A Blockchain-based Non-Repudiation Network Computing Service Scheme for Industrial IoT

Yang Xu, *Student Member, IEEE,* Ju Ren, *Member, IEEE,* Guojun Wang, *Member, IEEE,* Cheng Zhang, Jidian Yang, and Yaoxue Zhang, *Senior Member, IEEE*

***Abstract*—Emerging network computing technologies extend the functionalities of industrial IoT (IIoT) terminals. However, this promising service-provisioning scheme encounters problems in untrusted and distributed IIoT scenarios because malicious service providers or clients may deny service provisions or usage for their own interests. Traditional non-repudiation solutions fade in IIoT environments due to requirements of trusted third parties or unacceptable overheads. Fortunately, the blockchain revolution facilitates innovative solutions. In this paper, we propose a blockchain-based fair non-repudiation service provisioning scheme for IIoT scenarios in which the blockchain is used as a service publisher and an evidence recorder. Each service is separately delivered via on-chain and off-chain channels with mandatory evidence submissions for non-repudiation purpose. Moreover, a homomorphic-hash-based service verification method is designed that can function with mere on-chain evidence. And an impartial smart contract is implemented to resolve disputes. The security analysis demonstrates the dependability, and the evaluations reveal the effectiveness and efficiency.**

***Index Terms*—Network computing, industrial IoT, blockchain, non-repudiation, homomorphic hash.**

I. Introduction

THE past years have witnessed dramatic developments in the Internet of Things (IoT) in various fields, especially in the industrial area. Currently, industrial IoT (IIoT) is building more powerful industrial systems in different sectors, such as manufacturing, logistics, healthcare, and energy [1], with a market of more than $100 billion [2].

Meanwhile, with the continuous evolutions of the computing paradigm from the stand-alone model toward the network-oriented one, novel network computing technologies, such as cloud computing [3], edge computing [4], [5] and transparent computing [6], have emerged and significantly extended the functionalities of terminal devices, with the on-demand service-provisioning scheme via the network. For instance, transparent computing, which enables local lightweight terminals to run a richer set of service programs acquired from remote servers in turn with the support of its dynamic program loading and execution techniques [7], has already provided significant benefits in the resource-constrained IoT [8], [9], [10] and IIoT scenarios [11], [12].

Unfortunately, such a promising service-provisioning scheme is facing new security challenges in untrusted distributed IIoT scenarios: untrusted service providers may provide nonconforming or even malicious services, while dishonest clients may maliciously repudiate the acquirements of correct services for their own benefits or even false accusations purposes. Undoubtedly, the lack of effective non-repudiation dispute resolution mechanisms means there is no guarantee for these mutually distrusted parties, which causes their anxieties and concerns about using such a promising service provisioning technology and in turn impedes its further development and acceptance.

By now, many traditional non-repudiation mechanisms have already been proposed [13] and can be mainly categorized as trusted third party (TTP)-based schemes [14], [15], [16], and non-TTP-based approaches [17], [18]. However, the TTP-based solutions are not applicable to decentralized IIoT environments because they rely on a centralized TTP that may not exist in particular decentralized scenarios, and suffer single-point failures and TTP performance bottlenecks. Meanwhile, non-TTP-based solutions incur huge performance costs unaffordable for most recourse-constrained IIoT devices, and encounter difficulty in ensuring true fairness, namely, they cannot ensure that both participants can obtain enough non-repudiation evidence to prove that they have fulfilled their obligations, or none of them can get any valuable information. In fact, they provide only weak fairness which means it is possible that one side cannot acquire enough non-repudiation evidence for self-proving.

Recently, the blockchain revolution [19], [20], [21] has sparked many innovative approaches in many aspects, including IIoT sectors [22], [23], [24], [25]. Meanwhile, due to its unparalleled advantages in decentralization and tamper-resistance, blockchain technology has also been reasonably introduced in several non-repudiation dispute resolution systems [26], [27], [28], [29], [30]. For example, Aniello et al. propose a log-based dispute resolution [27] in which the arbiter can make accurate decisions, according to the records on the blockchain. Klems et al. [29] propose a dispute resolution by using a smart contract, in which the intercessors can periodically invoke an arbitration contract to check the transactions. On this basis, Zhou et al. [30] introduce a consortium blockchain to decrease

Manuscript received October 31, 2018; revised January 10, 2019, accepted January 29, 2019. This work is supported by the NSFC (No. 61632009, 61702562 and 61472451), the 111 Project (No. B18059), International Science & Technology Cooperation Program of China (No. 2013DFB10070), China Hunan Provincial Science & Technology Program (No. 2012GK4106) and the Guangdong Provincial Natural Science Foundation (No. 2017A030308006). Paper no. TII-18-2851. (Corresponding author: Ju Ren.)

Y. Xu, J. Ren, C. Zhang and Y. Zhang are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: {xuyangcsu; renju; zhangcheng_sy; zyx}@csu.edu.cn)

G. Wang is with the School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China (e-mail: csgjwang@gmail.com)

J. Yang is with the School of Software, Central South University, Changsha 410083, China (e-mail: gideonyang@csu.edu.cn)

the computational load, which brought a more economical and practical approach. However, some of these non-repudiation approaches only achieve weak fairness [26], [27], and some others are inefficient and depend on massive valuable evidence for dispute resolutions, posing a huge overhead challenge especially to blockchain systems [28], [29].

Motivated by the challenges mentioned above, in this paper, we propose a blockchain-based truly fair non-repudiation service provisioning scheme for IIoT scenarios. We introduce the tamper-resistant blockchain as a service publication proxy and an evidence recorder that records interactive evidence of service providers and IIoT clients during service provisioning processes. The client-required service program is separated into two non-executable parts by service providers in advance and delivers via on-chain and off-chain channels in separate interactive steps after recognizing evidence of previous steps to enforce on-chain evidence submissions for even off-chain behaviors. Specially, the major part of the program is delivered via a private off-chain channel after publishing corresponding evidence on the blockchain for avoiding the additional on-chain storage cost and the risk of abuse of permanently exposed program codes, while the remainder fragment will finally be published on the blockchain for delivery after recognizing the expected evidence of previous steps for truly fair non-repudiation purpose. Moreover, with the aid of homomorphic hash techniques and lightweight cryptographic solutions for the IIoT [31], [32], we design an effective service verification method that can function with mere lightweight evidential on-chain records of service provisioning processes rather than complete service program codes for supporting our service model. In addition, we implement an impartial automated adjudicative smart contract with the service verification method embedded in, to help service providers and IIoT clients resolve service disputes fairly and efficiently. Finally, the security analysis theoretically demonstrates the dependability of our scheme, and the evaluated results of our prototype reveal that the proposed approach provides effective non-repudiation services with true fairness in network computing enabled IIoT scenarios, with sufficient throughput capability and affordable comprehensive costs.

Summarily, our major contributions are fourfold.

