This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Feng, Wei; Yan, Zheng; Yang, Laurence T.; Zheng, Qinghua

Anonymous Authentication on Trust in Blockchain-Based Mobile Crowdsourcing

Published in:
IEEE Internet of Things Journal

DOI:
[10.1109/JIOT.2020.3018878](https://doi.org/10.1109/JIOT.2020.3018878)

Published: 15/08/2022

Document Version
Peer reviewed version

Please cite the original version:
Feng, W., Yan, Z., Yang, L. T., & Zheng, Q. (2022). Anonymous Authentication on Trust in Blockchain-Based Mobile Crowdsourcing. *IEEE Internet of Things Journal*, 9(16), 14185 - 14202.
<https://doi.org/10.1109/JIOT.2020.3018878>

Anonymous Authentication on Trust in Blockchain-Based Mobile Crowdsourcing

Wei Feng, Zheng Yan*, *Senior Member, IEEE*,
Laurence T. Yang, *Fellow, IEEE*, Qinghua Zheng, *Member, IEEE*

Abstract—Mobile crowdsourcing (MCS) has become an effective data collection method due to its mobility, low cost, and flexibility. However, since centralized MCS confronts severe security and privacy risks in reality, many researchers are devoted to building a decentralized MCS system based on blockchain. Despite the effectiveness of these schemes, they fail to offer anonymous authentication on the trust of MCS nodes, although privacy is a main concern in MCS and trust plays an important role in a series of MCS activities such as worker selection and truth discovery. Nevertheless, anonymous authentication on trust is not a trivial issue since trust evaluation usually conflicts with anonymity, which is a necessary privacy requirement in an open MCS environment. To tackle this problem, we leverage Intel Software Guard Extension (SGX) and propose a scheme to anonymously authenticate trust with trustworthy trust evaluation in a blockchain-based MCS system. The scheme employs an SGX-enabled cloud server to periodically alter user public/private key pairs and mix newly altered keys among a number of faked keys in order to ensure unlinkability. Besides, we consider the unique features of MCS and work out a novel trust evaluation method by aggregating both subjective feedback and objective behaviors. Finally, we conduct several analyses and experiments to illustrate its security and efficiency.

Index Terms—Anonymous Authentication, Mobile Crowdsourcing, Trust, Blockchain

1 INTRODUCTION

MOBILE Crowdsourcing (MCS) is an effective method for data collection and has attracted considerable attention during the past years. By using mobile devices to work as sensors to collect data, and thereby it can collect massive data easily anywhere and at any time. Besides, it avoids the deployment of infrastructures and achieves better flexibility and ubiquity compared to wireless sensor networks (WSN) [1]. MCS gains popularity in smart transportation, environment monitoring, weather monitoring, etc., and there are many MCS based applications, such as Moovit [2] and WeatherMob [3]. Most of existing MCS systems apply centralized architecture, which adopts a centralized service provider to provide MCS services. However, the centralized architecture usually confronts severe security and privacy problems due to single point of failure [4]. An MCS service provider is vulnerable to various inner or outer attacks, which may result in privacy leakage, data loss, or out of service. For another, the service provider is a rational party and may harm the profits of MCS users and workers in order to gain an extra benefit. Therefore, the centralized

MCS architecture is not feasible enough to be deployed in practice.

Currently, the popularity of blockchain provides an effective solution to the above issue by building up a decentralized MCS system [4]. Blockchain offers the possibilities of decentralization, immunity, transparency, etc., and is widely applied in cryptocurrency [5], smart contracts [6], Internet of Things (IoT) [7], Internet of Vehicles (IoV) [8], [9], [10], and so on. Till now, several blockchain-based MCS systems have been proposed. These schemes employ blockchain-based smart contracts instead of a centralized party to manage MCS nodes and MCS tasks. For example, Li et al. built CrowdBC, a decentralized MCS system based on Ethereum [4]; Feng et al. considered the unique features of MCS and designed a consensus mechanism for MCS, based on which they proposed a decentralized MCS system [11].

Motivations and Challenges: Although the aforementioned systems are effective to solve the problems in security and privacy caused by the dishonest or insecure centralized service provider, their open and decentralized architecture also introduces new privacy and trust challenges. Since there lacks a centralized party that can provide trustworthy certification on node validity, it is quite difficult for MCS nodes to build trust relationships with each other and to authenticate the trust of other nodes. Obviously, trust plays an important role in MCS task execution [1]. It helps selecting a group of reliable workers for task fulfillment and is also an important factor for collected data truth discovery. As a result, reliable trust evaluation is quite important in helping MCS end users to judge a worker's honesty and the quality of its working result [12]. In summary, the authentication on trust, which is a process to verify the trust value of a node, becomes essential in a practical MCS system.

However, authentication on trust is challenging in de-

- W. Feng is with the State Key Lab on Integrated Services Networks, the School of Cyber Engineering, Xidian University, Xi'an, China, 710071. E-mail: weifeng.ft@foxmail.com
- Z. Yan is with the State Key Lab on Integrated Services Networks, the School of Cyber Engineering, Xidian University, Xi'an, China, 710071 and the Department of Communications and Networking, Aalto University, P.O.X 15400, Espoo, Finland, 02150. E-mail: zyan@xidian.edu.cn
- L. T. Yang is with St. Francis Xavier University, Antigonish, NS, Canada, B2G 2W5. E-mail: ltyang@gmail.com
- Q. H. Zheng is with the School of Computer Science and Engineering, Xi'an Jiaotong University, Xi'an, China, 710049. E-mail: qhzheng@xjtu.edu.cn

Manuscript received May 30, 2020

centralized MCS. First, anonymous authentication on trust requires to associate a node's signing key to its trust, while there is no centralized party responsible for key generation, alteration, and revocation in a public blockchain-based MCS system. Besides, if directly employing blockchain-based smart contracts for key registration, alteration, and revocation, attackers can easily track the key alteration history of any node and then link all the activities of any node together. This greatly harms node privacy. Second, preservation on privacy, especially identity privacy is highly expected in MCS, while privacy requirements on anonymity and unlinkability conflict with accurate trust evaluation. Specifically, accurate trust evaluation requires to track and link node activities, while privacy preservation requires unlinkability between node activities to avoid malicious tracking. Third, the transparency and openness of blockchain increase the difficulty of identity privacy preservation since everyone including attackers can access the public data recorded in the blockchain. As a result, the attackers can track user activities easily.

Another challenge for anonymous authentication on trust is that there still lacks a practical trust evaluation scheme for decentralized MCS. Existing trust evaluation schemes for MCS usually rely on a trusted centralized party to guarantee the trustworthiness of the evaluated trust, and hence they cannot be applied into a blockchain-based MCS system directly. This is because blockchain leverages transparency to support public verification on the execution results of smart contracts, which conflicts with the privacy requirement of MCS nodes. Additionally, it is impractical to leverage smart contract for trust evaluation because a smart contract is costly and suffers from low efficiency. Moreover, trust in MCS is a comprehensive concept, which is related to node honesty, expertise, experience, etc. Accurate trust evaluation needs to resist various attacks raised by malicious MCS nodes, such as false bidding submission. Therefore, it is actually a challenge to provide accurate, decentralized, and privacy-preserving trust evaluation for anonymous authentication on trust.

Existing Solutions: The urgent requirements and challenges of trust and privacy motivate researchers to work out trustworthy and privacy-preserving solutions in blockchain-based MCS. In terms of blockchain, there are already several privacy solutions proposed [13], [14], [15], [16], [17]. However, most of them concentrate on transaction unlinkability in blockchain-based digital currency systems, like Mixcoin [13] and Zerocash [15]. Though effective to protect transaction privacy, these schemes fail to support identity privacy and anonymous authentication on trust and cannot be directly applied into MCS. There are few solutions of anonymous authentication so far. One typical solution introduces a centralized authority (CA) in blockchain to support authentication [17]. However, the CA can access the identity information of MCS nodes and confronts the risk of single point of failure. Besides, it does not provide authentication on trust. Based on the literature review [1], there still lacks a practical scheme for anonymous authentication on trust in blockchain-based MCS.

Existing trust evaluation schemes in MCS mainly consider the various factors that impact node trust in MCS in trust evaluation, such as worker expertise and quality

of contribution [18], [19], [20]. Typically, Amintoosi et al. considered node reliability and contribution quality to evaluate node trust [21]. More specifically, this work employs node locality, expertise, and timeliness to measure node's reliability. However, it does not consider potential attacks in MCS, such as false bidding submission. In terms of privacy preservation in trust evaluation, Hung et al. proposed a trust evaluation with privacy preservation for MCS [18], which employs trust cloaking to prevent attackers tracking nodes via trust. Nevertheless, it ignores the confidentiality of trust-related data used for trust evaluation. Some solutions employ homomorphic encryption or secure multi-party computation to solve this issue [22]. Though these schemes are effective, they usually suffer from low efficiency, especially when processing big data.

Our Solution: In recent years, Intel Software Guard Extension (SGX) has attracted considerable attention because of its ability to ensure trusted execution of an algorithm without revealing the content of processed data and processing result [23]. Owing to its advantage in security and confidentiality, people have developed many SGX-based applications with privacy preservation and trust, such as privacy-preserving collaborative machine learning [24]. Some works employ SGX to improve the security and privacy of blockchain. For example, Baliga et al. proposed using SGX for block generation to prevent centralization [25]. Besides, several SGX-based private smart contracts over public blockchain are proposed to resist privacy risks caused by the transparency of blockchain [26], [27], [28]. But with regard to anonymous authentication on trust and identity privacy preservation in Blockchain, we have not found any related existing work.

Inspired by the security and trust of SGX, in this paper, we propose to leverage Intel SGX to build a scheme for anonymous authentication on trust based on our previous work, i.e., MCS-Chain, a blockchain-based MCS system. MCS-Chain builds an efficient consensus mechanism based on trust, which enables miners to create new blocks by collecting a particular number of MCS payment-related messages and uniquely determines a block winner to avoid fork problem. Based on the consensus mechanism, it utilizes blockchain-based smart contracts to evaluate node trust and manage MCS tasks. MCS-Chain avoids single point of failure problem and achieves security and trust in MCS in a decentralized way. However, it cannot support anonymous authentication on trust and privacy-preserving trust evaluation.

Therefore, based on MCS-Chain, our scheme in this paper enables MCS nodes to periodically alter their public/private key pairs without revealing their real identities. Besides, we employ Mix-zone to mix newly altered keys among a number of faked keys to break the linkage of old and new keys in an SGX enclave. The MCS nodes can freely decide the frequency of key alteration based on their privacy requirements. Moreover, we consider the unique features of MCS and work out a novel trust evaluation method by aggregating both subjective feedback and objective behaviors. We also leverage SGX to protect trust-related data during trust evaluation. By making use of the SGX enclave, we successfully link newly updated node key pairs with the newly evaluated node trust in a secret and unlinkable

way and announce the node trust with its new public key in the blockchain, thus solve the dilemma between trust evaluation and node anonymity. Our scheme overcomes the privacy and trust challenges in decentralized MCS. The trust evaluation guarantees the accuracy of evaluated trust and achieves privacy preservation on trust-related data. The anonymous authentication on trust helps MCS nodes judge whether another node is trusted or not. Nodes can alter their keys without revealing the linkability between their old and new public keys. Therefore, it can solve the conflict problem between privacy and trust in the open and decentralized MCS.

Specifically, the contributions of this paper are summarized as below:

- 1) We employ SGX to achieve anonymous authentication on trust in blockchain-based MCS. The proposed scheme enables verification on the trust of MCS nodes without revealing their real identities. Based on our proposed key management scheme, MCS nodes can alter their public/private keys to prevent malicious tracking.
- 2) We develop a decentralized trust evaluation method based on blockchain in decentralized MCS. It considers both the trust based on subjective feedback and the trust based on objective behaviors to generate an aggregated trust value, which can overcome a number of potential attacks and can well reveal the real trust value of a node. We integrate the SGX technology with blockchain to achieve trustworthy and privacy-preserving trust evaluation in a decentralized MCS.
- 3) We conduct several security analyses to demonstrate the security as well as privacy of the proposed scheme. Besides, we implement both anonymous authentication on trust and trust evaluation to evaluate their efficiency, effectiveness, as well as the robustness of trust evaluation over a number of attacks.

