to the real trust of

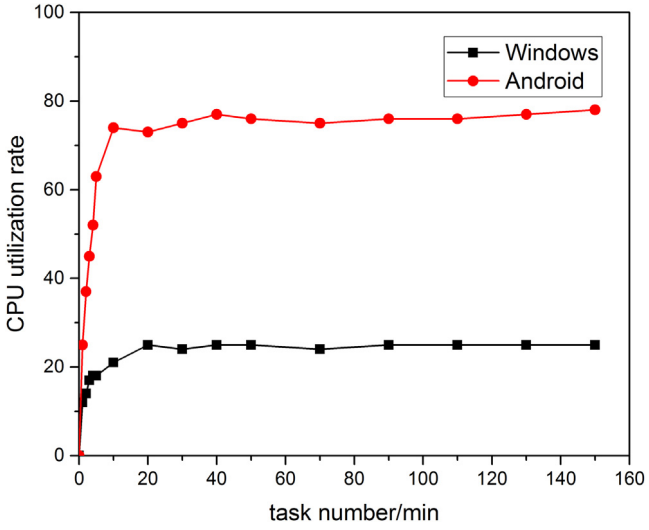

Fig. 10. CPU utilization rate with regard to task generate rate.

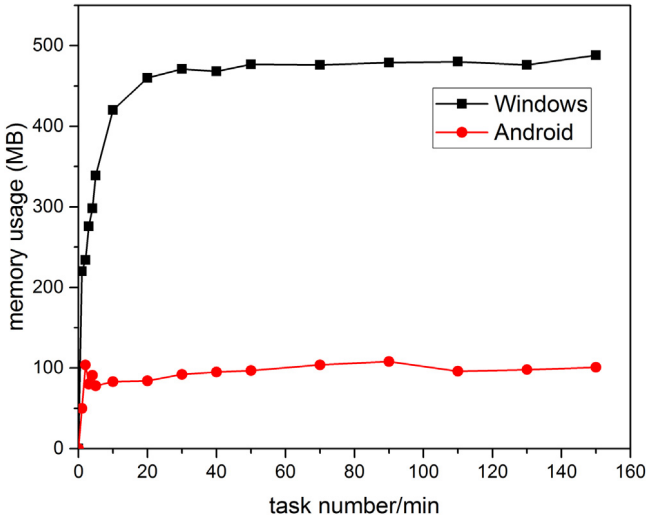

Fig. 11. Memory usage with regard to task generate rate.

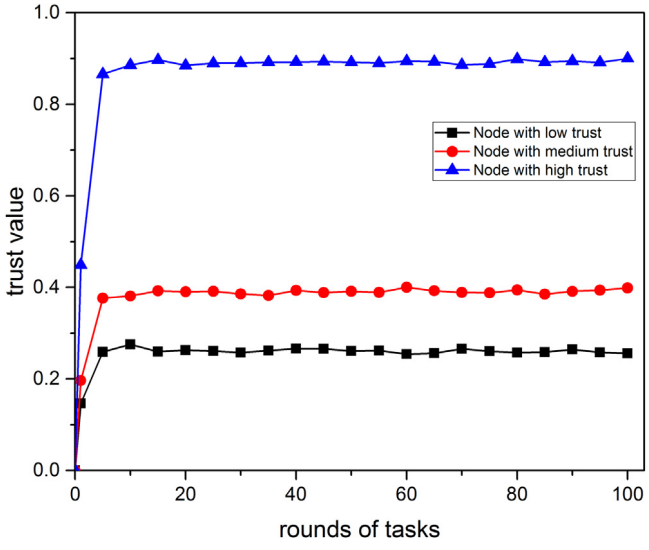

Fig. 12. Evolution of node trust.

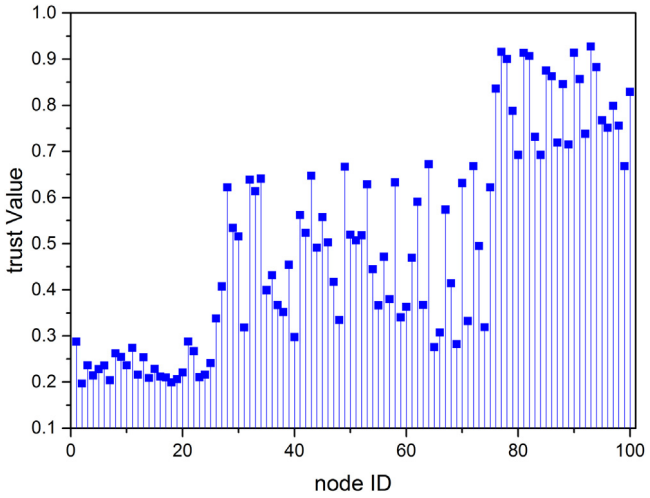

Fig. 13. Trust distribution of nodes.