(1) We propose a blockchain-based truly fair non-repudiation service provisioning scheme for IIoT scenarios, in which the blockchain is used as an evidence recorder and a service publication proxy. The required service program is cut into non-executable parts for delivery via on-chain and off-chain channels in separate steps after recognizing the previous on-chain evidence, which can not only reduce the burden on the blockchain and avoid the program disclosure risk, but also enforce evidence submissions of even off-chain behaviors so that ensures the true fairness of our non-repudiation scheme.

(2) We design a service verification method based on homomorphic hash techniques, which can correctly validate services based on mere lightweight on-chain evidence rather than complete service program codes, supporting the core functionality of our model.

(3) We implement an impartial and effective dispute resolution mechanism based on smart contract techniques, to fairly and efficiently settle service disputes among service providers and IIoT clients.

(4) We theoretically analyze the security of our scheme and implement an experimental prototype for evaluating its performance in terms of effectiveness and efficiency.

The rest of this paper is organized as follows. Section II introduces some related work and background knowledge. Section III introduces the concept of homomorphism and the homomorphic hash function that is used in this paper. In Section IV, we propose a blockchain-based non-repudiation network computing service provisioning scheme for IIoT scenarios and the security analysis. Then, Section V analyses the results of our approach. The last section summaries this paper and describes our future plans for improvement.

II. Related Work

In this section, we summarize the traditional approaches related to non-repudiation dispute resolution services, and provide some background about the blockchain and its latest applications for achieving non-repudiation dispute resolution services.

A. Traditional Approaches

Undoubtedly, non-repudiation methods and dispute resolution mechanisms are extremely important for service delivery processes. Generally, these approaches can be divided into two types: TTP-based schemes and non-TTP-based ones [13].

As for the former, Coffey et al. [14] propose a classic non-repudiation protocol in which the TTP acts as the middleman who helps the service provider and client exchange the services and corresponding receipts during the service processes. Asokan et al. [15] present a more general method with an off-line TTP that is activated only when needed. In this scheme, the service provider provides the client with the data encrypted by a TTP key so that the client can only obtain the decrypted service after acknowledging the acquirement to the TTP. On this basis, Zhou et al. [33] propose an improved protocol that minimizes the workload of TTP. These protocols use a TTP to achieve fairness. Unfortunately, these approaches can only work properly with a TTP that may not exist in distributed IIoT scenarios, and they can easily experience single-point failures and performance bottlenecks.

In regard to the non-TTP-based schemes, Markowitch et al. [17] propose a typical non-repudiation protocol in which the service provider initially sends the client the encryption data, and then randomly sends a fake or true decryption key for multiple iterations. The service provider will terminate the process if the client does not return the evidential receipt immediately in every iteration so that the client does not have enough time to try to decrypt the ciphertext with the received key within an iteration. However, the client still has an advantage as it has opportunities to obtain the true key when it decides to cheat. Mitsianis et al. [18] propose a further developed protocol in which the server sends a fragment of the decryption key instead in iterations. Apparently, for the non-TTP-based approaches, they have difficulty in ensuring true fairness, and have notable performance disadvantages due to the multi-iteration processes of confirmations, which can not be afforded by most IIoT devices.

B. Blockchain and Relevant Non-repudiation Applications

1) Blockchain: The blockchain [19], [20], [21] is essentially a dependable distributed ledger that keeps transactions in a chain of chronological blocks that are linked via hash values. This scheme achieves trust among untrusted entities without the involvements of TTPs and has advantageous features such as decentralization, tamper-resistance, anonymity, etc. According to the participation schemes, blockchains are classified into two types: the permissionless and the permissioned ones. The former, i.e., public blockchains such as Bitcoin [19] and Ethereum [34], are fully open to every equal entity, while the latter, consisting of consortium blockchains (e.g., Hyperledger [35]) and private blockchains, are only available to granted members. Specifically, the consortium blockchain is jointly controlled by a privileged minority compared with other ordinary participants with less permissions, and the private blockchain is dominated by a single dictator. In addition, the consensus mechanism is the key of the blockchain by which the distributed miners can achieve a consensus on block generations. Commonly used consensus mechanisms include the Proof of Work (hashrate-dominated), the Proof of Stake (stake-dominated), the Proof of Authority (authority-determined) [36], etc.

Meanwhile, the developments of blockchain infrastructures lead to the success of smart contracts [37]. A smart contract is a set of digital promises that can be triggered in order to have automatic and faithful executions. By deploying it on a blockchain, the viewable contract obligation codes are triggerable by granted members and the trustworthiness of execution results appended to the blockchain will be ensured by every miner. Recently, smart contract languages are further developed and become Turing-complete. Thus, they can implement rather complex algorithms [34]. The explosive proliferation of smart contracts in turn expands the applicability of the original blockchain, making it increasingly more vital.

2) Blockchain-based Non-repudiation Mechanisms: More recently, some blockchain-based non-repudiation schemes have also been studied. Zou et al. [26] propose a service contract management scheme (SCMS) based on the blockchain. The SCMS requires the service provider to publish contracts on the blockchain in advance, and then asks the client to submit receipts of actual service actions to the blockchain. Finally, an additional dispute arbitration protocol is used to verify whether the service providers fulfill their contractual obligations. A similar blockchain-based approach is proposed by Aniello et al. [27]. In their log-based dispute resolution mechanism, all the service events are signed and recorded on the blockchain as non-repudiation evidence. However, these schemes are too idealistic as they assume that users will honestly and voluntarily publish correct evidential receipts on the blockchain without enforcement mechanisms. Since malevolent clients can intentionally deny the correctness of services, these biased and imperfect schemes can only achieve weak fairness and are far from satisfactory in practice.

Meanwhile, the smart contract has also been used in dispute resolution processes due to its transparency and impartiality. Koulu [28] propose a conceptual general design that uses a smart contract as an alternative to online dispute resolution enforcement. Klems et al. [29] also utilize the smart contract to resolve disputes. In their scheme, the monitoring agents are introduced to periodically trigger the inspection smart contracts to examine the transactions. However, in this approach, the service provider has to disclose its massive valuable service information on the blockchain for dispute resolution, which leads to inefficiency and huge overhead. And extra needs of the monitoring agents also limited its application in IIoT environments.

Summarily, although there exist numerous approaches aiming at providing non-repudiation services, they have suffered from notable disadvantages such as TTP dependence, weak fairness, and high costs for service verification and dispute resolution in practical environments, especially the IIoT scenarios. Therefore, a more fair and efficient non-repudiation service provisioning scheme is urgently needed in IIoT-oriented network computing environments.

III. Preliminary

This section provides the necessary background knowledge of the homomorphism and homomorphic hash function used in this paper.

Homomorphism. Given a map $f : (A; *) \to (B; \odot)$ between two sets A and B with the same type, for any element $x, y \in A$ and $f(x), f(y) \in B$, if $f(x * y) = f(x) \odot f(y)$, then it has the feature of homomorphism. In a certain map, if the operator $*$ in A is addition or multiplication, then it satisfies the homomorphism of the addition or multiplication correspondingly.