Organization: The paper is organized as follows. Section 2 reviews the related works with regard to anonymous authentication, privacy preservation in blockchain and SGX applications. In Section 3, we present the system model, security model, as well as our design goals. In Section 4, we describe the details of the proposed scheme about anonymous authentication on trust and the trust evaluation method. After it, we illustrate the security analysis in Section 5 and experimental performance evaluation in Section 6, respectively. Finally, we conclude our paper in Section 7.

2 RELATED WORKS

In this section, we analyze existing works on anonymous authentication, privacy preservation in blockchain and the typical applications of Intel SGX.

2.1 Anonymous Authentication

Anonymous authentication is applied in various applications such as Vehicular Ad-Hoc Networking (VANET) [29], [30], social networking [22], [31], Mobile Ad-Hoc Network (MANET) [32], etc. A basic idea to achieve anonymity is utilizing pseudonyms rather than a real identity for communications and frequently changing pseudonyms to

avoid malicious tracking [33]. Its main drawback is that it introduces extra difficulty to trust evaluation since the trust value based on a revoked pseudonym needs to be mapped to its altered new one. In the past work, the centralized party that takes the responsibility to perform this match can easily track the behaviors of all nodes, thus should be fully trusted, otherwise, node privacy is hard to be ensured. Therefore, these schemes cannot be applied into decentralized scenarios without any fully trusted parties.

Differently, some schemes adopt group signature to achieve anonymity, which allows a user to sign a message on behalf of a group of users [34]. Therefore, there is no way to specify the identity from the groups and track the user by linking its signatures together. However, key revocation is still an open issue. It still lacks a very efficient group private key revocation method. To overcome the weaknesses of group signature, Jiang et al. proposed to employ hash-based message authentication code (HMAC) to realize efficient group key revocation [29]. The scheme includes a trusted authority that issues a secret to all valid vehicles, and a valid vehicle uses the secret to generate an HMAC on each message to prove its validity. However, this scheme relies on a centralized party, which can easily obtain the real identity of VANET users and track their messages. As a result, its efficacy highly relies on the trust of the centralized key issuer. Some other schemes employ ring signature to achieve anonymity [35], which helps hiding the real signer in a group of users. It is effective to protect identity privacy and resist malicious tracking. However, ring signature cannot well support trust evaluation since it is impossible to link user activities in the process of user trust evaluation.

In summary, existing anonymous authentication schemes either fail to preserve identity privacy in a decentralized way or cannot support accurate trust evaluation. Therefore, these works cannot be directly applied into blockchain-based MCS systems.

2.2 Privacy Preservation in Blockchain

Privacy preservation is a significant topic in blockchain and has drawn considerable attention. There are mainly four categories of privacy preservation schemes for blockchain, i.e., mix-coin based schemes [13], zero-knowledge proof based schemes [14], [15], ring signature based schemes [16], and centralized authority based schemes [17], [36]. Among them, mix-coin based schemes prevent transaction linkability in blockchain-based cryptocurrency systems like Bitcoin. The first mix-coin based scheme is Coinjoin [13], which simply employs a third party to combine several transactions together to break the linkage relationship between transaction input and output. Obviously, a dishonest third party could harm the privacy of users. Motivated by Coinjoin, researchers developed several mix-coin based schemes including Mixcoin [37], Coinswap [38], Coinshuffle [39], Blindcoin [40]. Though these methods are effective to resist malicious linking transactions, they cannot be directly applied for anonymous authentication on trust.

Similarly, zero-knowledge proof based schemes aim to preserve transaction privacy. Typical schemes include zero-coin [14] and zerocash [15]. Among them, zero-coin allows a coin holder to change its bitcoin to a zerocoin. Then, the

coin holder could change the zerocoin to a new bitcoin by proving to miners that he/she owns a zerocoin. The proof is constructed with zero-knowledge proof, which does not reveal any information about the previous bitcoin the user holds. Therefore, it is impossible to link the old coin and the new coin together. Based on a similar idea, the designers of zero-coin further design zerocash, which enables a user to directly prove to miners that he/she owns enough coins without revealing any identity information. It is possible to utilize zero-knowledge proof to build an anonymous authentication on trust. However, zero-knowledge proof suffers from poor efficiency and high computational cost. Besides, how to initialize a system in a decentralized environment (i.e., set up the private parameters needed for zero-knowledge proof), the authors did not provide a concrete secure method.

The other two types of schemes are able to construct anonymous authentication. However, the ring signature scheme [16], though effective in protecting user identity privacy, fails to support trust evaluation. The centralized authority based methods [17], [36] can support both trust evaluation and anonymous authentication. However, it still assumes a centralized and trustworthy authority existing in the system. Therefore, there still lacks a practical and effective scheme to anonymously authenticate trust in blockchain-based MCS.

2.3 Trust Evaluation in MCS

Trust plays a crucial role in many applications [41], [42], [43]. In MCS, it helps to select a group of trusted workers for task fulfillment and is also an important factor for collected data truth discovery. Many trust management schemes were proposed to evaluate the accurate trust of MCS workers. Ren et al. proposed to use contribution quality and worker bidding for worker trust evaluation [18]. However, they did not consider worker expertise, location, etc. Huang et al. further considered worker attributes like expertise and locality for trust evaluation [19], [20], but the scheme does not consider the subjective factors in trust. Therefore, the above schemes cannot evaluate accurate trust of MCS nodes.

One scheme that considers both is EndorTrust [19], which considers end users' feedback on the contribution of workers and endorsement of workers. EndorTrust employs three factors to evaluate endorsement of a worker, i.e., the reputation of its endorsers, degree of the endorse matters, and heterogeneity of the task conducted by the worker (e.g., expertise). Therefore, it can evaluate worker trust more accurately. However, for the objective factors, it only considers the attributes of workers, while a dishonest worker can conduct other malicious behaviors, such as false bidding submission and free riding attack, which cannot be detected by EndorTrust effectively. To summarize, there lacks a trust evaluation scheme that comprehensively considers both the subjective and objective factors.

2.4 SGX and Its Applications

Intel SGX, launched in 2013, does not depend on the security state of firmware and software. It extends the access control mechanism through a new set of instructions to achieve isolated operation between different programs. Meanwhile,

it ensures that the confidentiality of user-critical codes and data are not compromised by privileged malware [23]. Trusted execution of codes and confidentiality of data stored in SGX enclaves can be guaranteed. In addition, local attestation between local enclaves and remote attestation among enclaves on different machines are both supported. SGX has become a popular technique and been widely applied to ensure trust and privacy in various fields, like machine learning [44], big data [45], and data analytic [46] etc.

SGX has been used to enhance the security and privacy of blockchain [25], [26], [27], [28], [47]. Some employ SGX for block generation and validation to resist the centralization problem in blockchain and attackers cannot control a blockchain system by controlling a majority of miners, such as Proof of Elapsed Time (PoET) and Proof of Luck (PoL) [25], [47]. Besides, several SGX-based private smart contract schemes are developed to solve the privacy leakage problem due to the transparency of public blockchain [26], [27]. A typical one is Hawk [26] that employs managers equipped with SGX to execute smart contracts with disclosing any information about contract input. Similarly, Brandenburger et al. used SGX to protect smart contract privacy for Hyperfabric [28]. However, how to employ SGX technique to achieve anonymous authentication on trust has been rarely explored.

2.5 MCS-Chain

The scheme in this paper is based on MCS-Chain, a decentralized MCS system based on blockchain [11]. MCS-Chain includes four parts: blockchain-based MCS service platform, blockchain-based trust evaluation, consensus mechanism, and incentive mechanism. We here present the detailed design of them since MCS-Chain is the basis of this paper work.

Blockchain-based MCS platform: We build an MCS platform on top of blockchain for automatic MCS task execution. An MCS user can publish an MCS task in blockchain for fulfillment. All MCS workers willing to conduct the task can submit an application to blockchain for recording to execute the task. Then the MCS user can select a group of workers and reach an agreement with each selected worker on task fulfillment and payment. After task execution, each worker sends its contribution to the task to the MCS user and record a contribution digest in blockchain for verification.

Blockchain-based trust evaluation: When a task is finished, each participant generates feedback on other participants based on their roles (i.e., miner, worker, and end user). Miners receive and record all the feedbacks, and when a miner generates a new block, it will use the feedback in the created block to evaluate node trust. Other miners receiving the block will verify the correctness of the evaluated trust, which ensures the trustworthy trust evaluation.

Consensus mechanism: To handle resource waste and low efficiency issues of Bitcoin, MCS-Chain involves a novel consensus mechanisms based on trust, which is free from high computation and avoids fork issue. To be specific, in MCS-Chain, a miner can create a new block by collecting a number of payment messages whose accumulated payment exceeds the pre-defined threshold, so that it avoids high

computation. It also solves blockchain fork with a unique block selection algorithm that decides a unique block winner from a number of candidate miners by considering the block creation timestamp, the trust of creator, and so on. Besides, by limiting the winning times of a single miner, it can alleviate the centralization problem confronted by many real-world blockchain systems such as Bitcoin and Ethereum.

Incentive mechanism: MCS-Chain contains an incentive mechanism that motivates nodes to behave honestly for better trust. The incentive mechanism decides the payment gained by a worker or a miner. Generally, a miner/worker with a higher trust can obtain more payment, which not only motivates mobile users to participate in MCS but also encourages them to behave honestly.

3 PROBLEM STATEMENTS

In this section, we define the system model and security model. Then, we introduce our research assumptions with justifications and show the design goals.

3.1 System Model

Fig. 1: System Model

MCS-Chain+: The work in this paper is based on the structure of MCS-Chain [11], which builds a decentralized MCS system based on blockchain. However, it does not consider any privacy protection issue. We extend MCS-Chain to achieve advanced privacy, trust, and efficiency.

Fig. 1 shows the system model of the MCS-Chain+, where numerous nodes (including mobile devices, personal computers, and servers) are connected via various networks, like Mobile Ad-Hoc Networks (MANET), cellular networks, Wi-Fi, etc. All nodes are categorized into four types, i.e., end users, workers, miners, and cloud servers (storage system and SGX enabled cloud). Among them, the end users are the nodes that are incapable of finishing a data collection task, thus they employ the MCS system to distribute the task to a group of mobile users (i.e., workers) for execution. The workers are the mobile users that utilize their mobile devices as sensors for data collection to finish the task. The miners cooperatively maintain the blockchain system represented by $BC = B_0, B_1, \dots, B_k$, and $B_i, (0 \leq i \leq k)$ is the i th block. BC acts as a decentralized platform for MCS. Considering that there are already several blockchain-based decentralized MCS frameworks, it is possible to leverage blockchain to replace a centralized service provider to manage MCS tasks and nodes. A cloud server denoted by CS that is equipped with Intel SGX is responsible for node key alteration and trust evaluation by

setting up an SGX enclave EN_{CS} . Any cloud server with satisfied SGX equipped can take this responsibility.