evaluated nodes. Fig. 12 shows the average trust values of three types of nodes, which depicts the trust evolution of three types of nodes. From Fig. 12, we can see that our trust evaluation formula can quickly generate the accurate trust values for MCS nodes. Fig. 13 depicts the distribution of nodes after 100 tasks' execution. In this figure, the first 25 nodes have low trust evaluated, the last 25 nodes have high trust evaluated, and the remaining nodes' trust is in the middle. This result demonstrates the effectiveness of our evaluation method.

5.2.4. Comparison

To demonstrate the superiority of MCS-Chain scheme to other existing blockchain systems, we compare MCS-Chain with several blockchain systems, i.e., Bitcoin, Ethereum, Litecoin, Bitcoin Cash, and Primecoin, in both security and performance. In terms of security, we compare the realization of basic security properties, i.e., liveness, safety, decentralization, and fault tolerance. The comparison results are summarized in Table 3. Although all the other five blockchain claim to be decentralized systems, they face the challenge of centralization. Gencer et al. pointed out that for both Bitcoin and Ethereum, the mining power of several largest mining pools has occupied most of the total power. Therefore, it is possible that the several largest mining pools control the whole system by controlling the generation of new blocks. For Litecoin, Bitcoin Cash, and Primecoin, although they improve the efficiency of Bitcoin, they are basically based on PoW, and thus cannot achieve real decentralization. Different from them, MCS-Chain guarantees the decentralization by constraining the number of blocks generated by a single miner within a certain period. For fault tolerance, Eyal et al. demonstrated that the consistency of Bitcoin can be destroyed when attackers control more than 25% miners. Similarly, the other three blockchain systems also face the same problem, and as a result their fault tolerances are also 25%. The fault tolerance of Ethereum is related to how many coins are held by malicious miners. When a single miner holds more than half of existing coins in Ethereum, the miner can conduct 51% attacks [55]. Therefore, if the malicious miners control more than 50% coins, the security of Ethereum will be harmed. As proved in Section 5.1, with certain parameter setting, MCS-Chain can tolerate more than 56% malicious miners.

In terms of performance, we mainly compared MCS-Chain with Bitcoin and Ethereum in block generation time, transaction confirmation time, computation overhead, throughput, and communication cost, as shown in Table 3. We can see that block generation

Table 3
Performance comparison results.

Metrics	Security property				Performance				
	Liveness	Safety	Decentralization	Fault tolerance	Block generation time	Transaction confirmation latency	Computation overhead	Throughput	Communication complexity
Bitcoin	Yes	Yes	No	25%	600 s	3600 s	High	7 transactions per second	$O(N)$
Bitcoin Cash	Yes	Yes	No	25%	600 s	3600 s	High	56 transactions per second	$O(N)$
Litecoin	Yes	Yes	No	25%	180 s	900 s	Medium	60 transactions per second	$O(N)$
Primecoin	Yes	Yes	No	25%	60 s	360 s	High	70 transactions per second	$O(N)$
Ethereum	Yes	Yes	No	Depends on coins that the malicious miners hold.	12 s	72 s	Medium	30 transactions per second	$O(N)$
Our scheme ($Pay = 2000$)	Yes	Yes	Yes	56%	53 s	30 s	Low	34 messages per second	$O(N)$

N is the total number of miners.