Homomorphic hash function. The homomorphic hash function is a kind of collision-resistant hash function that has the feature of homomorphism. Specifically, a homomorphic hash function h can map data of an arbitrary size to a fixed size message and satisfies $h(x * y) = h(x) \odot h(y)$. In addition, $h(x') \neq h(x)$ if $x' \neq x$. Currently, the most commonly used collision-resistant homomorphic hash function was raised by Krohn [38] as follows.

Let p and q be two large random primes and $q \mid (p-1)$. For a message block b_j with that is an $1 \times m$ vector, i.e., $b_j = (b_{1,j}, \cdots, b_{t,j}, \cdots, b_{m,j})$, each $b_{t,j}$ is an element of $\mathbb{Z}_q$ where $\mathbb{Z}_q$ is a set of nonnegative integers whose members are less than q. Then, its hash function is given as follows:

$$h_G(b_j) = \prod_{t=1}^{m} g_t^{b_{t,j}} \bmod p$$

Here, G is a set of hash parameters $G = (p, q, <\boldsymbol{g}>)$ where $<\boldsymbol{g}>$ is a $1 \times m$ row vector consisting of g_ts. Each g_t in $<\boldsymbol{g}>$ is a random element of $\mathbb{Z}_p$, of order q. The addition of two message blocks b_i and b_j can be computed as follows:

$$b_i + b_j = (b_{1,i} + b_{1,j}, \cdots, b_{m,i} + b_{m,j}) \bmod q$$

Thus, the homomorphism of the hash function is given by the following equation:

$$\begin{aligned} h_G(b_i + b_j) &= \prod_{t=1}^{m} g_t^{b_{t,i}+b_{t,j}} \bmod p \\ &= (\prod_{t=1}^{m} g_t^{b_{t,i}} \bmod p \times \prod_{t=1}^{m} g_t^{b_{t,j}} \bmod p) \bmod p \\ &= (h_G(b_i) \times h_G(b_j)) \bmod p \end{aligned}$$

The collision-resistance of the homomorphic hash function h_G is guaranteed by the computational intractability of the discrete logarithm problem. If p and q have sufficient sizes, then there is no known polynomial time algorithm for solving the discrete logarithm problem [38], [39].

IV. Approach

A. Threat Model

We assume that either the service provider or client will attempt to cheat the other side by performing malicious activities during the service provisioning process due to their self-interests, namely, the service provider may provide fake services while the client will try to deny the acquisition of the correct services. However, neither of them will perform unreasonable abnormal behaviors at the expense of their own interests, e.g., behaviors such as accepting incorrect services on the client side that are beyond our consideration.

B. Service Model

The service model of our scheme is shown in Fig. 1. Our scheme is built upon a consortium blockchain with the PoA consensus mechanism because this kind of blockchain network, which is jointly managed by several authoritative nodes with an authority-determined consensus mechanism, has good performance and will not incur extra maintenance overhead due to the resource-constrained IIoT devices [24], [25]. In our scheme, there exist three kinds of entities, namely, the service provider (SP, providing service programs), the IIoT client (C, requesting and executing the service programs) and the arbitration node (AN, privileged node in the blockchain that maintains the distributed ledger and executes smart contracts). Every entity above has an account address in the blockchain network so that it can initiate transactions and trigger smart contracts. And the data field of the original blockchain transaction is extended for containing extra information used in our approach like the requested service name, the hash value, the confirmation, and the evidential code fragment of a service program.

When the system initializes, the adjudicative smart contract is deployed on the blockchain with an embedded standard hash map of service names and corresponding homomorphic hash values of service programs. And a sufficient number of unique crypto-collectibles (i.e., the differentiated crypto-currency) are issued to every client account and will be used as synchronous tokens in the future service provisioning process between the SP and client in order to control the step sequence. Additionally, a set of homomorphic hash parameters $G = (p, q, <\mathbf{g}>)$ is selected for our scheme. Besides, to guarantee the fairness for its own interests, the SP will separate every complete service program S into two non-executable parts that would be delivered in different phases in advance, i.e., a small fragment S_1 for evidential on-chain publication and a major part S_2 for efficient and private off-chain delivery. Note that since an executable service program has an important non-executable header structure of small size without which the remaining major part cannot be executed interdependently, the service codes can be technically separated into non-executable parts by cutting a part of its header, for example, in our scenario, S_1 can be a random part of the tiny `ELF` header of an `ELF` file, or a random part of the small `PE` header of an `EXE` file, without which the remainder S_2 is unable to run.

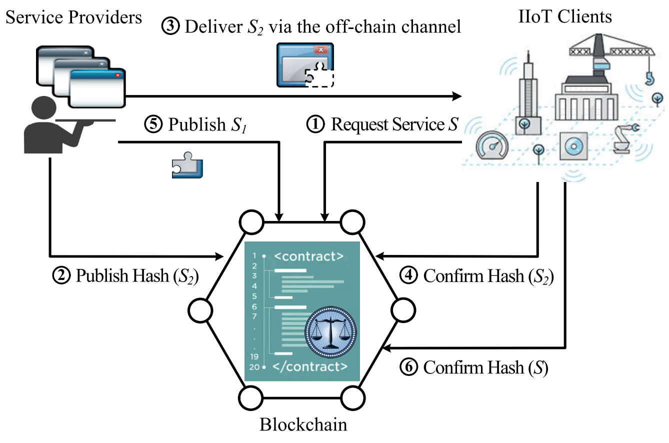


Fig. 1: The service model and workflow

The detailed workflow of the service provisioning process is demonstrated as follows (also see in Fig. 1).

- **Step 1**: [Service request]

The client C_i sends a request transaction including the service name S and synchronizing crypto-collectible $Token_i$ to the blockchain account of SP_i.

- **Step 2**: [Publishing $Hash(S_2)$]

SP_i calculates the hash value $Hash(S_2)$ of S_2 through the homomorphic hash function (*cf.* Section III Preliminary), and then encapsulates it with $Token_i$ in a transaction that will be sent to the blockchain account of C_i as the on-chain evidence.

Note that our scheme only records the small hash value of the large S_2 on the blockchain as lightweight evidence because the hash value is equally authentic with the smaller size.

- **Step 3**: [Delivering S_2 via the off-chain channel]

After publishing $Hash(S_2)$ on the blockchain, SP_i sends the major part of S_2 to C_i via the off-chain channel.

Note that S_2 is not executable in this phase so that C_i is forced to carry out following steps (e.g., publishing corresponding confirmation as evidence) if it still wants to acquire the complete executable service program finally, which helps to achieve a non-repudiation scheme with true fairness. Additionally, the off-chain delivery of the large program part helps to reduce the burden on the blockchain and can avoid the program disclosure risk.

• **Step 4**: [Confirming $Hash(S_2)$]

C_i calculates the homomorphic hash value of the received S_2 and compares it with that recorded on the blockchain. If hash values match, C_i sends the corresponding confirmation transaction with $Token_i$ to the blockchain account of SP_i. Otherwise, C_i will invoke the smart contract to terminate the current service process prematurely due to unknown exceptions. Note that SP_i can also trigger the smart contract to cancel the service process in this step if there is no response from C_i within a reasonable period of time.