Block Structure: For better understanding, we summarize the block structure of MCS-Chain+ in Fig. 2. It contains the hash value of previous block B_{k-1} , the timestamp TS_k when the block B_k is generated, the public keys of the SGX enclaves established by different servers, the public key hash of the B_k 's creator M_j , and hashes of MCS-related data $DT^H = \{H(dt_i)\}_{1 \leq i \leq |DT^H|}$. Blockchain records $H(dt_i)$ rather than $H(ct_i)$ because the data dt_i may be further re-encrypted for different intentions. For example, a data previously encrypted ABE can be re-encrypted with homomorphic encryption to support data processing with privacy protection. Recording $H(ct_i)$ in blockchain is not feasible to support data verification in this scenario. Specifically, BC records hashes of four types of MCS-related data, i.e., task metadata, bidding messages, worker incentive information, contribution evaluation results, while the full data contents are stored in S_{str} in ciphertext. All data hashes are organized by their task id ID^{Task} . Besides, it also records revoked key list $KList_{rev}$, altered key list $KList_{alt}$, key registration list $KList_{reg}$, and the hash digests π_{DT} of the corresponding trust-related data used to evaluate trust of public keys in $KList_{rev}$. Each item in $KList_{rev}$ includes the old public keys. Each item in $KList_{alt}$ includes a new key with its corresponding trust value. For $KList_{reg}$, it only records the newly registered public keys since they are all attached with an initial trust value (i.e., 0.001).

Outer Storage System: Different from MCS-Chain, MCS-Chain+ includes an extra storage system identified by S_{str} . The storage system stores data of MCS activities including node feedback and behavior data, and it can protect data security with an extra access control mechanism [48]. BC only records the data hashes for validity verification. Specifically, when a node N_i wants to store a piece of data dt , it selects an encryption K , encrypts dt , and sends the encrypted data ct to S_{str} . It also sends dt 's hash $H(dt)$ to BC for recording. When the enclave EN_{CS} setup by CS is willing to request the plain dt , it gains the decryption key K from N_i via secure channel and requests ct from S_{str} . Then EN_{CS} could decrypt ct with K in its protected memory. With the $H(dt)$ in BC , EN_{CS} is able to check the validity of the received data.

Procedures of Task Execution: The process of a task execution in MCS-Chain+ can be described as below. An end user can publish a task to BC , and workers willing to conduct the task can submit a request for the task with the worker's attributes such as location, expertise, etc. BC employs smart contract for worker selection, and the selected workers conduct the task and return their contribution to BC . We should note that all data including task metadata, worker attributes, and contributions are protected with the outer storage system S_{str} , and BC only records their digests. Herein, we use SGX-based smart contracts to process these data without revealing their contents. Since our paper mainly focuses on identity privacy preservation regarding node key alteration and trust evaluation and there are already several privacy-preserving smart contracts based on SGX [26], we will not detail the design of privacy-preserving incentive scheme and quality evaluation scheme.

Fig. 2: Block Structure of MCS-Chain+

3.2 Security Model and Research Assumptions

3.2.1 Security Model

No fully trusted node exists in the system. To be specific, all the miners, workers, the cloud servers, and end users behave rationally and are profit-driven, and they may behave dishonestly or maliciously to harm the benefits of others. They refer to the public information recorded in *BC* to make decisions. In our security model, either a miner, an end user, a cloud server, or a worker can be an adversary, and we consider an adversary that can corrupt a number of miners, end users, cloud servers, or workers. However, it cannot break the security of *BC* or *SGX*. It can conduct several attacks aiming at the proposed scheme and trust evaluation. Herein, we present a series of potential attacks targeting at trust evaluation as below:

- Bad mouthing attack:** Some trust evaluation is performed based on node feedback. However, an adversary can always provide negative feedback on a node regardless of its behaviors. Bad mouthing attack ruins the trust of an honest node, which impacts its motivation to participate in MCS activities. Therefore, a practical trust evaluation should be able to detect the bad mouthing attack and eliminate its negative impact on evaluation.
- Ballot stuffing attack:** Similar to the bad mouthing attack, ballot stuffing attack aims at feedback-based trust evaluation as well. An adversary can generate positive feedback on another node even it is dishonest or malicious. The ballot stuffing attack helps increasing trust of dishonest nodes quickly and hence harms fairness. In reality, trust evaluation should be able to resist the ballot stuffing attack.
- White washing attack:** This attack aims at a trust-based system where a node with low trust can re-join the system with a new identity in order to get rid of its low trust record. A common solution to resist the white washing attack is to attach a unique credential to each node for identity verification. However, this solution requires a centralized and trusted authority for node identity management, which may suffer from single point of failure.

3.2.2 Research Assumptions

We hold the following assumptions in the design of our scheme with justification based on the system model and security model.

- We assume the security and trust of Intel *SGX*, which provides the confidentiality of data processing in *SGX* enclaves and ensures the trusted execution of the codes in the enclaves. In our scheme, we do not consider side channel attacks on *SGX*, which has been widely discussed in many works with several proposed countermeasures [49], [50]. Besides, no one can forge the signature generated by *SGX* enclaves.
- We assume the security and trust of the blockchain, which means though there are malicious and compromised miners, attackers cannot control the blockchain system by controlling a sufficiently number of miners, this assumption is based on the MCS-Chain work in [11]

3.3 Technical Challenges and Design Goals

In this subsection, we analyze the technical challenges confronting anonymous authentication on trust and present the design goals of our scheme.

3.3.1 Technical Challenges

We consider a series of technical challenges to design a scheme for privacy-preserving authentication on trust in decentralized MCS. Specifically, we take into account the following three challenges:

- It is a challenge to support both anonymous authentication on trust and accurate trust evaluation in decentralized environment simultaneously.
- It is a challenge to achieve the confidentiality of trust-related data while supporting efficient, verifiable, and accurate trust evaluation
- It is a challenge to build a comprehensive and accurate trust evaluation by taking into account various attacks.

3.3.2 Design Goals

The goal of this work is to overcome the above challenges by proposing an anonymous and trustworthy authentication scheme in a decentralized MCS system based on blockchain. It can protect node identity privacy and at the same time provide accurate trust evaluation. It also intends to enable MCS nodes to authenticate the trust of others and provide a reliable evidence for them to justify their decision. Detailed design goals are presented as follows:

- We aim to build an *SGX*-based anonymous authentication scheme on trust for decentralized MCS, which prevents identity privacy leakage, key alteration forging and malicious tracking.
- We aim to design a privacy-preserving trust evaluation scheme that evaluates node trust accurately and meanwhile protect trust-related data from leakage. Besides, it considers both subjective feedback and objective factors of a node to improve the accuracy of evaluation.

4 ANONYMOUS AUTHENTICATION ON TRUST BASED ON SGX

In this section we first overviews the proposed scheme and then present the design details. For easy presentation, we summarize the notations in the paper in Table 1.

TABLE 1: Notations

Notation	Description
$CS/M_i/N_j/S_{str}$	The Cloud Server/ Miner i / MCS node j /Storage server
EN_{CS}	The enclave setup by CS
$PK_{CS}^{EN}, SK_{CS}^{EN}$	The key pairs of the enclave in CS
GSK_{CS}^{QE}	The private key of Quote Enclave (QE) in CS
GPK	The public group key of SGX
s_{N_i}	The key alteration signal of N_i
$SIGN(\cdot)$	The algorithm for signature generation
$VERIFY(\cdot)$	The algorithm for signature verification
$ENC(\cdot)$	The encryption algorithm
$DEC(\cdot)$	The decryption algorithm
$Perm_\gamma(\cdot)$	The permutation function
$KList_{reg}$	The key registration list
$KList_{rev}/KList_{alt}$	The revoked/alterd key list
π_{DT}	The hashes of all trust-related data used for trust evaluation
dt/ct	The plain/encrypted MCS-related data, respectively
$N_{Thr}^{alt}/N_{Thr}^{reg}$	The threshold on public key number for revoked/alterd list issue
N_l^{tv}	The required minimum number of keys with TV_l in key alteration issue
$T_{Thr}^{reg}/T_{Thr}^{reg}$	The time threshold for key registration list/alterd key list issue
Bt_r/Bt_m	The bloom filters that store trusted behavior patterns/untruted behavior patterns, respectively
$TV_{j,r,k}/TB_{j,r,k}$	The feedback/behavior-based trust of N_j for its role of r in the k th block
$TE_{i \rightarrow j,r,id}$	The feedback generated by N_i for N_j for its role r w.r.t. task id
θ	The threshold to judge whether a node is honest according to its evaluated trust
n_h	The number of nodes involved in a certain MCS task whose trust is higher than θ
ω_1/ω_2	The weights to aggregate the two types of trust
α_1, α_2	The adjusting parameters to establish ω_1

4.1 Overview

The proposed scheme is composed of three parts: SGX-based key management, blockchain-based anonymous authentication on trust, and blockchain-based trust evaluation.

1. SGX-based Key Management

Fig. 3 depicts SGX-based key management. A cloud server downloads the agreed codes from a community or blockchain and sets up an enclave EN_{CS} as well as the public/private key pair for the enclave. Miners verify whether EN_{CS} is correctly established via remote attestation, and the public key of the correctly established enclave will be recorded in BC . We allow multiple cloud servers to work for key management simultaneously, which helps improving efficiency and avoiding the single point of failure problem.

All MCS nodes employ the hash values of their public keys as identifiers to gain anonymity. A node, i.e., N_j , willing to join in MCS first selects a key pair and registers its public key to BC and obtains its initial trust value (i.e., 0.001) with the assistance of EN_{CS} . All nodes periodically

alter their key pairs to prevent malicious tracking. If a node N_j wants to alter its key pair, it selects a new key pair and generates a key alteration signal to EN_{CS} in a secure way. EN_{CS} collects all trust-related data from S_{str} and invokes trust evaluation to evaluate its latest trust. EN_{CS} constantly receives and processes key alteration signals from nodes. When the number of un-published altered keys exceeds a predefined number, or the EN_{CS} has waited for a sufficient long period, it will batch all public keys together, generate two key lists (i.e., revoked key list and altered key list) and send them to BC for recording.

Fig. 3: Scheme Procedure

2. Anonymous Authentication on Trust

BC verifies the published key lists and records the valid lists into BC . Once the lists are confirmed, the keys in the revoked key list are considered as revoked and can no longer be used. Nodes can use the keys in the altered key list to participate in MCS activities and authenticate trust with each other, and anyone can verify the trust of another node by querying the trust corresponding to its public key from BC .

3. Privacy-Preserving Trust Evaluation

In our scheme, all trust-related data are stored and transmitted in ciphertext, and we employ SGX enclave setup by the cloud server to evaluate trust to ensure data confidentiality when evaluating trust. Our trust evaluation is composed of two parts: feedback-based trust evaluation and behavior-based trust evaluation. EN_{CS} conducts trust evaluation each time it receives a key alteration signal from a node. EN_{CS} first collects the feedback on this node from S_{str} and calculates its feedback-based trust TV . Besides, EN_{CS} deploys a series of normal behavior patterns and abnormal behavior patterns ahead of time, and it will request behavior data from S_{str} and use the pre-deployed patterns to evaluate whether the node's behaviors are honest or not. With the evaluation results, the EN_{CS} further generates the behavior-based trust and finally aggregates both feedback-based trust and behavior-based trust to get the final trust.

4.2 SGX-based Key Management

The SGX-based key management consists a series of algorithms, i.e., *System Setup*, *Key Registration*, and *Key Alteration and Revocation*, and we describe their details in this subsection.

4.2.1 System Setup

In *System Setup*, CS sets up an SGX enclave with the agreed codes (represented as *code*) for key management, which can be published in community or blockchain and is publicly verified. First, CS sets up an enclave with *code*. The enclave, denoted as EN_{CS} has a public/private enclave key pairs $(PK_{CS}^{EN}, SK_{CS}^{EN})$. Then Quote Enclave (QE) of CS signs on the content of EN_{CS} with its group private key GSK_{CS}^{QE} to generate a signature as a proof π_{CS}^{EN} . Miners can verify the correctness of the enclave by verifying the generated π_{CS}^{EN} with the assistance of Intel Server. If the miners reach consistency on the validity of the established enclave, they record PK_{CS}^{EN} into BC . In this case, the CS is ready for MCS-Chain+ key management. We should note that our scheme allows multiple cloud servers to work for trust management simultaneously, which helps resisting the single point of failure.