time in MCS-Chain is only larger than that in Ethereum and is smaller than all other four blockchain systems. But in MCS-Chain, the generation time of a block can be dynamically adjusted by setting different thresholds of accumulated payment amount. Since a large threshold can increase the difficulty of block generation, and as a result the number of blocks generated simultaneously can be reduced, we adopt a relatively large threshold for block generation ($PAY = 2000$). Nevertheless, our block confirmation time is much shorter than Bitcoin, Litecoin, Primecoin, and Bitcoin Cash, and is similar to Ethereum. This is mainly because when multiple blocks generated at the same time, all blockchain systems except ours need to wait for at least six blocks to decide which branch is the longest. Hence, although block generation delay in Ethereum is much smaller than that in MCS-Chain, the efficiency deviation in terms of block confirmation time between MCS-Chain and Ethereum is not large. Another important performance metric is computation overhead. As known to all, miners in Bitcoin and Bitcoin Cash have to conduct time consuming computations to find the next block. Therefore, their computation overhead is extremely high. Similarly, Primecoin implements another PoW in which miners need to find a certain prime to generate a block. This makes the computation valuable, but its computation overhead is still quite high. To alleviate this problem, Litecoin reduces the difficulty of puzzle in blockchain. Nonetheless, miners in Litecoin still spend a lot of computing capacities in solving the puzzle. Differently, Ethereum leverages PoS instead of PoW, and thus greatly reduces computation overhead. Similarly, the proposed consensus algorithm in this paper also avoids all the time-consuming computing operations in PoW, and thus greatly improves efficiency. However, it also requires that miners conduct some computation. Compared with them, MCS-Chain requires no time-consuming computations. Therefore, our scheme achieves high efficiency and its computational overhead is low. For throughput, the throughput of Bitcoin is limited to the block size, and suffers from low throughput of 7 transactions per second. Bitcoin Cash, Litecoin and Primecoin respectively achieve a throughput of 56, 60, and 70 transactions per second. Ethereum improves the throughput by accelerating the block generation and can process around 30 transactions per second. For MCS-Chain, the experimental result shows that it can process as many as 20 tasks per minute. In the experiments, the execution of each task requires at least 102 messages, including 1 task request, 50 bidding messages, 1 worker selection confirmation message, 25 task execution results, and at least 25 feedbacks. Therefore, the maximum throughput is 34 messages per second. Therefore, MCS-Chain has a better throughput scalability than Ethereum. With regard to communication cost, MCS-Chain is based on the same network architecture of public blockchain systems, e.g., Bitcoin and Ethereum. So MCS-Chain has the same communication complexity as Bitcoin and Ethereum. To be specific, in the process of MCS message transmission, when a miner receives an MCS message, such as task request, bidding message, task execution results, etc., it should transmit the message to all other miners in the network. Therefore, the communication complexity of MCS-Chain is $O(LN)$ for L messages, where L is the total number of messages and N is the number of miners. In the process of block generation, since the block has to be transmitted to all miners in the network, the communication complexity is $O(N)$. Table 3 summarizes the comparison results of communication comparison results between MCS-Chain and the other five blockchain systems with regard to block generation.

We also compare MCS-Chain with Tweetchain, an alternative blockchain for crowdsourcing [47]. The main idea of Tweetchain is to leverage messages in social networks (e.g., Twitter) to publish transactions. As a result, the trust of the Tweetchain still depends on the trust of the centralized service provider. The authors of Tweetchain argued that by building a meshed transaction chain, the service providers of social networks are unable to forge, delete, or alter transactions. Besides, the availability of these service providers is high, and building Tweetchain on multiple social networks can achieve extremely high availability. Tweetchain is in nature a crowdsourcing system based on a centralized and trustworthy crowdsourcing platform. The difference between Tweetchain and traditional centralized crowdsourcing architecture is that the platform just provides information exchange services and does not realize other functions like worker selection or data processing. Since Tweetchain employs one or multiple centralized entities to improve performance, here we mainly compare MCS-Chain with Tweetchain in terms of security. For Tweetchain, the centralized service provider can maliciously deny the publication of some messages and thus hinder the normal operation of Tweetchain. In this case, Tweetchain fails to achieve liveness since some transactions cannot be published to all users. When utilizing multiple social networks, attackers can conduct double spending by publishing different transactions at the same time slot in different social networks. Therefore, it cannot well achieve safety. Since it still employs one or multiple centralized service providers as an information publishing platform, it can only achieve limited decentralization because the security still partly depends on centralized entities. Therefore, our scheme outperforms Tweetchain in security.

6. Conclusions

In this paper, we proposed a novel blockchain, in which a new block is generated when the accumulated payment amount

waiting to be recorded in the next block exceeds the pre-defined threshold. The blockchain design solves the fork issue and centralization problem existing in most public blockchain systems. Based on it, we proposed a decentralized and trustworthy MCS system, named MCS-Chain, where no any centralized service provider exists. We proved the security of MCS-Chain in theory, and implemented the MCS-Chain system in both Windows platform and Android platform. We further conducted a number of experiments to test the performance of MCS-Chain. The experimental results demonstrated the effectiveness and efficiency of MCS-Chain. However, privacy has not been explored in this paper. In the future, we will further explore privacy in MCS-Chain. In particular, we will concentrate on anonymous authentication on trust with unlinkability and preserve privacy in MCS-Chain.