According to the similar reason mentioned in Step 4, C_i is forced to publish corresponding confirmation for the off-chain event if it still hopes to obtain the complete executable service program. Otherwise, neither C_i nor SP_i will benefit from the early termination of the service-provisioning process, which shows the true fairness provided by our scheme.

• **Step 5**: [Publishing S_1]

SP_i sends a transaction that includes the non-executable tiny code fragment S_1 and $Token_i$ to the blockchain account of C_i as lightweight evidence when the previous confirmation transaction from C_i is available on the blockchain.

Note that publishing a randomly cut non-executable fragment of service program is harmless to the service provider as the rest major part are transmitted privately. However, copyright control, i.e., preventing against resource leakage from the client side, is out of the scope of this paper and can be complementary to our work.

• **Step 6**: [Confirming $Hash(S)$]

C_i obtains S_1 and restores the service S, then it decides to submit the final confirmation to the blockchain account of SP_i or to invoke the smart contract for arbitration. Similar to Step 4, SP_i is also eligible to trigger the smart contract for arbitration if there is no response from C_i within a reasonable period of time.

Note that since the last piece of code fragment is published on the blockchain as evidence in Step 5, the smart contract can resolve disputes according to the existing on-chain evidence without client's final confirmation, which ensures the true fairness of our scheme.

C. Dispute Resolution Mechanism

The adjudicative smart contract SC will be triggered by either the SP or the client in order to terminate the current service process in Step 4. Undoubtedly, when a client does not publish a positive confirmation in Step 4, there must be something exceptional (and bad) happened in the service process, e.g., the client is dishonest or it finds that the hash value of the received S_2 is unequal to that one on the blockchain. Therefore, it is an effective precautionary mechanism that helps to avoid further losses and ensures that no one will be exploited. In addition, this mechanism will not be abused due to the economic reason, i.e., every party has already been equally charged for its previous behaviors (*cf.* Section V-B, Paragraph ***Gas Consumption***).

Besides, the SC will also be invoked when the client or the SP applies for the arbitration in the final step. It first uses the homomorphic hash function to compute the hash value of the minor code fragment S_1^* that is published on the blockchain. Detailedly, as for $S_1 = (b_{1,1}, b_{2,1}, \cdots, b_{n,1})$, the SC calculates $Hash(S_1^*) = \prod_{t=1}^{n} g_t^{b_{t,1}} \bmod p$. Note that since S_1^* is shorter than S_2^*, the remaining positions will be padded with zeroes temporarily for calculation. And then the SC obtains $Hash(S_1^* + S_2^*)$ by calculating $Hash(S_1^*) \times Hash(S_2^*) \bmod p$ and compares it with the prestored $Hash(S)$ on the blockchain. At last, the SC will publish a judgment transaction on the blockchain. If $Hash(S_1^* + S_2^*)$ is equal to prestored $Hash(S)$, the SC will bring in a verdict that the client is malicious because it brings a false charge. Otherwise, it concludes that the SP is malicious (see **Algorithm** 1).

Algorithm 1 The Main Algorithm of Smart Contract

```
Input: the small code fragment $S_1^*$, the confirmed $Hash(S_2^*)$
       and the standard $Hash(S)$ from the blockchain
Output: the $Verdict$
1: $Hash(S_1^*) \leftarrow \prod_{t=1}^{n} g_t^{b_{t,1}} \bmod p$
2: $Hash(S_1^* + S_2^*) \leftarrow (Hash(S_1^*) \times Hash(S_2^*)) \bmod p$
3: if ($Hash(S_1^* + S_2^*) == Hash(S)$) then
4:     $Verdict = Malicious\ client$
5: else if then
6:     $Verdict = Malicious\ SP$
7: end if
8: return the $Verdict$
```

With the support of our homomorphic-hash-based service verification method, our smart contract can fairly resolve service disputes with lightweight on-chain evidence (i.e., a small part of the service program codes together with a hash value), thereby bringing about tremendous savings for the entire blockchain system, and avoiding illegal usage risks caused by complete program code exposure.

D. Security Analysis

In this section, we theoretically analyze the security of the proposed scheme.

We believe that the transactions that are recorded on the blockchain are tamper-resistant and the execution of the smart contract is trustworthy. Therefore the prestored $Hash(S)$ is trustworthy and the smart contract can obtain the correct hash value of S_1. In addition, according to our threat model, neither the SP nor the client will perform unreasonable abnormal behaviors at the expense of their own interests, therefore, $Hash(S_2^*)$ published on the blockchain is the correct hash value of the received S_2^* when it is confirmed by the client.

As for the service provider side, if the malicious SP cheats in either Step 2 (publishing $Hash(S_2)$) or Step 3 (delivering S_2 to the client via the off-chain channel), then the client C_i can find the mismatched hash values because there is no polynomial algorithm for the SP to forge a fake S_2' that does not equal to S_2 that satisfies $Hash(S_2') = Hash(S_2)$ according to the collision-resistant feature of the homomorphic hash function [38]. Therefore, C_i will claim it in Step 4 which leads to the termination of the service. If the malicious SP cheats in Step 5 (publishing S_1), the smart contract can find the mismatched hash values because it is impossible for the SP to forge a fake S_1' which does not equal to $S-S_2$ that satisfies $Hash(S_2+S_1') = Hash(S)$, where $Hash(S_2+S_1') = Hash(S_2) \times Hash(S_1')$. There are also some intricate scenarios in which the SP cheats in multiple steps. A malicious SP may provide a fake $Hash(S_2')$ and the corresponding fake S_2'. In this case, although the SP can pass the validation in Step 4, the smart contract can find the malicious behavior because the SP cannot counterfeit a fake S_2' that satisfies $Hash(S_2' + S_1) = Hash(S)$. Another similar attacking scenario is that a malicious SP further provides a fake S_1' instead of the true S_1 in the former case. Similarly, the smart contract can detect the cheat because there is no $Hash(S' = S_1' + S_2')$ that satisfies $Hash(S') = Hash(S)$ while S' is unequal to S.

In regard to the client side, if C_i does not confirm the $Hash(S_2)$ in Step 4, the SP will not perform Step 5. Therefore, C_i cannot obtain the complete executable service S such that the fairness is guaranteed. If C_i does not confirm the $Hash(S)$ in Step 6, the smart contract can also render a correct verdict according to the evidence recorded on the blockchain as mentioned above.

V. Evaluation

We implemented a prototype to evaluate the effectiveness and performance of the proposed approach. Based on *Geth 1.8.17*, we built a specific consortium blockchain network with the PoA consensus mechanism (5 arbitration nodes) in our intranet and deployed the adjudicative smart contract for dispute resolution services. For comparison, we also built a blockchain with the PoW consensus mechanism (5 mining nodes) in the same environment. Note that we set a quite low mining difficulty (i.e., difficulty = *0x*131072) for the PoW algorithm to facilitate and simplify the practical experimental processes. We simulated 50 IIoT clients to request 27 kinds of service programs from 6 virtual service providers. The detailed configurations of the devices are shown in Table. I. Additionally, for visualization purposes, we illustrated some experimental results through the visual user interface provided by the Ethereum Mist Wallet.