4.2.2 Key Registration

All MCS nodes employ the hash values of their public keys as identifiers. However, if a node N_i plans to participate in MCS activities, it has to register its public key into EN_{CS} . It first picks up a key pair (PK_{N_i}, SK_{N_i}) , generates a signature $sig = SIGN(H(PK_{N_i}), SK_{N_i})$, where $H(\cdot)$ denotes one-way hash function. Then it encrypts them with the public key of EN_{CS} PK_{CS}^{EN} and generates a registration signal as follows:

$$s_{N_i}^{reg} = ENC(PK_{N_i} || sig_{N_i}, PK_{CS}^{EN}) \quad (1)$$

EN_{CS} decrypts and verifies the validity of $s_{N_i}^{reg}$. If PK_{N_i} has not been used by others, it triggers an initial trust value (i.e., 0.001) to PK_{N_i} and inserts PK_{N_i} to key registration list $KList_{reg} = \{PK_{N_i}\}$. EN_{CS} periodically publishes key registration list to BC , which verifies the correctness of the list and record it into BC if it is valid. Once the list is recorded into BC , the keys in the list are successfully registered.

4.2.3 Key Alteration and Revocation

Each public key in BC is attached with an expiring time t_e , and once it meets t_e , the public key is revoked and can be no longer used. Therefore, MCS nodes must alter their keys periodically. The process of key alteration is divided into three steps, i.e., key alteration signal generation, key alteration signal processing, and public key list issue, which are described as follows.

Step 1: Key Alteration Signal Generation

Each MCS node can generate a signal to CS for key alteration. Suppose there is a node N_i that wants to alter its key pair, it first select a new key pair $(PK_{N_i}^{new}, SK_{N_i}^{new})$ and generates a signature on $H(PK_{N_i}^{old} || PK_{N_i}^{new})$ with its old private key $SK_{N_i}^{old}$. Then, it encrypts $PK_{N_i}^{old} || PK_{N_i}^{new} || sig_{N_i}$ with PK_{CS}^{EN} . Then N_i sends the key alteration signal s_{N_i} to CS , and the format of s_{N_i} is defined as follows:

$$s_{N_i} = ENC(PK_{N_i}^{old} || PK_{N_i}^{new} || sig_{N_i}, PK_{CS}^{EN}), \quad (2)$$

where $sig_{N_i} = SIGN(PK_{N_i}^{old} || PK_{N_i}^{new}, SK_{N_i}^{old})$

Step 2: Key Alteration Signal Processing

CS 's enclave EN_{CS} constantly receives and processes key alteration signals. When receiving such a signal of N_i , EN_{CS} decrypts the signals and verifies the signature. If it is a valid signature, EN_{CS} requests all trust-related data from S_{str} , decrypts the data, verifies their validity with the data hashes queried from BC , and evaluates N_i 's trust $TV_{N_i,t}$. EN_{CS} maintains two lists in its trusted memory, i.e., revoked key list $KList_{rev}$ and alteration key list $KList_{alt}$, where $KList_{rev} = \{PK_{N_i}^{old}\}$ stores old public keys of key alteration requesters and $KList_{alt} = \{PK_{N_i}^{new} || TV_{N_i,t}\}$ stores new public keys of alteration requesters and the newly evaluated trust values. After obtaining the trust value of N_i denoted as $TV'_{N_i,t}$, EN_{CS} first processes $TV'_{N_i,t}$ with trust anonymization and obtains the anonymized trust value $TV_{N_i,t}$. Specially, given an integer N_{TV} , we have $TV_{N_i,t} = \lfloor TV'_{N_i,t} \cdot N_{TV} \rfloor / N_{TV}$. In this way, the different trust values are mapped to N_{TV} values, which helps preventing attackers tracking MCS nodes with their trust values. The value of N_{TV} is determined ahead of time by balancing privacy and trust. Then EN_{CS} inserts $PK_{N_i}^{old}$ into $KList_{rev}$ and inserts $PK_{N_i}^{new} || TV_{N_i,t}$ into $KList_{alt}$. The $KList_{rev}$ aims to prevent the reuse of old keys as new keys.

EN_{CS} sets a timer when it receives the first key alteration signal. If the timer reaches the predefined time threshold T_{thr}^{alt} , or the number of un-published public keys reaches the predefined number threshold N_{thr}^{alt} , EN_{CS} publishes the two lists. Besides, it will also publish a proof π_{DT} that proves the validity of trust-related data for trust evaluation, and it contains the hashes of all trust-related data used during trust evaluation. π_{DT} aims to support miners to verify the correctness and completeness of the trust-related data used for evaluation.

Before publishing the key alteration results, EN_{CS} needs to conduct some extra processing on them. First, EN_{CS} needs to insert some fake old and new public keys into each list so that $KList_{alt}$ must contain N_l^{tv} new public keys with trust value TV_l . This aims to prevent attackers from linking nodes' new keys with their old keys through analysis on trust values' relationships. We assume the EN_{CS} maintains a number of virtual nodes that pretend to be normal nodes as miners, workers, or end users. The virtual nodes will participate in virtual MCS tasks generated by virtual nodes and have various trust values. When inserting fake keys into $KList_{rev}$ and $KList_{alt}$, EN_{CS} selects virtual nodes of the required trust value, generates new key pairs for them, and inserts their old public keys and new public keys into the two lists respectively. Second, EN_{CS} invokes a permutation function $Perm_\gamma(\cdot)$ and uses different randoms γ_1 and γ_2 to arrange the order of the entries in each list. In this way, attackers cannot infer the linkable relationship from a key's location in the list. Algorithm 1 shows the algorithm of key alteration signal processing.

Step 3: Public Key List Issue

After processing the two lists, EN_{CS} signs $KList_{rev}$, $KList_{alt}$ and π_{DT} , and sends them to the miners. Miners verify both the signature and π_{DT} to check whether EN_{CS} uses the correct and all the trust-related data for trust evaluation. If the verification holds, they record them into BC . Once the block recording the two lists is confirmed, the old keys in $KList_{rev}$ become invalid, and key alteration

Algorithm 1 Key Alteration Signal Processing Algorithm

Input: N_{thr}^{alt} , two randoms $\gamma_1, \gamma_2, \{N_l^{tv}\}, T_{thr}^{alt}$
Output: $\pi_{DT}, KList_{rev}, KList_{alt}$

```

1:  $\pi_{DT} \leftarrow \phi$ 
2:  $KList_{rev} \leftarrow \phi$ 
3:  $KList_{alt} \leftarrow \phi$ 
4: Initialize a timer:  $timer \leftarrow 0$ 
5: While  $KList_{alt}$  is not full AND  $timer < T_{thr}^{alt}$  Do
6:   If new signal  $s_{N_i}$  arrives Then
7:      $PK_{N_i}^{old}, PK_{N_i}^{new}, sign_{N_i} \leftarrow DEC(s_{N_i}, SK_{N_i}^{EN})$ 
8:     If  $VERIFY(sign_{N_i}, PK_{N_i}^{old} || PK_{N_i}^{new}, PK_{N_i}^{old}) = true$  Then
9:       Collect all data  $DT_{PK_{N_i}^{old}}$  on  $PK_{N_i}^{old}$ 
10:      Evaluate the latest trust  $TV'_{N_i,t}$  on  $PK_{N_i}^{old}$  with  $\{DT_{N_i}\}$ 
11:       $TV_{N_i,t} \leftarrow \lfloor TV'_{N_i,t} \cdot N_{TV} \rfloor / N_{TV}$ 
12:       $DT_{PK_{N_i}^{old}}^H \leftarrow \{H(dt) | dt \in DT_{PK_{N_i}^{old}}\}$ 
13:       $\pi_{DT} \leftarrow \pi_{DT} \cup \{DT_{PK_{N_i}^{old}}^H\}$ 
14:       $KList_{rev} \leftarrow KList_{rev} \cup \{PK_{N_i}^{old}\}$ 
15:       $KList_{alt} \leftarrow KList_{alt} \cup \{PK_{N_i}^{new} || TV_{N_i,t}\}$ 
16:    End If
17:  End If
18: End While
19: For each trust value  $TV_l$  Do
20:   Set  $n_l$  as the number of keys with trust  $TV_l$  in  $KList_{alt}$ 
21:   If  $n_l < N_l^{tv}$  Then
22:     For  $1 \leq k \leq N_l^{tv} - n_l$  Do
23:       Select a fake key  $PK_{N_k}^{old}$  with trust  $TV_l$ 
24:       Generate a random  $PK_{N_k}^{new}$ 
25:        $KList_{rev} \leftarrow KList_{rev} \cup \{PK_{N_k}^{old}\}$ 
26:        $KList_{alt} \leftarrow KList_{alt} \cup \{PK_{N_k}^{new} || TV_l\}$ 
27:     End For
28:   End If
29: End For
30:  $KList_{rev} \leftarrow Perm_{\gamma_1}(KList_{rev})$ 
31:  $KList_{alt} \leftarrow Perm_{\gamma_2}(KList_{alt})$ 
32: Return  $\pi_{DT}, KList_{rev}, KList_{alt}$ 

```

requesters can use the altered keys for MCS activities.

4.3 Anonymous Authentication on Trust

When a node N_i has a valid public/private key pair (PK_{N_i}, SK_{N_i}) , it can participate in MCS activities and prove to others its trust value. There are several cases where a node needs to prove its trust value, such as when a worker applies to conduct a certain task. We take this case as an example to show how anonymous authentication on trust is performed. When a worker N_i wants to prove its trust value to an end user N_j to apply for the task published by N_j , it simply hashes the content of the task con , concatenates it with its public key PN_{N_i} , signs on it. Then N_i sends the signed message $H(con) || PK_{N_i} || SIGN(H(con), SK_{N_i})$ to N_j .

With the received message, N_j can verify the trust of N_i with the following procedure. N_j first verifies the signature with PK_{N_i} . If the signature verification holds, it queries the latest trust value corresponding to PK_{N_i} from BC . It will check whether PK_{N_i} is involved in any revoked key list. If not, it further checks whether PK_{N_i} is involved in any altered key list and if yes it gets the trust of PK_{N_i} . Finally, BC returns the latest trust value to N_j .

4.4 Blockchain-Based Trust Evaluation

Our scheme involves a trust evaluation algorithm, which is invoked by EN_{CS} to evaluate trust of key alteration

requestors. We comprehensively take into account the subjective feedback and objective behaviors of MCS nodes and propose two blockchain-based trust evaluation methods, i.e., *Feedback-based Trust Evaluation* and *Behavior-based Trust Evaluation*. Among them, the feedback-based trust evaluation evaluates node trust according to feedback information, while behavior-based evaluation detects the honest or malicious behaviors of MCS nodes with a series of pre-trained behavior patterns and determines the trust with the detection results. The trust of a node is generated by aggregating the above two trust evaluation results together. We will detail the two methods as well as the aggregation method of the two evaluated trusts as below.

4.4.1 Feedback-Based Trust Evaluation

A node in our scheme can act as different roles, i.e., miner, worker, or end user. Considering the different requirements and responsibilities of these roles, the trust of a node as one role cannot represent its trust value of another role. Therefore, the feedback-based trust evaluation is evaluated based on role r by considering the corresponding feedback collected from relevant nodes.