Acknowledgments

This work is sponsored by the National Key Research and Development Program of China (Grant 2016YFB0800704), the National Natural Science Foundation of China (Grants 61672410 and U1536202), the Academy of Finland (Grant 308087), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), the Key Lab of Information Network Security, Ministry of Public Security (Grant C18614), and the China 111 project (Grants B16037 and B08038).

References

[1] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, Y.T. Hou, A survey on security, privacy, and trust in mobile crowdsourcing, IEEE Internet Things J. 5 (4) (2018) 2971–2992.
[2] D. Shin, D. Aliaga, S.M. Tunçer, S. Kim, D. Zünd, G. Schmitt, Urban sensing: Using smartphones for transportation mode classification, Comput. Environ. Urban Syst. 53 (2015) 76–86.
[3] B. Kantarci, H.T. Mouftah, Trustworthy sensing for public safety in cloud-centric internet of things, IEEE Internet Things 1 (4) (2014) 360–368.
[4] C. Chen, Y. Huang, Y. Liu, C. Liu, L. Meng, Y. Sun, Interactive crowdsourcing to spontaneous reporting of adverse drug reactions, in: International Conference on Communication (ICC) IEEE Sydney NSW, 2014, pp. 4275–4280.
[5] S. Hu, L. Su, H. Liu, H. Wang, T.F. Abdelzaher, SmartRoad: Smartphone-based crowd sensing for traffic regulator detection and identification, ACM Trans. Sensor Netw. 11 (4) (2015) 1–27.
[6] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, B.Y. Zhao, Defending against sybil devices in crowdsourced mapping services, in: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, 2016, pp. 179–191.
[7] B. Kosucu, B. Arnrich, C. Ersoy, A crowdsourced SkyMap, in: Signal Processing and Communications Applications Conference (SIU), 2014, pp. 1283–1286.
[8] A. Overeem, J.C. Robinson, H. H. Leijnse, G.J. Steeneveld, B.K. Horn, R. Uijlenhoet, Crowdsourcing urban air temperatures from smartphone battery temperatures, Geophys. Res. Lett. 40 (15) (2013) 4081–4085.
[9] Y. Zhang, H. Zhang, S. Tang, S. Zhong, Designing secure and dependable mobile sensing mechanisms with revenue guarantees, IEEE Trans. Inf. Forensics Secur. 11 (1) (2016) 100–113.
[10] J. Sun, H. Ma, Privacy-preserving verifiable incentive mechanism for online crowdsourcing markets, in: Proceeding of International Conference on Computer Communication and Networks, Shanghai, 2014, pp. 1–8.
[11] H. Jin, L. Su, B. Ding, N. Borisov, Enabling privacy-preserving incentives for mobile crowd sensing systems, in: IEEE International Conference on Distributed Computing Systems, 2016, pp. 344–353.
[12] C. Prandi, S. Ferretti, S. Mirri, P. Salomoni, A trustworthiness model for crowdsourced and crowdsensed data, in: Proceedings of International Conference on Trustcom/BigDataSE/ISPA IEEE Helsinki, 2015, pp. 1261–1266.
[13] J. Zhang, V.S. Sheng, J. Wu, X. Wu, Multi-class ground truth inference in crowdsourcing with clustering, IEEE Trans. Knowl. Data Eng. 28 (4) (2016) 1080–1085.
[14] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, (2008), http://www.bitcoin.org/bitcoin.pdf.
[15] G. Wood, Ethereum: A Secure Decentralized Generalized Transaction ledger, in: Ethereum project yellow paper, 115 2014 p. 1-32.
[16] I. Eyal, A.E. Gencer, E.G. Sirer, R.V. Renesse, Bitcoin-NG: A scalable Blockchain protocol, in: NSDI, 2016, pp. 45–59.
[17] I. Eyal, E.G. Sirer, Majority is not enough: Bitcoin mining is vulnerable, ACM Commun. 61 (7) (2018) 95–102.
[18] A.E. Gencer, S. Basu, I. Eyal, R.V. Renesse, E.G. Sirer, Decentralization in Bitcoin and Ethereum Networks, (2018), arXiv preprint arXiv:1801.03998.
[19] S. Luo, Y. Sun, Z. Wen, Y. Ji, C2: Truthful incentive mechanism for multiple cooperative tasks in mobile cloud, in: IEEE International Conference on Communication, 2016, pp. 1–6.
[20] Y. Wen, J. Shi, Q. Zhang, X. Tian, Z. Huang, H. Yu, X. Shen, Quality-driven auction-based incentive mechanism for mobile crowd sensing, IEEE Trans. Veh. Technol. 64 (9) (2015) 4203–4214.
[21] H. Mousa, S.B. Mokhtar, O. Hasan, O. Younes, M. Hadhoud, L. Brunie, Trust management and reputation systems in mobile participatory sensing applications: A survey, Comput. Netw. 90 (2015) 49–73.
[22] H. Amintoosi, S.S. Kanhere, A reputation framework for social participatory sensing systems, Mob. Netw. Appl. 19 (1) (2014) 88–100.