TABLE I: Specifications of devices

Parameter	Arbitration (mining) node	Service provider	IIoT client
CPU	3.0 GHz	2.5 GHz	900 MHz
RAM	8 GB	8 GB	1 GB
Storage	512 GB	1 TB	8 GB
Network	1000 Mb	100 Mb	100 Mb
OS	Ubuntu Server 16.04.1 LTS	CentOS 7.2-1511	Snappy Ubuntu Core

A. *Effectiveness*

In this section, we evaluated the effectiveness of our system.

1) Testing Setting

We set a pair of SPs and client to simulate malicious behaviors to see whether our approach can correctly resolve the service disputes. According to our service model, there exist the following independent malicious behaviors which will lead to disputes between the SPs and IIoT clients[1]:

(i) A malicious SP publishes a fake $Hash(S_2')$ in Step 2.
(ii) A malicious SP delivers a fake S_2' to the client in Step 3.
(iii) A dishonest client denies that it receives the correct S_2 from the SP in Step 4.
(iv) A malicious SP publishes the fake S_1' on the blockchain in Step 5 after the client confirms the hash value $Hash(S_2)$ in Step 4, and
(v) A malevolent client refuses to admit the correctness of the S_1 and maliciously applies for service dispute arbitration in Step 6.

Furthermore, these independent malicious behaviors may happen together and the combined cases are given as follows[2]:

(vi) A malicious SP provides a fake $Hash(S_2')$ and the corresponding fake S_2'.
(vii) A malicious SP provides a fake $Hash(S_2')$ and a fake S_2' whose hash value is not equal to $Hash(S_2')$, and
(viii) A malicious SP provides a fake $Hash(S_2')$, the corresponding fake S_2' and a fake S_1'.

In the experiments, we tested all the possible cases mentioned above several times to see whether our approach can correctly resolve the disputes. All the arbitration results were recorded on the blockchain in the form of transactions.

2) Result Analysis

According to the transactions on the blockchain, our system impartially rendered correct verdicts for every experimental dispute. Specially, we discussed several typical cases in terms of the malicious SP and client respectively, to concretely demonstrate the effectiveness of our system.

- **Malicious Service Provider**

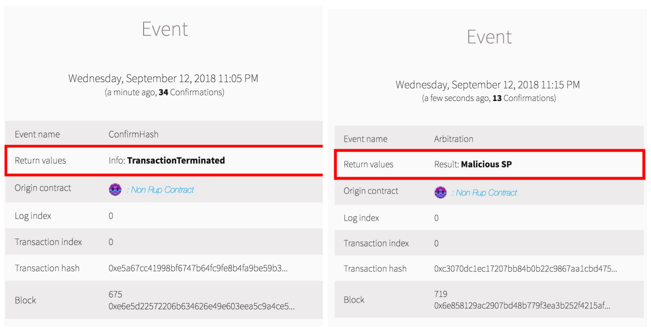
There are six malicious cases initiated by a malevolent service provider, i.e., cases (i), (ii), (iv), (vi), (vii) and (viii). Some visual results that were observed via an Ethereum Mist Wallet are shown in Fig. 2.

Among them, Fig. 2(a) reveals the result of the case (i). We can see that the smart contract terminated the service process. Since the client rejected the hash value in Step 4 when it found that $Hash(S_2)$ did not match that of the received S_2, the smart contract was triggered in order to terminate the abnormal service process in time to avoid unnecessary resource consumption. Similar results are observed in the case (ii) and case (vii).

In addition, Fig. 2(b) shows the arbitration result of case (iv). In this case, the client invoked the smart contract for the arbitration because it did not acquire the correct service. We can see that the smart contract made an accurate judgment, i.e., the service provider is malicious during the service process. Similar phenomena were observed in both case (vi) and case (viii).

[1]We assume that neither the SP nor the client will perform abnormal behaviors at the expense of their own interests, e.g., behaviors like accepting incorrect services on the client side are out of our considerations.

[2]In practice, the SP and the client may cheat concurrently, i.e., a malicious SP provides a fake $Hash(S_2)$ and a fake S_2, and the malicious client denies that the hash value matches the data in Step 4. We treat it as case (iii).



Event

Wednesday, September 12, 2018 11:05 PM
(a minute ago, **34** Confirmations)

Event name	ConfirmHash
Return values	Info: **TransactionTerminated**
Origin contract	: Non Rup Contract
Log index	0
Transaction index	0
Transaction hash	0xe5a67cc41998bf6747b64fc9fe8b4fa9be59b3...
Block	675 0xe6e5d22572206b634626e49e603eea5c9a4ce5...

Event

Wednesday, September 12, 2018 11:15 PM
(a few seconds ago, **13** Confirmations)

Event name	Arbitration
Return values	Result: **Malicious SP**
Origin contract	: Non Rup Contract
Log index	0
Transaction index	0
Transaction hash	0xc3070dc1ec17207bb84b0b22c9867aa1cbd475...
Block	719 0x6e858129ac2907bd48b779f3ea3b252f4215af...

(a) Service Terminated (b) Arbitration Result

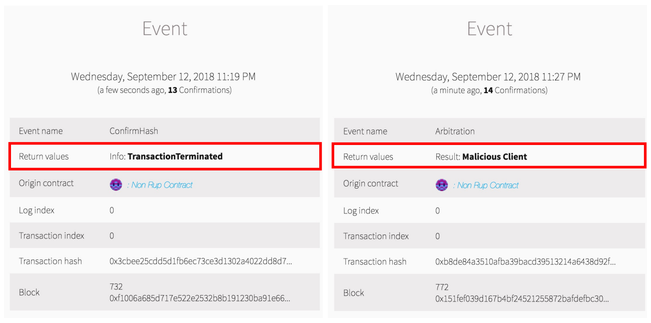
Fig. 2: Arbitrations Against Malicious SPs.

- **Malicious Client**

A dishonest client may cheat in case (iii) or case (v).

As shown in Fig. 3(a), the smart contract was triggered to cancel the service process immediately when a malicious client denied that the service provider sent a correct S_2 in Step 4. Therefore, a malicious client can never obtain the entire service successfully without honest confirmations.

In addition, Fig. 3(b) shows the arbitration result of case (v). In this case, the service provider triggered the smart contract for arbitration because the client did not confirm the correctness of the entire service in Step 6. We can see that our system correctly concluded that the client was malicious.



Event

Wednesday, September 12, 2018 11:19 PM
(a few seconds ago, **13** Confirmations)

Event name	ConfirmHash
Return values	Info: **TransactionTerminated**
Origin contract	: Non Rup Contract
Log index	0
Transaction index	0
Transaction hash	0x3cbee25cdd5d1fb6ec73ce3d1302a4022dd8d7...
Block	732 0xf1006a685d717e522e2532b8b191230ba91e66...