A robust trust evaluation scheme should be able to resist bad-mouthing attack and ballot stuffing attack to alleviate the negative feedback of dishonest feedback. Therefore, in our scheme, for each feedback, we first calculate its deviation from the average feedback and generate a weight as well to balance its impact on the final result. We should note that accuracy of feedback is also affected by a series of other factors, including task attributes (e.g., task quantity and its average payment) and node attributes (e.g., node trust, worker expertise and experience). For example, a worker with expertise and rich experience on a task can judge the task execution quality better than those with low expertise and poor experience. Therefore, we further take advantage of these knowledge (e.g., task quantity, the transaction volume, and the behavioral trust value, etc.) to calculate a weight of each feedback. Based on the above idea, we design the formula for feedback-based trust evaluation as below:

$$TV_{j,r,k} = \frac{e^{\frac{-|k-k_i|}{\tau}}}{e^{\frac{-|k-k_i|}{\tau}} + 1} TV_{j,r,k_j} + \frac{1}{(e^{\frac{-|k-k_i|}{\tau}} + 1) \cdot O} \sum_{i,s}^{I,S} \omega_{i \rightarrow j,r,id} \cdot TE_{i \rightarrow j,r,id} \cdot (1 - dv_{i,j,r}) \quad (3)$$

where $dv_{i,j,r} = |TE_{i \rightarrow j,r,id} - \frac{\sum_{i,s}^{I,S} TE_{i \rightarrow j,r,id} \cdot \omega_{j,r,id}}{\sum_{i,s}^{I,S} \omega_{i \rightarrow j,r,id}}|$ is the difference between the individual feedback $TE_{i \rightarrow j,r,id}$ of node N_i to role r of node N_j for the task id and the mean of feedback, $O = \sum_{i,s}^{I,S} \omega_{i \rightarrow j,r,id} (1 - dv_{i,j,r})$. $\omega_{i \rightarrow j,r,id}$ is a feedback weight. In our scheme, we use the latest trust of a node as the weight of its feedback. This is reasonable since trust is a comprehensive concept that indicates the expertise, honesty, experience and other properties of a node. Therefore, it can reveal the trustworthiness of the feedback generated by the node.

Apart from the feedback-related information, past trust values also affect current trust evaluation, but the effect decays over time. We use a parameter τ to control this decay.

$|k - k_i|$ is the gap between the current block k and the block k_i that records the previous trust value, which affects the degree of decay. The trust value contributed by the node feedback is calculated according to Formula. 3.

Algorithm 2 Behavior-Based Trust Evaluation Algorithm

Input: Bt, Bd, BE_j
Output: $TB_{j,r,k}$

```

1: For all behaviors in  $BE_j$  Do
2:   If behavior  $b \in BE_j$  matches pattern  $pt \in Bt$  Then
3:     Get the weight  $w_{pt}$  of  $pt$ 
4:      $P \leftarrow P + w_{pt}$ 
5:   Else If  $b \in BE_j$  matches a pattern  $pd \in Bd$  Then
6:     Get the weight  $w_{pd}$  of  $pd$ 
7:      $NP \leftarrow NP + w_{pd}$ 
8:   End If
9: End For
10:  $TB_{j,r,k} \leftarrow \frac{P}{P+NP+1}$ 
11: Return  $TB_{j,r,k}$ 

```

4.4.2 Behavior-Based Trust Evaluation

The feedback-based trust cannot fully guarantee trustworthy trust evaluation since malicious nodes may compromise or collude with a number of nodes to increase its own trust or maliciously decrease the trust of others. To tackle this problem, we design another trust evaluation algorithm, i.e., behavior-based trust evaluation, to work together with feedback-based trust evaluation. It directly evaluates the behaviors of nodes and issues a low behavior-based trust value to those nodes that behave maliciously. In this way, even if malicious nodes could get a high feedback-based trust, its low behavior-based trust prevents it from getting high aggregated trust since we use the behavior-based trust to weight the feedback-based trust.

The behavior-based trust evaluation first trains a series of trusted/untrusted behavior patterns through machine learning. Then according to the behavior data recorded in BC and their matching with these patterns (e.g., classification of different kinds of behaviors), it accumulates the weights of successful matching and performs the trust evaluation based on objective behaviors. The trust contributed by the behaviors is measured by the matching score. Obviously, a higher matching score means a higher trust. Initial trust behavior patterns (such as providing task execution results on time and results are trustworthy, correctly processing collected data, paying on time, etc.) and untrusted behavior patterns (such as the task execution results are not provided on time or the collected data is distorted, incorrectly processing collected data, reasonable remuneration unpaid, etc.) are generated by tagged supervised machine learning classification and analysis based on a real-world behavioral data set and then further updated by the newly accumulated behavior records and confirmed trust values in BC .

One concrete implementation method is we use two weighted Bloom Filters Bt and Bd to store trusted behavior patterns and untrusted behavior patterns. BE_j denotes a set of behaviors of node N_j that are collected after block $k - 1$'s generation, $TB_{i,r,k}$ denotes the behavior trust of node N_i with the role r (miner, end user, of worker) in block k , then the evaluation of $TB_{i,r,k}$ is summarized in Algorithm 2.

4.4.3 Aggregated Trust Generation

The results of $TV_{j,r,k}$ and $TB_{j,r,k}$ are aggregated and normalized to obtain the final trust evaluation result. One aggregation formula is shown in Formula (4).

$$T_{j,r,k} = w_1 \cdot TV_{j,r,k} + w_2 \cdot TB_{j,r,k} \quad (4)$$

where $w_1 + w_2 = 1$.

Here we present the method to set up the values of w_1 and w_2 . Obviously, the effectiveness of feedback-based trust evaluation is influenced by the proportion of honest nodes related to the MCS task. If most of them are dishonest or corrupted by the adversary, the evaluated trust value will deviate from its real value because most of feedback is inaccurate. Nevertheless, the proportion of honest nodes has less influence on the behavior-based trust evaluation. Therefore, we consider w_1 and w_2 to be dynamic. We first decide a threshold θ that is used to estimate the number of honest nodes according to their latest evaluated trust. Given a group of N nodes, suppose the number of nodes whose trust is over θ is n_h , then we apply the Sigmoid function to model the impact of n_h/N on w_1 since directly using linear modeling cannot properly adjust the value of w_1 . For example, when 30% of nodes are dishonest, the feedback-based trust evaluation can still achieve relatively high accuracy. In this case, we do not need to assign a low weight to feedback-based trust. Thus, we have:

$$\begin{aligned} \omega_1 &= \frac{1}{1 + e^{-\alpha_1(n_h/N - \alpha_2)}} \\ \omega_2 &= 1 - \omega_1 \end{aligned} \quad (5)$$

Where α_1 and α_2 are adjusting parameters that that respectively determine the increase radio of ω_1 with the change of n_h/N and control when ω_1 can exceed 0.5. They are system parameters and can be set ahead of time. In this way, when the proportion of honest nodes is low, ω_1 will be small, and hence it has little negative impact on trust evaluation. The trust value is between $[0, 1]$, 0 for fully untrusted, and 1 for fully trusted. In the initial block, we set all nodes with a trust value of 0.001 for all roles and the task record of a node is empty. The above initial configuration helps overcoming white washing attack because the trust value will start from zero if a node re-registers with a new identity unlinked to any old ones.

5 SECURITY ANALYSIS

We evaluate the security of our scheme with the metrics summarized below:

- Unforgeability:** All nodes except the enclave in the cloud server cannot forge a valid key alteration notice (i.e., $KList_{rev}$ and $KList_{alt}$ issued by the enclave of the cloud server).
- Correctness:** Correctness requires (1) each new public key in $KList_{alt}$ should be attached with a trust value that is correctly calculated by EN_{CS} ; (2) for each key in $KList_{rev}/KList_{alt}$, the corresponding new/old public key must exists in $KList_{alt}/KList_{rev}$.
- Confidentiality on Trust-Related Data:** This metric requires that the trust evaluation does not reveal any information of trust-related data.

- d) **Anonymity and Unlinkability:** Anonymity and Unlinkability refers to that no node including the cloud server can reveal the real identity of MCS nodes, and they can neither link the old key of an MCS node with its new one. This property is important to ensure node identity privacy and prevent malicious tracking by attackers.
- e) **Resistance to Attacks on Trust Evaluation:** The trust evaluation can eliminate the negative impact of bad mouthing attack and white washing attack.
- f) **Resistance to Trust Analysis:** The adversary cannot link the different public keys of any node with its trust.

5.1 Unforgeability

It is obvious that our scheme achieves unforgeability since the key alteration notice is generated and signed by the SGX enclave in the cloud server. The SGX enables everyone to verify the correctness of the enclave deployment and its output. Based on the security assumption of SGX, no entity is able to forge an output of SGX enclave that can successfully pass verification. Therefore, our scheme achieves unforgeability.

5.2 Correctness

The correctness of our scheme holds if the data provided to Algorithm 1 are correct and Algorithm 1 is correctly executed. Algorithm 1 is executed by the trusted SGX enclave EN_{CS} set up by the cloud server CS , which ensures the trusted execution of the key alteration algorithm. Therefore, to result in incorrect altered key lists output by EN_{CS} , the only way is to provide false data to EN_{CS} .

The Algorithm 1 requires two types of data, i.e., trust-related data from S_{str} and key alteration signal s_{N_i} sent from key alteration requester N_i . Since when publishing $KList_{rev}$ and $KList_{alt}$, EN_{CS} will publish the hash list of trust-related data as well, of which the correctness can be verified by every node by checking the corresponding hash value in BC . If the trust-related data provided to EN_{CS} is false, it can be detected by every node and the key alteration result will not be recorded in BC . As a result, for any key alteration result that passes the verification of miners, its trust-related data for trust evaluation are correct.

For each signal s_{N_i} , it includes its both old public key and new public key. The public key based signature mechanism guarantees its validity and integrity. Therefore, others including CS are unable to modify its content because EN_{CS} can easily detect the modification with the signature. This ensures the correctness of the key alteration signals.

In summary, our scheme ensures the algorithms for the key alteration processing and the trust evaluation are correctly executed, and all data provided to EN_{CS} for key alteration signal processing and trust evaluation are correct, and the key alteration signals and trust-related data for them are correct. Therefore, our scheme achieves correctness.

5.3 Confidentiality on Trust-Related Data

All trust-related data are stored in S_{str} in ciphertext. When EN_{CS} conducts trust evaluation, it needs to request decryption key from the owners of the trust-related data, and the decryption key is transmitted via a secure channel

and attackers cannot obtain the content of the decryption. Then, EN_{CS} decrypts the trust-related data and processed in a secure way. Given the confidentiality and security of the SGX enclave, attackers cannot infer any information on the trust-related data by compromising EN_{CS} . Therefore, data confidentiality holds in data storage, transmission, and processing, and our trust evaluation scheme can achieve confidentiality on trust-related data.

5.4 Anonymity and Unlinkability

Obviously, our scheme achieves anonymity since our scheme leverages the hash of a periodically altered public key as identifier rather than the real identity of an MCS node. Therefore, it is impossible for attackers to obtain the real identity of an MCS node. For unlinkability, an attacker is able to obtain the plain revoked key lists, altered key lists, and encrypted key alteration signals. Therefore, it can infer the possible linkage between old keys and new keys in two ways: 1) analyzing the revoked key list and the corresponding altered key lists and 2) analyzing the encrypted key alteration signals. We prove that the attacker cannot effectively link the old key and its corresponding new key in either way as below:

Proof: We first prove it is ineffective to disclose linkage relationship between old and new keys by analyzing the revoked key list and the corresponding altered key lists. Since public keys in the revoked key list and altered key list are permuted with two permutation functions $Perm_{\xi_1}(\cdot)$ and $Perm_{\xi_2}(\cdot)$, given the randomness of the permutation functions, any node including the cloud server is unable to infer the linkable relationship between the old keys and the new keys with the location of public keys in the list. Besides, in case the attackers infer any information from the difference of trust attached to keys, we set up a threshold N_l^{tv} for each trust value, and for each trust value TV_l , the list issued must contain more public keys with the trust TV_l than the threshold N_l^{tv} . If the number of public keys with trust value TV_l is smaller than N_l^{tv} , EN_{CS} will generate fake keys and insert them into the list. To summarize, the attackers cannot gain any information on the linkable relationship between old and new key pairs but the trust, while there exist at least N_l^{tv} for trust value TV_l . Therefore, the probability to guess the linkable relationship between the old and new key pairs is $1/N_l^{tv}$.