[23] R.W. Ouyang, M. Srivastava, A. Toniolo, T. Norman, Truth discovery in crowdsourced detection of spatial events, IEEE Trans. Knowl. Data Eng. 28 (4) (2016) 1047–1060.
[24] D. Zhao, X.Y. Li, H. Ma, Budget-feasible online incentive mechanisms for crowdsourcing tasks truthfully, IEEE/ACM Trans. Netw. 24 (2) (2016) 647–661.
[25] I. Krontiris, T. Dimitriou, Privacy-respecting discovery of data providers in crowd-sensing applications, in: Proceeding of IEEE International Conference on Distributed Computing in Sensor Systems, Cambridge MA, 2013, pp. 249–257.
[26] J. Ren, Y. Zhang, K. Zhang, K. Zhang, X. Shen, Exploiting mobile crowdsourcing for pervasive cloud services: Challenges and solutions, IEEE Commun. Mag. 53 (3) (2015) 98–105.
[27] H. Choi, S. Chakraborty, Z.M. Charbiwala, M.B. Srivastava, SensorSafe: A framework for privacy-preserving management of personal sensory information, in: Proceeding of Workshop Secure Data Manag, 2011, pp. 85–100.
[28] Y. Yuan, F.Y. Wang, Towards blockchain-based intelligent transportation systems, in: IEEE 19th International Conference on Intelligent Transportation Systems, 2016, pp. 2663–2668.
[29] F. Tian, An agri-food supply chain traceability system for China Based on RFID & blockchain technology, in: IEEE International Conference on Service Systems and Service Management Kunming, 2016, pp. 1–6.
[30] K. Korpela, J. Hallikas, T. Dahlberg, Digital supply chain transformation toward blockchain integration, in: Proceedings of International Conference on System Sciences, 2017.
[31] G. Zyskind, O. Nathan, A. Pentland, Decentralizing privacy: Using blockchain to protect personal data, in: IEEE Security and Privacy Workshops, 2015, pp. 180–184.
[32] Z. Yan, L. Peng, Trust evaluation based on Blockchain in pervasive social networking, IEEE Blockchain Newsl. (2018) 1–4.
[33] E. Heilman, A. Kendler, A. Zoha, S. Goldberg, Eclipse attacks on Bitcoin's peer-to-peer network, in: USENIX Security Symposium, 2015, pp. 129–144.
[34] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G.M. Voelker, S.A. Savage, Fistful of Bitcoins: Characterizing payments among men with no names, in: Proceedings of ACM Conference on Internet Measurement Conference, 2013, pp. 127–140.
[35] M. Vukolić, The quest for scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication, in: Springer International Workshop on Open Problems in Network Security, 2015, pp. 112–125.
[36] E. Heilman, F. Baldimts, S. Goldberg, Blindly.Signed. Contracts, Blindly signed contracts anonymous On-Blockchain and Off-Blockchain Bitcoin transactions, in: Springer International Conference on Financial Cryptography and Data Security, 2016, pp. 43–60.
[37] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, R.H. Deng, CrowdBC: A Blockchain-based decentralized framework for crowdsourcing, in: IACR Cryptol., Vol. 444, 2017.
[38] S. Wang, A. Taha, J. Wang, Blockchain-Assisted Crowdsourced Energy Systems, (2018) arXiv preprint arXiv:1802.03099.
[39] G. Pinto, J.P. Dias, H.S. Ferreira, Blockchain-based PKI for crowdsourced IoT sensor information, in: arXiv preprint arXiv:1807.03863, (2018).
[40] Y. Lu, Q. Tang, G. Wang, ZebraLancer: Private and anonymous crowdsourcing system atop open blockchain, in: arxiv preprint arxiv:1803.01256, (2018).
[41] K.C. Asmoredjo, A. Hovanesyan, S.M. To, C.M. Wong Loi Sing, Decentralized Mortgage Market: An Open Market for Real-estate Crowdsourcing, 2017.
[42] M. Yang, T. Zhu, K. Liang, W. Zhou, R.H. Deng, R. H, A Blockchain-based location privacy-preserving crowdsensing system, in: Future Generation Computer Systems, 2018.
[43] S. Hu, L. Hou, G. Chen, J. Weng, J. Li, Reputation-based distributed knowledge sharing system in Blockchain, in: Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2018, pp. 476–481.
[44] G.K. Bhatia, P. Kumaraguru, A. Dubey, A.B. Buduru, V. Kaulgud, WorkerRep: Building trust on crowdsourcing platform using Blockchain, in: IIIT-Delhi, 2018.
[45] A. Fujihara, Proposing a system for collaborative traffic information gathering and sharing incentivized by Blockchain technology, in: International Conference on Intelligent Networking and Collaborative Systems, 2017, pp. 170–182.