Event

Wednesday, September 12, 2018 11:27 PM
(a minute ago, **14** Confirmations)

Event name	Arbitration
Return values	Result: **Malicious Client**
Origin contract	: Non Rup Contract
Log index	0
Transaction index	0
Transaction hash	0xb8de84a3510afba39bacd39513214a6438d92f...
Block	772 0x151fef039d167b4bf24521255872bafdefbc30...

(a) Service Cancelled (b) Arbitration Result

Fig. 3: Arbitrations Against Malicious Clients.

In conclusion, as a truly fair non-repudiation service provisioning scheme, our system can not only terminate the abnormal service processes in time but also detect the malicious party after the service process, which contributes to enhancing the security of the IIoT-oriented network computing systems.

B. Efficiency

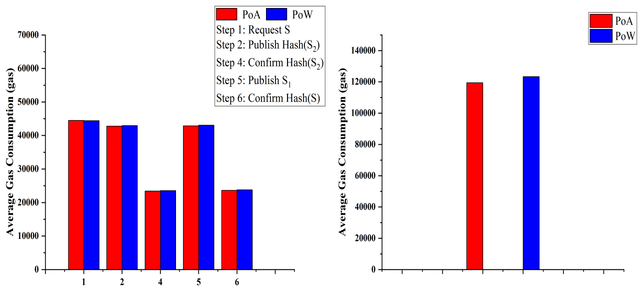
In this section, we evaluated the performance of our system in terms of gas consumption, transaction latency, transaction throughput, bloating rate, and the overhead on the IIoT client.

1) Gas Consumption

In the Ethereum blockchain ecosystem, the gas is a significant concept whose amount reflects the execution consumption for handling a blockchain transaction consisting of multiple operations. A small fee will be charged to reward blockchain miners for handling the transaction and it is determined by gas quantity $\times$ gas price, where the gas price is the unit price of gas whose cryptocurrency unit is gwei (10^8 gwei = 1 ether [basic unit of Ethereum cryptocurrency] $\approx$ 210 USD). Since the gas consumption indicates the operating cost from the perspective of an Ethereum blockchain network, we evaluated it to study the major operating cost of our system. Note that we set the default gas price as 12.5 gwei in our system based on the recommendation of Ethereum community.

We simulated the service provisioning processes and arbitration events for 500 times each under the PoA and PoW consensus mechanisms, and subsequently calculated the corresponding average gas consumptions.

As shown in Fig. 4, we find that the total gas consumption for a service provisioning process is less than $200,000$ gas, which is only worth approximately 0.002503 ether (≈ 0.526 USD). In addition, the average gas consumption of a dispute arbitration process is less than $60,000$ units of gas, i.e., 0.00075 ether (≈ 0.158 USD). Apparently, benefiting from the lightweight of our evidential transactions and the efficiency of homomorphic-hash-based dispute resolution mechanism, our system only incurred quite small economic costs that are very affordable in practice.



(a) Service Provisioning Process (b) Arbitration Event

Fig. 4: The Average Gas Consumption

2) Transaction Latency

The transaction latency shows how fast the blockchain system can handle transactions. In this experiment, we initiated 100000 transaction requests randomly every day and kept the system running for a week. The transaction delay time under two consensus mechanisms were recorded in order to measure the average transaction latency of our system.

As we can see from Fig. 5, for different time periods, the average transaction latency keeps stable, which is only approximately 21 ms under the PoA mechanism compared with approximately 52 ms under the PoW mechanism (that is more than twice as slow (approximately 250%) even with the very low mining difficulty), which also reflects the superiority of the PoA mechanism employed by our scheme.

Overall, this experiment result shows that the average transaction latency of our system is stable and negligible in both service provisioning and arbitration processes. Hence, our system which is built upon a consortium blockchain with

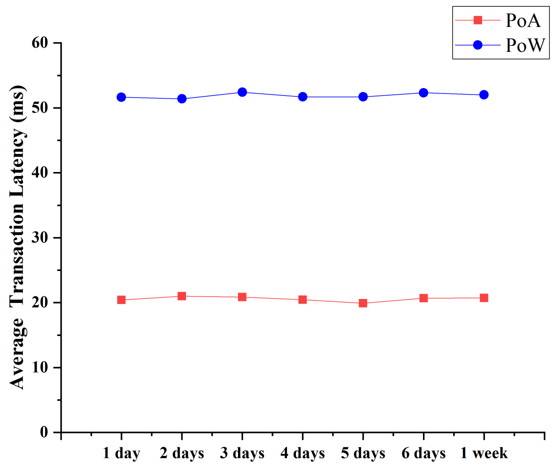


Fig. 5: The Average Transaction Latency

PoA consensus mechanism is responsive enough to handle corresponding tasks.

3) Transaction Throughput

The transaction throughput represents the concurrent transaction processing ability of a blockchain system. Hence, we studied the arbitration throughput of our system under the PoA and PoW mechanisms respectively.

As shown in Fig. 6, the average arbitration throughput of our PoA-based scheme increases stably along with the growth of the concurrent disputes, and it gradually reaches a stable peak, i.e., approximately 4500 every second when the number of concurrent disputes is greater than 4500/s. In regard to the PoW-based scheme, the arbitration throughput curve flattens much earlier with a maximum value of 1500 per second. Obviously, the throughput of our PoA-based scheme is superior to that of the PoW-based one and is large enough to meet the practical needs for dispute resolutions.

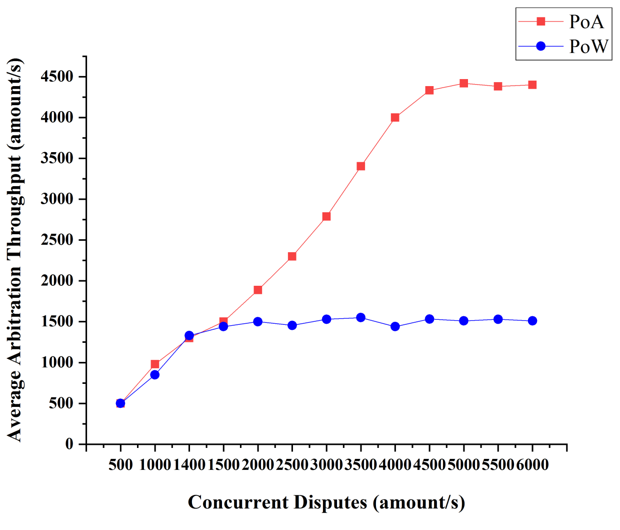


Fig. 6: The Average Throughput

4) Bloating Rate

As another critical performance indicator of a blockchain system, the bloating rate represents the growth rate of the data volume stored in a blockchain network, which is inversely proportional to the scalability and sustainability of a blockchain system. Therefore, we also evaluated the bloating rate of our blockchain system under several different situations (see in Table. II).