We next prove that the adversary can obtain no extra information by analyzing ciphertext of key alteration signal. The proof relies on the security of the encryption algorithm. Specifically, we assume the encryption algorithm achieves ciphertext indistinguishability under chosen plaintext attack (IND-CPA). We also assume the adversary $\mathcal{A}$ can obtain plain signatures in the key alteration signals, though in reality, the plain signatures in key alteration signals are protected by encryption. However, we assume $\mathcal{A}$ cannot infer the linkage between the old and new keys with the corresponding signature and the signature is a random from the view of $\mathcal{A}$. The assumption is reasonable because our scheme can achieve it easily by letting EN_{CS} issue a unique secret r for each key alteration signal generator. When a node N_i generates a key alteration signal, it generates a signature on $H(PK_{N_i}^{old} || PK_{N_i}^{new} || r)$ rather than

$H(PK_{N_i}^{old} || PK_{N_i}^{new})$. In this way, even $\mathcal{A}$ obtains the plain signature, it cannot infer the linkage between old key and new key with the signature since the signature involves a random r , which differs for each key alteration signal.

Based on the above assumptions, we define our analysis model between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{C}$ plays as a node. We enable the $\mathcal{A}$ to access the plaintext of signatures in key alteration signals. We also introduce a random oracle $\mathcal{O}$, which responds the queries from $\mathcal{A}$ with a random selected value as signature. With these settings, we design **Game 1** as below.

- i) **Setup:** $\mathcal{C}$ sets up an enclave, obtain the key pairs of SGX and transmits the public key PK_{CS}^{EN} to $\mathcal{A}$.
- ii) **Querying Phase:** $\mathcal{A}$ chooses a number of old and new key pairs (PK^{old}, PK^{new}) . For each key pair, it queries a random oracle $\mathcal{O}$ for the signature sig . For each query, $\mathcal{O}$ selects a random r and returns r as sig to $\mathcal{A}$. Then the adversary $\mathcal{A}$ then sends the triple tuples $(PK^{old}, PK^{new}, sig)$ to $\mathcal{C}$, which returns the corresponding the ciphertext c to $\mathcal{A}$.
- iii) **Challenge:** $\mathcal{A}$ selects one public keys PK^{old} as old keys and two public keys PK_0^{new} and PK_1^{new} as new keys, and $\mathcal{A}$ sends PK^{old} , PK_0^{new} , and PK_1^{new} to $\mathcal{C}$. $\mathcal{C}$ selects a random $b \in \{0, 1\}$, and queries $\mathcal{O}$ for the signature sig with PK^{old} and PK_b^{new} . Then $\mathcal{C}$ encrypts the $PK^{old} || PK_b^{new} || sig$ and returns the ciphertext c to $\mathcal{A}$.
- iv) **Guess:** $\mathcal{A}$ outputs $b' \in \{0, 1\}$. If $b = b'$, $\mathcal{A}$ wins the game.

With the above game, we define the security against ciphertext analysis as below.

Definition 1: The adversary cannot obtain the linkage relationship between old and new public keys, if $\forall$ polynomial-time algorithm $\mathcal{A}$, $|Pr[\mathcal{A} \text{ wins Game1}] - \frac{1}{2}|$ is negligible.

For easy presentation, we define another game **Game 2** to present IND-CPA as below.

- i) **Setup:** $\mathcal{C}$ set up the public encryption key PK and private decryption key SK . Then, it sends PK to $\mathcal{A}$.
- ii) **Query Phase:** $\mathcal{A}$ selects a number of plaintexts $\{m_i\}$ and queries $\mathcal{C}$ for the ciphertexts. $\mathcal{C}$ encrypts all plaintexts in $\{m_i\}$ and returns ciphertext $\{c_i\}$ to $\mathcal{A}$.
- iii) **Challenge:** $\mathcal{A}$ selects two messages m_0 and m_1 , where $|m_0| = |m_1|$, and it sends them to $\mathcal{C}$. $\mathcal{C}$ selects a random $b \in \{0, 1\}$ and encrypts m_b . Then, it sends back the ciphertext c to $\mathcal{A}$.
- iv) **Guess:** $\mathcal{A}$ outputs b' . If $b' = b$, $\mathcal{A}$ wins the game.

Given the above game, we defines IND-CPA with *Definition 2*.

Definition 2: Attackers cannot infer the linkage relationship with the ciphertext of key alteration signal, if $\forall$ polynomial-time algorithm $\mathcal{A}$, $|Pr[\mathcal{A} \text{ wins Game2}] - \frac{1}{2}|$ is negligible.

We here give the theorem indicating that the proposed scheme is secure against ciphertext analysis in *Theorem 1*.

Theorem 1: If the encryption algorithm in our scheme achieves IND-CPA, the attackers cannot infer the linkage relationship with the ciphertext of key alteration signal.

Proof: We suppose that $\mathcal{A}$ is a polynomial-time adversary and can win the **Game 1** with a non-negligible

advantage ε . To be specific, after the Query Phase of **Game 1**, it selects PK^{old} , PK_0^{new} , and PK_1^{new} , and obtains the two signatures sig_1 and sig_2 on $H(PK^{old} || PK_0^{new})$ and $H(PK^{old} || PK_1^{new})$ respectively. Then $\mathcal{C}$ selects a $b \in \{0, 1\}$, encrypts $PK^{old} || PK_b^{new} || sig_b$, and returns the ciphertext c . Then $\mathcal{A}$ can win then game with P_1 , where $|P_1 - \frac{1}{2}| = \varepsilon$.

Next, we prove that $\mathcal{A}$ can also win **Game 2** with non-negligible advantage. Specifically, in the Querying Phase of **Game 2**, it constructs the plaintext m with the following method. It selects two public keys PK_1 and PK_2 and obtains the signatures sig via the random oracle $\mathcal{O}$. Then it sets the message $m = PK_1 || PK_2 || sig$. In the Challenge Phase, it selects three public keys, PK_0 , PK_1 , and PK_2 and queries $\mathcal{O}$ for the signatures sig_0 and sig_1 on $H(PK_0 || PK_1)$ and $H(PK_0, PK_2)$. Then it sets $m_0 = PK_0 || PK_1 || sig_0$ and $m_1 = PK_0 || PK_1 || sig_1$ and sends them to $\mathcal{C}$ for encryption.

Since we employ the same encryption algorithm in both games, we could see that the challengers $\mathcal{C}$ in **Game 1** and **Game 2** are the same. Besides, we could also see that $\mathcal{A}$ conduct the same queries and challenges in both games. Therefore, it $\mathcal{A}$ can win **Game 1** with advantage ε , it can win **Game 2** with advantage ε as well. However, it conflicts with the assumption that the encryption achieves IND-CPA. Therefore, the polynomial-time adversary $\mathcal{A}$ can only win the **Game 1** with negligible advantage.

5.5 Conditional Traceability and Non-Repudiation

Our scheme requires that each valid message should contain a signature, and hence, a node cannot deny it has sent a message it signs. Besides, our scheme can be extended to support conditional traceability with some simple design. First, each node should register its real identity to an enclave established by a cloud server. Then, the enclave verifies the identity and issues an initial public/private key pair to the node. The enclave records the public key in blockchain. When dispute happens, for example, N_x detects the dishonest or malicious behavior of N_y , N_x generates a request on the dispute, which includes the current public key PK_{N_y} of N_y , the current public key PK_{N_x} of N_x , the proof π_m that proves the misbehavior of N_y . Then it generates the signature sig_{N_x} on the request with its private key and sends the request to blockchain for recording.

Here we assume an enclave maintains the key alteration records of all users that have sent key alteration signals to the enclave. An enclave, identified by EN_{CS} , if detects the request by N_x , will check the correctness of π_m . If the proof passes the verification, EN_{CS} checks all key alteration records related to PK_{N_y} , and finally, EN_{CS} obtains the whole key alteration history of N_y as well as N_y 's real identity. In case that a user register itself or alter its public keys to different enclaves set up by different cloud servers, the enclaves can exchange key alteration records about N_y to trace N_y 's key alteration history and disclose its real identity. Since the disclosure of N_y 's identity only when it conduct malicious behavior, our scheme achieves conditional traceability.

5.6 Resistance to Attacks on Trust Evaluation

Based on our security model, our trust evaluation algorithm should be able to resist three attacks, i.e., bad mouthing

attack, ballot stuffing attack, and white washing attack. In this part, we analyze its resistance to the above attacks.

Resistance to bad mouthing attack/ballot stuffing attack: For each feedback generated, it is weighted with its deviation to the average value of all pieces of feedback. In this way, the dishonest feedback is attached with a small weight to reduce its impact on the feedback-based trust evaluated. Besides, the final trust is the aggregation of feedback-based trust and behavior-based trust, while the latter is not related to node feedback, which also helps resisting bad mouthing attack as well as ballot stuffing attack.

Resistance to white washing attack: For white washing attack, our scheme set a very low trust value (i.e., 0.001) as initial trust. Therefore, a malicious node cannot obtain high trust by re-registering as a new node. Besides, white washers are requested to pay for key registration. Frequently registering keys will incur high expense for malicious nodes. In this way, our scheme can well prevent the white washing attack.

5.7 Resistance to Trust Analysis

Our scheme can resist trust analysis by mapping node trust value into several discrete trust levels. In this way, though an adversary can obtain a certain information of a node with its trust level, since there are a number of nodes with the same trust level, the adversary cannot distinguish a single node from the nodes with the same trust level. Besides, for each trust level, an SGX enclave, when publishing a key alteration list, SGX will involve a certain number of nodes with this trust level in the list, which also preventing to link revoked public key with its corresponding altered key of any node. In summary, our scheme can well prevent identity disclosure and public key tracking caused by trust analysis.

5.8 Resistance to $(n - 1)$ attack

Our scheme can resist $(n - 1)$ attack by enabling the SGX enclave of the cloud server to insert fake keys into both key alternation list and key revocation list even when the number of altered public keys reach the threshold. To be specific, for any trust level TV , the SGX enclave will always insert another N_s^{tv} fake keys into the two lists respectively. In this way, even the key alternation only include 1 altered public key generated by honest MCS user while the remaining $N_s^{tv} - 1$ keys are generated by the adversary, there are N_s^{tv} fake keys, which makes the adversary can only distinguish the real altered public key of the honest user with a probability of $\frac{1}{N_s^{tv} + 1}$.

6 EXPERIMENTAL EVALUATION

This section reports the experimental results of both key alteration and trust evaluation. We conducted three experiments, i.e., *experiment on performance of authentication on trust*, *experiment on trust evaluation with simulations*, and *experiment on trust evaluation with a real-world dataset*.

6.1 Simulation Setup

6.1.1 Experimental Methodology and Environment

We applied simulations to evaluate the performance of the authentication on trust scheme and the trust evaluation.

TABLE 2: Experimental Settings

	Parameter	Setting
Exp. 1	MCS Node Number	100
	Cloud Server Number	1
Exp. 2	Dishonest Node Number	$n(0 \leq n \leq 100)$
	Normal Node Number	$\frac{2}{3}(100 - n)$
	Honest Node Number	$\frac{1}{3}(100 - n)$
	Real Trust of Nodes	$\begin{cases} \text{Dishonest} : [0, 0.3) \\ \text{Normal} : [0.3, 0.7) \\ \text{Honest} : [0.7, 1] \end{cases}$
	$\omega_{i \rightarrow j, r, k}$ ω_1	$TV_{i, r, k} \frac{1}{1 + e^{-3.6(\frac{n_h}{N} - 0.45)'}}$ <p>where n_h is the number of nodes whose evaluated trust is over θ, and θ is the upper bound of dishonest nodes' evaluated trust</p>
Exp. 3	ω_2	$1 - \omega_1$
	Initial Trust	0.001
	Node Number (Part 1)	20
	Node Number (Part 2)	30
	$\omega_{i \rightarrow j, r, k}$ ω_1, ω_2	0.001 0.5

This is because real experiments in MCS are difficult to organize. For another, for trust evaluation, simulation enables us to evaluate our trust evaluation scheme in a controlled environment. It enables us to adjust particular parameters to observe their impact on trust evaluation [21], [51].