[46] J. Zou, B. Ye, L. Qu, Y. Wang, M.A. Orgun, L. Li, A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services, in: IEEE Transactions on Services Computing, http://dx.doi.org.10.1109/TSC.2018.2823705.
[47] F. Buccafurri, G. Lax, S. Nicolazzo, A. Nocera, Tweetchain: An Alternative to Blockchain for crowd-based applications, in: Springer International Conference on Web Engineering, 2017, pp. 386–393.
[48] K. Fan, S. Wang, Y. Ren, K. Yang, Z. Yan, H. Li, Y. Yang, Blockchain-based secure time protection scheme in IoT, IEEE Internet Things J. (2018) http://dx.doi.org/10.1109/JIOT.2018.2874222.
[49] K. Fan, Y. Ren, Z. Yan, S. Wang, H. Li, Y. Yang, Secure time synchronization scheme in IoT based on Blockchain, in: IEEE Blockchain, 2018.
[50] Z. Yan, Y. Chen, Y. Shen, A practical reputation system for pervasive social chatting, J. Comput. System Sci. 79 (5) (2013) 556–572.
[51] Z. Yan, Y. Chen, Y. Shen, PerContRep: A practical reputation system for pervasive content services, Supercomputing 70 (3) (2014) 1051–1074, Springer.
[52] A. Baliga, Understanding Blockchain consensus models, in: Persistent, 2017.
[53] B. Alpern, F.B. Schneider, Recognizing safety and liveness, Distrib. Comput. 2 (3) (1987) 117–126.
[54] C. Decker, R. Wattenhofer, Information propagation in the Bitcoin network, in: IEEE International Conference on Peer-to-Peer Computing (P2P), 2013.
[55] X. Li, P. Jiang, T. Chen, X. Luo, Q. Wen, A survey on the security of blockchain systems, Future Gener. Comput. Syst. (2017) http://dx.doi.org/10.1016/j.future.2017.08.020.

Wei Feng received the B.Sc. degree in telecommunications engineering from Xidian University, Xian, China, in 2011, where he is currently pursuing the Ph.D. degree in information security. His research interests include information security, privacy preservation, and trust management in social networking. He is currently a visiting research scholar in Aalto University, Finland.

Zheng Yan is currently a professor at the Xidian University, China and a visiting professor and Finnish academy research fellow at the Aalto University, Finland. She received the Doctor of Science in Technology from the Helsinki University of Technology, Finland. Before joining academia in 2011, she was a senior researcher at the Nokia Research Center, Helsinki, Finland, since 2000. Her research interests are in trust, security, privacy, and security-related data analytics. She is an associate editor of IEEE Internet of Things Journal, Information Fusion, Information Sciences, IEEE Access, and JNCA. She served as a general chair or program chair for a number of international conferences including IEEE TrustCom 2015. She is a founder steering committee co-chair of IEEE Blockchain conference. She received several awards, including the 2017 Best Journal Paper Award issued by IEEE Communication Society Technical Committee on Big Data and the Outstanding Associate Editor of 2017 for IEEE Access.