TABLE II: Blockchain Bloating Rate

Transaction Rate	Per Minute	Per Hour	Per Day
6,000 transactions/min	93.75 KB	5.493 MB	131.835 MB
30,000 transactions/min	468.75 KB	27.466 MB	659.180 MB
60,000 transactions/min	937.5 KB	54.931 MB	1.287 GB
90,000 transactions/min	1.373 MB	82.38 MB	1.931 GB

According to Table. II, for example, when there are 6000 transactions appended to the blockchain every minute (i.e., 100 transactions per second), the bloating rate of the blockchain system would be 93.75 KB per minute, 5.493 MB per hour and 0.13 GB per day.

Undoubtedly, the low bloating rate indicates that our system significantly benefits from the off-chain delivery mechanism and the technical support of homomorphic-hash-based service verification method, and only imposes a quite low storage overhead to the blockchain maintainers even in task-intensive environments, and thus is practical and acceptable in industrial information systems.

5) Overhead on IIoT clients

As the resource-constrained devices, the IIoT clients are very sensitive to performance overheads. In our system, the communication costs are negligible to IIoT clients and there are no storage costs on the IIoT client side. Thus, the most noteworthy costs for an IIoT client are the computing costs, namely, the computational overhead for running the homomorphic hash function to obtain the hash value of the major program fragment S_2 in Step 4. Thus, we studied it through the experiment. Detailedly, we tested several service samples from the IoT-oriented network computing system [9], and the experimental results are shown in Table. III.

TABLE III: Computing Cost

Code Size	Processing Time	Throughput Rate
16 KB	$1,261.72$ ms	12.681 KB/s
160 KB	$1,322.25$ ms	121.006 KB/s
1 MB	$4,367.62$ ms	234.45 KB/s
10 MB	$8,919.05$ ms	1.148 MB/s

According to Table III, the IIoT client can calculate the corresponding hash value in approximately 1 second with an average throughput rate of 121.006 KB/s when the code size is 160 KB[3], and this rate increases with the growth of the code size (e.g., more than 1 MB/s for a 10 MB code fragment), which is efficient enough for practical use. Therefore, our scheme

[3]Generally, the sizes of service programs for IIoT devices are small.

only imposes a small overhead on the resource-constrained IIoT clients so that they can be very responsive in the service provisioning processes.

Summarily, our blockchain-based non-repudiation system offers significant performance with affordable additional overheads. Therefore, we believe that it is able to efficiently handle the service provisioning and arbitration process in practical IIoT-oriented network computing scenarios.

VI. Conclusion

In this paper, we have proposed a blockchain-based truly fair non-repudiation service provisioning scheme for IIoT-oriented network computing system. In our approach, the blockchain is used as a service publication proxy and an evidence recorder that records interactive evidence of service providers and IIoT clients in service provisioning processes. The client-required service program is split into non-executable parts and then delivered via on-chain and off-chain channels in different phases, for not only limiting the burden on the blockchain network but also ensuring the mandatory on-chain evidence submission for fair non-repudiation purpose. Further, we have designed a lightweight service verification method with the aid of the homomorphic hash technique that can verify services based on mere lightweight on-chain evidence rather than complete service program codes to support the functionality of our system. Additionally, the impartial automated adjudicative smart contract with the service verification method embedded in is also implemented to help the service providers and IIoT clients resolve service disputes fairly with low costs. Finally, we have theoretically demonstrated the dependability of the proposed scheme through the security analysis, and the evaluation results of our designed prototype revealed its effectiveness and efficiency in providing fair non-repudiation services.

In the future, we intend to deploy our approach in a real network computing enabled IIoT platform for further practical evaluations. In addition, we would like to integrate more features, such as deposit and reputation mechanisms, in order to establish a more sophisticated and effective solution. Moreover, since IIoT devices have to store massive amounts of generated data in remote servers in network computing scenarios, the corresponding non-repudiation dispute resolution mechanisms will also be considered in our future solutions.

References

[1] D. Li, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] Grand View Research, "Industrial internet of things (IIoT) market analysis and segment forecasts to 2025," 2017.

[3] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Special Publication 800-145*, pp. 1–7, 2011.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.

[5] H. Liu, Y. Zhang, and T. Yang, "Blockchain-enabled security in electric vehicles cloud and edge computing," *IEEE Netw.*, vol. 32, no. 3, pp. 78–83, 2018.

[6] Y. Zhang and Y. Zhou, "Transparent computing: A new paradigm for pervasive computing," in *Proc. Int. Conf. Ubiquitous Intell. Comput.*, 2006, pp. 1–11.

[7] J. He, Y. Zhang, J. Lu, M. Wu, and F. Huang, "Block-stream as a service: a more secure, nimble, and dynamically balanced cloud service model for ambient computing," *IEEE Netw.*, vol. 32, no. 1, pp. 126–132, 2018.

[8] H. Guo, J. Ren, D. Zhang, Y. Zhang, and J. Hu, "A scalable and manageable IoT architecture based on transparent computing," *J. Parallel Distrib. Comput.*, vol. 118, pp. 5–13, 2018.

[9] X. Peng, J. Ren, L. She, D. Zhang, J. Li, and Y. Zhang, "BOAT: A block-streaming App execution scheme for lightweight IoT devices," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1816–1829, 2018.

[10] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, 2017.

[11] Y. Zhang and Y. Zhou, "Transparent computing: A promising network computing paradigm," *Comput. Sci. Eng.*, vol. 19, no. 1, pp. 7 –20, 2017.

[12] W. Li, B. Wang, J. Sheng, K. Dong, Z. Li, and Y. Hu, "A resource service model in the industrial IoT system based on transparent computing," *Sensors*, vol. 18, no. 4, pp. 981–1003, 2018.

[13] S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," *Comput. Commun.*, vol. 25, no. 17, pp. 1606–1621, 2002.

[14] T. Coffey, P. Saidha, and P. Burrows, "Analysing the security of a non-repudiation communication protocol with mandatory proof of receipt," in *Proc. 1st Int. Symp. Inform. Comm. Technol.* Trinity College Dublin, 2003, pp. 351–356.

[15] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for fair exchange," in *Proc. 4th ACM Conf. Comput. Comm. Security*. ACM, 1997, pp. 7–17.

[16] H. Berthold, A. Michael and B. Ruth, "A fair non-repudiation service in a web services peer-to-peer environment," *Compt. Standards & Interfaces*, vol. 30, no. 6, pp. 372–378, 2008.

[17] O. Markowitch and Y. Roggeman, "Probabilistic non-repudiation without trusted third party," in *Proc. 2nd Conf. Security Comm. Netw.*, vol. 99, 1999, pp. 25–36.

[18] J. Mitsianis, "A new approach to enforcing non-repudiation of receipt," *Manuscript*, 2001.

[19] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, pp. 1–9, 2008.

[20] Z. Zheng, S. Xie, H. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web. Grid. Serv.*, pp. 1–25, 2016.

[21] D. J. Yaga, P. M. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," *NIST Interagency/Internal Report (NISTIR) - 8202*, pp. 1–59, 2018.

[22] N. Teslya and I. Ryabchikov, "Blockchain platforms overview for industrial IoT purposes," in *Proc. 22nd Conf. Open Innov. Assoc.*, 2018, pp. 250–256.