All the three experiments were conducted in a laptop running 64-bit Ubuntu 18.04 with 1.6-GHz Intel Core i5 Quad-CPU and 8G RAM. The experiment about authentication on trust was conducted in C++ Language with Intel Integrated Performance Primitives (IPP) Cryptography Library and Intel SGX SDK in a simulation mode, and the two experiments about trust evaluation were conducted in Python Language.

Extended Epinions Dataset: We used the Extended Epinions dataset to test trust evaluation performance. This dataset was given directly by Epinions staff to Paolo Massa and has been widely used in the research of reputation or trust system [52], [53]. This dataset contains the distrust lists (which users are distrusted by which users) that are not shown publicly, but kept private. It consists of 841,372 rating statements on other users from 132,000 users, among these users there are around 85,000 users that receive at least one statement. Besides, it also contains 3,024,664 statements on articles from users. We used the Extended Epinions dataset for two reasons. First, it consists rating statements between users, which can be used as the feedback among MCS nodes. Second, the rating statements on articles from users can be used to simulate the performance of MCS task execution.

6.1.2 Experimental Settings

We summarize the experimental settings in Table 2

a) Settings of Exp. 1: This experiment aims to test the efficiency of our scheme of anonymous authentication on trust. We simulated 100 MCS nodes and 1 cloud server and tested

the operation time of four key algorithms in our scheme, i.e., *Key Alteration Signal Generation*, *Key Alteration Signal Processing*, *Signature Generation*, and *Authentication on Trust*. Since there are only a few of anonymous authentication schemes for MCS, and ZebraLancer of one of the works that achieves both anonymous authentication and accountability. Therefore, our scheme is compared with ZebraLancer in efficiency [36].

b) Settings of Exp. 2: We setup an experiment to test the trust evaluation accuracy and its resistance to bad mouthing attack and allot stuffing attack. We set up 100 nodes, which includes n dishonest MCS nodes, where n varies from 0 to 100. The number of honest nodes is set as $\frac{1}{3}(100 - n)$. We will specify the difference between different types of nodes in the next paragraph. All nodes participate in 100 tasks, and in each task, each node generates 20 behaviors that are used to evaluate their behavior-based trust. Besides, they each generate feedback on the others. In this experiment, we first conducted feedback-based trust evaluation only on these nodes to test its performance, and then we tested the whole trust evaluation scheme.

We herein present the behavior model of these nodes. Specifically, we assume that there a real trust for each node whose value scope is $[0, 1]$ and the real trust of a node keeps unchanged, and each node behaves based on its real trust. All nodes are categorized into three types based on their real trust, as shown in Table 2. We consider that the honest nodes are those who are honest and with high expertise and abundant experiences, and they behave honestly, generate reliable contributions, and evaluate other nodes accurately; the normal nodes are those who are honest but lack expertise or experiences, and hence they behave honestly but may generate unreliable contributions or feedback; the dishonest nodes are dishonest in terms of their expertise and may behave maliciously with high probability.

We further specify the parameters of our scheme. First, the initial trust of all nodes when the experiment starts is set as 0.001. Second, we set the weight $\omega_{i \rightarrow j, r, id}$ for feedback-based trust evaluation as $TV_{i, r, k}$. This is because in MCS, trust is a concept that comprehensively considers the honesty level, expertise, experience, etc., and all of them affects the accuracy of the generated feedback. Therefore, it is reasonable to employ trust of a node to determine the weight of its feedback. Third, we set the weight ω_1 and ω_2 with a parameter $\theta \in (0, 1)$. More specifically, when conducting feedback-based trust evaluation, suppose the number of evaluators whose current trust is higher than θ is n_h and the number of all evaluators is N , then we set $\omega_1 = \frac{1}{1 + e^{-\alpha_1(n_h/N - \alpha_2)}}$ and $\omega_2 = 1 - \omega_1$. We call θ the upper bound of dishonest nodes. We design like this because when dishonest evaluators occupy a large proportion of evaluators, the evaluated feedback-based trust may greatly deviate from the real trust. Therefore, we use the proportion of honest nodes to adjust the weight of feedback-based trust. However, in reality, it is challenging to get the a node's real trust. Considering the inaccuracy of trust evaluation in some cases, we set θ as a relatively high, i.e., 0.5. Additionally, we learned the optimal values of the other parameters from historical experimental results and set $\alpha_1 = 3.6$ and $\alpha_2 = 0.45$.

Settings of Exp. 3: We conducted an extra experiment

Fig. 4: Operation time of key alteration signal generation/processing. Fig. 5: Operation time of signature generation and authentication on trust.

on trust evaluation based on the Extended Epinions dataset. We considered the users' trust statements on other users as their feedback on others. Besides, we simulated the execution of several tasks by considering the rating statement on each article as the contribution of a worker. The procedure of this experiment is as follows. We first clustered 1,300,000 rating statements on different articles into several clusters and then marked whether a cluster is composed of honest statements or dishonest statements by checking the average deviation of each statement from the mean.

After dividing the selected statements into several honest or malicious clusters, we trained a classifier that can determine the most possible cluster a new statement belongs to. Then we simulated several groups of nodes in testing since the dataset contains no label indicating whether a statement on an article is honest. Each simulated group contains 20 nodes, and the percentage of malicious behavior (denoted as p) is 0, 20%, 40%, 60%, 80%, and 100%. Each node has 100 rating statements on some articles. We used the trained classifier to check the behaviors of these nodes and then employed the proposed behavior-based trust evaluation to evaluate their trust.

Apart from the behavior-based trust evaluation, we also conducted the full trust evaluation based on the Extended Epinions dataset. We randomly selected 100 nodes that are not inside the training set, and used the behavior patterns obtained to evaluate their behavior-based trust. Then for each node we selected 100 trust/distrust statements from other users on them to calculate their feedback-based trust. However, the number of statements on some of the selected nodes is less than 100. To address this issue, we generated random values varying from 0.3 to 0.7 as statements, which implies that other nodes are uncertain on them.

In this experiment, we set $\omega_1 = \omega_2 = 0.5$ and $\omega_{i \rightarrow j, r, k} = 0.001$ for all nodes. This is because in the Extended Epinions dataset, each user has at most one statement on another user. To solve this problem, we cannot obtain the previous trust of theirs with the trust evaluation scheme proposed in this paper. Therefore, we attach equal importance to feedback-based trust and behavior-based trust, and set the trust of each node as the initial trust, i.e., 0.001.

6.2 Results of Exp. 1

We first present the efficiency testing results of key alteration signal generation/processing. Fig. 4 plots the operation time with regard to the number of key alteration signals of key alteration signal generation by MCS nodes and key

Fig. 6: Experimental results on feedback-based trust evaluation with regard to the percentage of dishonest nodes

alteration signal processing by the cloud server. From the figure we can see that the operation time of both algorithms is linear to the number of key alteration signals. Besides, the average operation time of the two operations is $0.842ms$ and $0.965ms$, respectively. Therefore, the proposed key alteration is quite efficient and will not introduce much computation to either the key alteration requester or the cloud server.

Fig. 5 plots the operation time with regard to the number of messages of signature generation and authentication on trust. The operation time of both algorithms is linear to the number of messages. To further illustrate the superiority of our scheme, we compare it with ZebraLancer [36], which introduces a centralized authority and employs zero-knowledge proof for anonymous authentication. ZebraLancer costs around $63s$ to generate a proof for anonymous authentication, and it takes at least $6.2ms$ to verify the validity of the proof. While in our scheme, a node just needs to sign the message, and others can verify its trust by verifying the signature and querying the trust value based on its public key from BC . The average time of signature generation is $0.885ms$, and the average time of authentication on trust is $0.972ms$, which is much more efficient than ZebraLancer.

6.3 Results of Exp. 2

This subsection shows the simulation results of trust evaluation. Besides, we tested the accuracy of our trust evaluation scheme under different percentages of dishonest nodes to evaluate its resistance to attacks. We also compare its performance with the scheme in [51]

6.3.1 Experiment on Feedback-based Trust Evaluation

We first evaluated the performance of the trust evaluation scheme by comparing the evaluated trust of a node and its real trust. Fig. 6 depicts the trust evaluation results when the numbers of honest nodes, normal nodes, and dishonest nodes are 30, 40, 30, respectively. From the figure, in this case, we can see that the evaluated trust of all nodes is quite close to their real trust, which proves the effectiveness of our feedback-based trust evaluation.

We further tested the average deviation from each node's feedback-based trust to their real trust under different percentage of dishonest nodes. The detailed experimental settings can refer to Table. 2 and Section 6.1.2. We also implemented the trust evaluation scheme in [51] under the

Fig. 8: Experimental results on aggregated trust evaluation with regard to the percentage of dishonest nodes

same setting. Fig. 7 depicts the comparison results. We can see that when the proportion of malicious nodes is lower than 50%, our scheme outperforms the scheme in [51]. However, when the percentage of malicious nodes is beyond 50%, the work in [51] achieves lower deviation than our work. This is mainly because in this case, the malicious nodes almost dominate the whole system, and they can maliciously generate negative feedback to honest nodes and increase their own trust. As a result, the trust is greatly deviated from node real trust, and cannot reveal the honesty or expertise of nodes. Therefore, our scheme suffers from poor performance. From this test, we can see that the trust evaluated only based on feedback has drawbacks, and thus it is essential to employ behavior-based trust evaluation to overcome this weakness.

6.3.2 Experiment on Trust Evaluation Based on Feedback and Behaviors

In this experiment, we implemented the complete trust evaluation scheme with the same experimental settings. From the results depicted in Fig. 7, We are aware that the feedback-based trust evaluation cannot ensure trust accuracy, especially the number of malicious nodes in the evaluator group is high. Therefore, we introduce the behavior-based trust evaluation to make complement to the feedback-based trust evaluation. Fig. 8 presents the aggregated trust of each node when the number of honest nodes, normal nodes, and dishonest nodes are 30, 40, 30, respectively. We can see that combining the two trust evaluation schemes together improves the trust evaluation accuracy.

Besides, we also evaluated the deviation of aggregated trust based on both node feedback and behaviors to the corresponding real trust and compare our scheme with the work in [51], as shown in Fig. 9. The figure illustrates the trust evaluation based on both node feedback and behaviors outperforms the scheme in [51] regardless of the percentage of the malicious nodes. In addition, even when the percentage of dishonest nodes is high, our trust evaluation scheme still keeps much better than [51]. The largest average deviation of our scheme is still below 0.25, which proves that our scheme can well overcome the negative impact due to dishonest nodes. This is mainly because when the percentage of malicious nodes is high, our scheme can overcome its negative impact by decreasing its weight when aggregating the two types of trust since the malicious nodes have less impact on behavior-based trust.

Fig. 10: Experimental results on behavior-based trust evaluation

Fig. 11: Experimental results on aggregated trust generation

6.4 Results of Exp. 3

In this subsection, we conduct some extra experiments on a real-world dataset, i.e., the Extended Epinions dataset [52], [53] and present the results.

6.4.1 Experimental Results of Behavior-Based Trust Evaluation

We mainly focused on the false result submission attack in MCS, where malicious nodes submit false data to interfere the discovery of real data truth. We simulated the execution of MCS tasks by considering each article in the Extended Epinions dataset as an MCS task and considering ratings from various users as the task execution results. Our experiment clusters the selected 1,300,000 statements into several clusters and check whether the comments in each cluster is honest or not. Then, we trained a pattern for each cluster to check whether a new rating statement outside the 1,300,000 statements is honest or malicious.

Fig.10 depicts the experimental results, from which we can see that the nodes with a lower p have higher trust. For those who always behave maliciously, the behavior-based trust is almost zero. Therefore, our scheme can effectively detect malicious behaviors of MCS nodes, and the behavior-based trust can well reveal the credibility of node behaviors.

6.4.2 Experimental Results of Trust Evaluation on Node Feedback and Behaviors

In the last experiment, we tested both the feedback-based trust evaluation and the behavior-based trust evaluation on the Extended Epinions dataset.

Fig. 11 illustrates the experimental results. We can see that the feedback-based trust of some nodes greatly deviates from their behavior-based trust. This demonstrates that feedback-based trust or behavior trust cannot individually reveal the real trust value of nodes in a good way. In practice, it is possible that the dishonest behaviors of malicious nodes cannot be detected by others and hence the malicious nodes can gain high feedback-based trust. However, our algorithm can effectively detect malicious or dishonest behaviors of a node, and thereby it cannot gain high aggregated trust. In summary, the feedback-based trust and behavior-based trust is complementary to each other, and the aggregated trust can well improve the accuracy of trust evaluation.

7 CONCLUSION

In this paper, we leveraged the Intel SGX to design a novel scheme to anonymously authenticate trust in MCS based

on blockchain. The proposed scheme enables MCS nodes to verify the trust of others in an anonymous way. We proposed a key management scheme that enables the nodes to alter their key pairs and link the new key pair with the current node trust value in a secure and unlinkable way with the assistance of an SGX enclave. Besides, we develop a robust trust evaluation algorithm that aggregates both subjective feedback and objective behaviors of MCS nodes, which can resist some potential attacks. We conducted a series of security analyses and experiments. The analysis and experimental results illustrate the security and efficiency of the proposed scheme. In our future work, we will further study context-aware trust evaluation in MCS for achieving better accuracy and flexibility. Based on it, we will step ahead to develop a context-aware trust predication scheme in decentralized MCS in order to help decision making for different task execution.

ACKNOWLEDGEMENT

The work is supported in part by the National Natural Science Foundation of China under Grants 61672410 and 61802293, the Academy of Finland under Grants 308087, 314203 and 335262, the National Postdoctoral Program for Innovative Talents under grant BX20180238, the Project funded by China Postdoctoral Science Foundation under grant 2018M633461, the Key Lab of Information Network Security, Ministry of Public Security under grant No. C18614, the open grant of the Tactical Data Link Lab of the 20th Research Institute of China Electronics Technology Group Corporation, P.R. China under grant CLDL-20182119, the Shaanxi Innovation Team project under grant 2018TD-007, and the 111 project under grant B16037.

REFERENCES

- [1] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, and Y. T. Hou, "A survey on security, privacy, and trust in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2971–2992, 2017.
- [2] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and 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*, ser. MobiSys '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 179–191. [Online]. Available: <https://doi.org/10.1145/2906388.2906420>
- [3] B. Ko?ucu, B. Arnrich, and C. Ersoy, "A crowdsourced skymap," in *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, 2014, pp. 1283–1286.
- [4] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. H. Deng, "Crowdbc: A blockchain-based decentralized framework for crowdsourcing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1251–1266, 2018.
- [5] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.
- [6] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
- [7] S. Huh, S. Cho, and S. Kim, "Managing iot devices using blockchain platform," in *2017 19th international conference on advanced communication technology (ICACT)*. IEEE, 2017, pp. 464–467.
- [8] Z. Lu, Q. Wang, G. Qu, and Z. Liu, "Bars: a blockchain-based anonymous reputation system for trust management in vanets," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 98–103.

- [9] B. Yin, Y. Wu, T. Hu, J. Dong, and Z. Jiang, "An efficient collaboration and incentive mechanism for internet of vehicles (ioV) with secured information exchange based on blockchains," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1582–1593, 2019.
- [10] L. Cheng, J. Liu, G. Xu, Z. Zhang, H. Wang, H.-N. Dai, Y. Wu, and W. Wang, "Sctsc: A semicentralized traffic signal control mode with attribute-based blockchain in iovs," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1373–1385, 2019.
- [11] W. Feng and Z. Yan, "Mcs-chain: Decentralized and trustworthy mobile crowdsourcing based on blockchain," *Future Generation Computer Systems*, vol. 95, pp. 649–666, 2019.
- [12] Z. Yan, Y. Chen, and Y. Shen, "A practical reputation system for pervasive social chatting," *Journal of Computer and System Sciences*, vol. 79, no. 5, pp. 556–572, 2013.
- [13] G. Maxwell, "Coinjoin: Bitcoin privacy for the real world," in *Post on Bitcoin forum*, 2013.
- [14] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 397–411.
- [15] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.
- [16] S. Noether, "Ring signature confidential transactions for monero," *IACR Cryptology ePrint Archive*, vol. 2015, p. 1098, 2015.
- [17] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Crowdsourcing knowledge atop open blockchain, privately and anonymously," *arXiv preprint arXiv:1803.01256*, 2018.
- [18] J. Ren, Y. Zhang, K. Zhang, and X. S. Shen, "Sacrm: Social aware crowdsourcing with reputation management in mobile sensing," *Computer Communications*, vol. 65, pp. 55–65, 2015.
- [19] K. L. Huang, S. S. Kanhere, and W. Hu, "A privacy-preserving reputation system for participatory sensing," in *37th Annual IEEE Conference on Local Computer Networks*. IEEE, 2012, pp. 10–18.
- [20] —, "Are you contributing trustworthy data?: the case for a reputation system in participatory sensing," in *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*. ACM, 2010, pp. 14–22.
- [21] H. Amintoosi and S. S. Kanhere, "A reputation framework for social participatory sensing systems," *Mobile Networks and Applications*, vol. 19, no. 1, pp. 88–100, 2014.
- [22] Z. Yan, W. Feng, and P. Wang, "Anonymous authentication for trustworthy pervasive social networking," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 3, pp. 88–98, 2015.
- [23] V. Costan and S. Devadas, "Intel sgx explained," *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.
- [24] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 619–636.
- [25] A. Baliga, "Understanding blockchain consensus models," in *Persistent*, 2017.
- [26] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.
- [27] R. Yuan, Y.-B. Xia, H.-B. Chen, B.-Y. Zang, and J. Xie, "Shadoweth: Private smart contract on public blockchain," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 542–556, 2018.
- [28] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric," *arXiv preprint arXiv:1805.08541*, 2018.
- [29] S. Jiang, X. Zhu, and L. Wang, "An efficient anonymous batch authentication scheme based on hmac for vanets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2193–2204, 2016.
- [30] L. Chen, S.-L. Ng, and G. Wang, "Threshold anonymous announcement in vanets," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 605–615, 2011.
- [31] Z. Yan, P. Wang, and W. Feng, "A novel scheme of anonymous authentication on trust in pervasive social networking," *Information Sciences*, vol. 445, pp. 79–96, 2018.
- [32] Y. Zhang, W. Liu, and W. Lou, "Anonymous communications in mobile ad hoc networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2005, pp. 1940–1951.
- [33] M. Raya and J.-P. Hubaux, "Securing vehicular ad hoc networks," *Journal of computer security*, vol. 15, no. 1, pp. 39–68, 2007.
- [34] J. Guo, J. P. Baugh, and S. Wang, "A group signature based secure and privacy-preserving vehicular communication framework," in *2007 Mobile Networking for Vehicular Environments*. IEEE, 2007, pp. 103–108.
- [35] K. A. Shim and Y. R. Lee, "Efficient identity-based ring signature scheme with anonymity and system thereof," Sep. 24 2013, uS Patent 8,543,811.
- [36] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Private and anonymous crowdsourcing system atop open blockchain," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 853–865.
- [37] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 486–504.
- [38] G. Maxwell, "Coinswap: transaction graph disjoint trustless trading (2013)," URL: <https://bitcointalk.org/index.php>.
- [39] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 345–364.
- [40] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 112–126.
- [41] Z. Yan, Y. Chen, and Y. Shen, "Percontrep: a practical reputation system for pervasive content services," *The Journal of Supercomputing*, vol. 70, no. 3, pp. 1051–1074, 2014.
- [42] Z. Yan and P. Cofta, "A mechanism for trust sustainability among trusted computing platforms," in *International Conference on Trust, Privacy and Security in Digital Business*. Springer, 2004, pp. 11–19.
- [43] Z. Yan, *Trust modeling and management in digital environments: from social concept to system development*. Information Science Reference, 2010.
- [44] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 619–636.
- [45] R. Pires, D. Gavril, P. Felber, E. Onica, and M. Pasin, "A lightweight mapreduce framework for secure processing with sgx," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 2017, pp. 1100–1107.
- [46] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "Vc3: Trustworthy data analytics in the cloud using sgx," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 38–54.
- [47] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *proceedings of the 1st Workshop on System Software for Trusted Execution*. ACM, 2016, p. 2.
- [48] Y. Ma, Y. Wu, J. Ge, and L. Jun, "An architecture for accountable anonymous access in the internet-of-things network," *IEEE Access*, vol. 6, pp. 14451–14461, 2018.
- [49] J. Seo, B. Lee, S. M. Kim, M.-W. Shih, I. Shin, D. Han, and T. Kim, "Sgx-shield: Enabling address space layout randomization for sgx programs," in *NDSS*, 2017.
- [50] M.-W. Shih, S. Lee, T. Kim, and M. Peinado, "T-sgx: Eradicating controlled-channel attacks against enclave programs," in *NDSS*, 2017.
- [51] Z. Yan, W. Ding, V. Niemi, and A. V. Vasilakos, "Two schemes of privacy-preserving trust evaluation," *Future Generation Computer Systems*, vol. 62, pp. 175–189, 2016.
- [52] P. Massa, K. Souren, M. Salvetti, and D. Tomasoni, "Trustlet, open research on trust metrics," *Scalable Computing: Practice and Experience*, vol. 9, no. 4, 2008.
- [53] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 17–24.

Wei Feng received the B.Sc. degree in telecommunications engineering from the Xidian University, Xian, China, in 2011, where he got the Ph.D. degree in information security in 2020. He has been in Aalto University, Finland from 2017 to 2019, working as a visiting scholar. His research interests include information security, privacy preservation, blockchain, mobile crowdsourcing, and trust management.

Zheng Yan received the B.Eng. degree in electrical engineering and the M.Eng. degree in computer science and engineering from the Xi'an Jiaotong University in 1994 and 1997, respectively, the second M.Eng. degree in information security from the National University of Singapore in 2000, and the licentiate of science and the doctor of science in technology in electrical engineering from Helsinki University of Technology. She is currently a professor at the Xidian University and a visiting professor at the Aalto

University. Her research interests are in trust, security, privacy, and security-related data analytics. Prof. Yan serves as a general or program chair for 30+ international conferences and workshops. She is a steering committee co-chair of IEEE Blockchain international conference. She is also an area editor or an associate editor of many reputable journals, e.g., IEEE Internet of Things Journal, Information Sciences, Information Fusion, JNCA, etc.

Laurence T. Yang is a fellow of the Canadian Academy of Engineering. He received the B.E. degree in Computer Science and Technology and B.Sc. degree in Applied Physics both from Tsinghua University, Beijing, China, and the Ph.D. degree in Computer Science from the University of Victoria, Victoria, BC, Canada. He is a professor in Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His research interests include parallel and distributed computing, embedded and

ubiquitous/pervasive computing, and big data. His research has been supported by the National Sciences and Engineering Research Council, Canada, and the Canada Foundation for Innovation.

Qinghua Zheng received the Ph.D. degree in System Engineering from the Xi'an Jiaotong University, Xi'an, China. He is currently a distinguished professor in this university. He achieved the National Funds for Distinguished Young Scientists, and is among the first batch of leading scientists of the "Ten-Thousand Talents Project", a candidate of the "New Century National BaiQianWan Talents Project" in China. He is leading the Innovation Team of National Natural Science Foundation of China, the Inno-

vative Team of Ministry of Education, and the Shaanxi Key Scientific and Technological Innovation Team. His main research interests are in intelligent e-learning, big data mining and application, as well as software reliability.