[23] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat*, vol. 13, no. 6, pp. 3154–3164, 2017.

[24] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, 2018.

[25] Y. Xu, G. Wang, J. Yang, J. Ren, Y. Zhang, and C. Zhang, "Towards secure network computing services for lightweight clients using blockchain," *Wireless Commun. Mobile Comput.*, pp. 1–12, 2018.

[26] Y. Zou, J. Wang and O. Mehmet, "A dispute arbitration protocol based on a peer-to-peer service contract management scheme," in *Proc. IEEE. Int. Conf. Web. Services (ICWS)*, 2016, pp. 41–48.

[27] R. Aniello, L. Baldoni and F. Lombardi, "A blockchain-based solution for enabling log-based resolution of disputes in multi-party transactions,"

in *Proc. Int. Conf. Software. Engrg. Defence. Appl.* Springer, 2016, pp. 53–58.

[28] R. Koulu, "Blockchains and online dispute resolution: smart contracts as an alternative to enforcement," *SCRIPTed - A J. Law, Technol. Soc.*, vol. 13, no. 1, pp. 40–69, 2016.

[29] M. Klems, J. Eberhardt, S. Tai, S. Härtlein, S. Buchholz, and A. Tidjani, "Trustless intermediation in blockchain-based decentralized service marketplaces," in *Proc. Int. Conf. Service-Oriented Compt. (ICSOC)*. Springer, 2017, pp. 731–739.

[30] L. Zhou, G. Wang, T. Cui, and X. Xing, "Cssp: The consortium blockchain model for improving the trustworthiness of network software services," in *Proc. IEEE Int. Symp. Parallel and Distributed Processing Appl. and IEEE Int. Conf. Ubiquitous Compt. Comm. (ISPA/IUCC)*. IEEE, 2017, pp. 101–107.

[31] K. R. Choo, S. Gritzalis, and J. Park, "Cryptographic solutions for industrial Internet-of-Things: Research challenges and opportunities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3567–3569, 2018.

[32] W. Tang, K. Zhang, R. Ju, Y. Zhang, and X. S. Shen, "Flexible and efficient authenticated key agreement scheme for BANs based on physiological features," *Accepted to be appeared in IEEE Trans. Mobile Comput.*, 2018.

[33] J. Zhou and D. Gollmann, "An efficient non-repudiation protocol," in *Proc. 10Th Comput. Security Foundations Workshop (CSFW)*. IEEE, 1997, pp. 126–132.

[34] Ethereum, 2018. [Online]. Available: https://www.ethereum.org/.

[35] Hyperledger, 2018. [Online]. Available: https://www.hyperledger.org/.

[36] S. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain," in *Proc. Italian Conf. Cybersecurity*, 2017, pp. 1–11.

[37] N. Szabo, "Smart contracts: Building blocks for digital markets," *EXTROPY: J. Transhumanist Thought*, 1996.

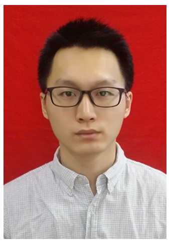
[38] M. Krohn, M. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Proc. IEEE Symp. Security & Privacy*. IEEE, 2004, pp. 226–240.

[39] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing," in *Proc. Annual Int. Cryptology Conf.* Springer, 1994, pp. 216–233.

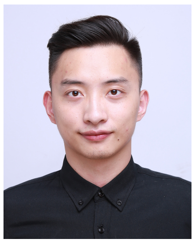
Yang Xu (S'17) received the B.Sc. degree in information security from Central South University, Changsha, China in 2011. He is currently working toward the Ph.D. degree in the School of Computer Science and Engineering at Central South University, Changsha, China. From November 2015 to February 2017, he was also a visiting Ph.D. Student in the Department of Computer Science and Engineering at Texas A&M University, TX, USA. He is a student member of IEEE and CCF. During the past 5 years, he has published over 15 papers in international conferences and journals. His major research areas include network computing, blockchain technique and cyber security.

Ju Ren (S'13, M'16) received the B.Sc. (2009), M.Sc. (2012), Ph.D. (2016) degrees all in computer science, from Central South University, China. During 2013-2015, he was a visiting Ph.D. student in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Currently, he is a professor with the School of Computer Science and Engineering, Central South University, China. His research areas include IoT, wireless communication, network computing and cloud computing. He is a co-recipient of the best paper award of IEEE IoP 2018 and the most popular paper award of Chinese Journal of Electronics. He currently serves as an associate editor for IEEE TVT and PPNA. He is a member of IEEE and ACM.

Guojun Wang (M'08) received B.Sc. degree in Geophysics, M.Sc. degree in Computer Science, and Ph.D. degree in Computer Science, at Central South University, Changsha, China, in 1992, 1996 and 2002, respectively. He is a Pearl River Scholar Professor of Higher Education in Guangdong Province, a Doctoral Supervisor of School of Computer Science and Educational Software, Guangzhou University. He had been a Professor at Central South University; an Adjunct Professor at Temple University, USA; a Visiting Scholar at Florida Atlantic University, USA; etc. He is a member of IEEE, ACM, and IEICE. His research interests include cloud computing, mobile computing, dependable computing and cyber security.



Cheng Zhang was born in Shenyang, Liaoning, China in 1995. He received his B.S. degree in Computer Science & Technology from Shenyang University of Technology, Shenyang, China in 2017. He is currently a graduate student of the School of Computer Science and Engineering at the Central South University, Changsha, China, and a member of the Transparent Computing Laboratory. He currently is pursuing the M.Sc degree in Computer Science and Technology. During the period of school, he wins scholarship many times. His research interests mainly focus on cyber security, blockchain technique, consensus protocols for distributed systems, cloud computing, edge computing and transparent computing.



Jidian Yang was born in Shaoxing, Zhejiang, China, in 1993. He received the B.S. degree in Wood Science and Engineering from Tianjin University of Science & Technology, Tianjin, China in 2016. He is currently a student of School of Software, Central South University, Changsha, China and a member of the Transparent Computing Laboratory of Central South University. Currently, he is pursuing his M.S. degree in Software Engineering. His research interests mainly focus on routing security, network security, blockchain technique, routing algorithm for wireless sensor networks, reinforcement learning, machine learning, and stochastic network optimization. He has already published several papers in international journals.

Yaoxue Zhang (M'17, SM'18) received his B.Sc. degree from Northwest Institute of Telecommunication Engineering, China, in 1982, and his Ph.D. degree in computer networking from Tohoku University, Japan, in 1989. Currently, he is a professor with the School of Computer Science and Engineering, Central South University, China, and a professor with the Department of Computer Science and Technology, Tsinghua University, China. His research areas include computer networking, operating systems, pervasive computing, transparent computing, and big data. He has published over 200 papers in international journals and conferences, as well as 9 monographs and textbooks. Currently, he serves as the Editor-in-Chief of Chinese Journal of Electronics. He is a fellow of the Chinese Academy of Engineering.