Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 2051693, 12 pages
https://doi.org/10.1155/2018/2051693

Research Article

Towards Secure Network Computing Services for Lightweight Clients Using Blockchain

Yang Xu,[1] **Guojun Wang**,[2] **Jidian Yang**,[1] **Ju Ren**,[1] **Yaoxue Zhang**,[1] **and Cheng Zhang**[1]

[1]*School of Information Science and Engineering, Central South University, Changsha 410083, China*
[2]*School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China*

Correspondence should be addressed to Yang Xu; xuyangcsu@csu.edu.cn

Received 27 July 2018; Revised 18 October 2018; Accepted 1 November 2018; Published 13 November 2018

Guest Editor: Constantinos Kolias

The emerging network computing technologies have significantly extended the abilities of the resource-constrained IoT devices through the network-based service sharing techniques. However, such a flexible and scalable service provisioning paradigm brings increased security risks to terminals due to the untrustworthy exogenous service codes loading from the open network. Many existing security approaches are unsuitable for IoT environments due to the high difficulty of maintenance or the dependencies upon extra resources like specific hardware. Fortunately, the rise of blockchain technology has facilitated the development of service sharing methods and, at the same time, it appears a viable solution to numerous security problems. In this paper, we propose a novel blockchain-based secure service provisioning mechanism for protecting lightweight clients from insecure services in network computing scenarios. We introduce the blockchain to maintain all the validity states of the off-chain services and edge service providers for the IoT terminals to help them get rid of untrusted or discarded services through provider identification and service verification. In addition, we take advantage of smart contracts which can be triggered by the lightweight clients to help them check the validities of service providers and service codes according to the on-chain transactions, thereby reducing the direct overhead on the IoT devices. Moreover, the adoptions of the consortium blockchain and the proof of authority consensus mechanism also help to achieve a high throughput. The theoretical security analysis and evaluation results show that our approach helps the lightweight clients get rid of untrusted edge service providers and insecure services effectively with acceptable latency and affordable costs.

1. Introduction

The Internet of Things (IoT) industry has evolved remarkably in the last decade. Currently, there exist more than 13 billion connected IoT devices and this number would increase to 30 billion in the near future [1]. Meanwhile, the emerging network computing technologies, typically, fog/edge computing [2, 3] and transparent computing [4, 5], have significantly extended the abilities of the existing resource-constrained IoT devices, through the network-based service provisioning and sharing mechanisms. For example, in the IoT-oriented edge transparent computing scenario [6, 7], with the aid of block-stream code loading and execution techniques [8], the resource-constrained wearable devices (e.g., wristbands and smartwatches) are enabled to alternately run numerous applications obtained from either the cloud servers or close edge servers (e.g., personal computers), which goes beyond the original capabilities of these local devices (see Figure 1) [9].

However, such flexible and scalable service provisioning paradigms bring increased security risks to terminal devices unintentionally. Comparing to the traditional closed architectures, the attack surfaces of network computing systems have inevitably increased due to the opening service sharing over the network [10, 11]. The frequent-changing exogenous service codes loading on the clients from the remote servers via the network can be unreliable, fragile, and even harmful to the host terminals in absence of adequate security mechanisms [12]. To make things worse, the various edge servers intermingled with vulnerable and malicious ones certainly heighten the risks.

Some early studies have already been done for protecting the terminals from illegal services in network computing

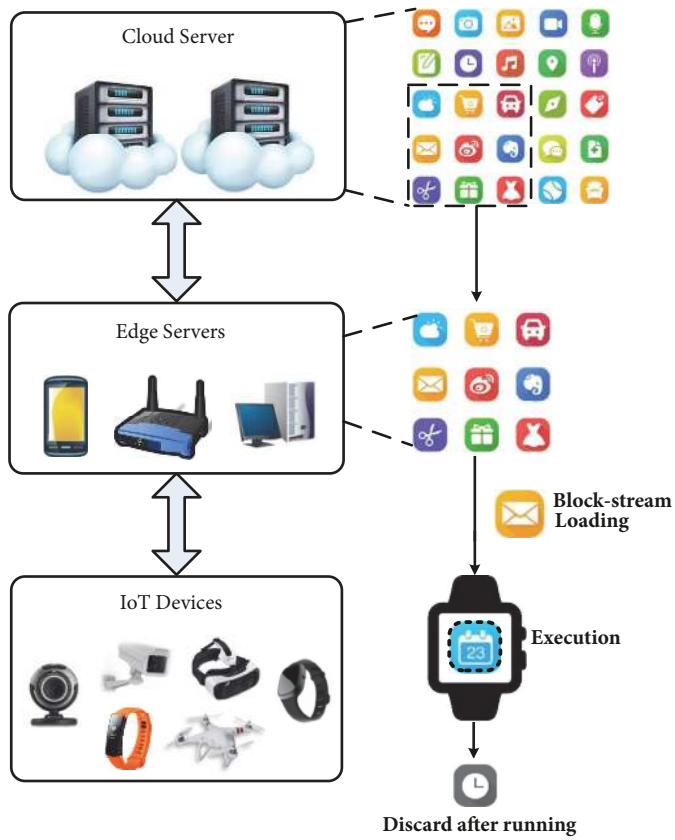


Figure 1: The IoT-oriented edge transparent computing scenario.

scenarios [12–16]. Based on integrity verification techniques, these works equip the terminals with the abilities to check the validities of the acquired service programs before executions with the help of static information (e.g., hash checksum) prestored in local trusted firmware [12–15] or trusted platform module (TPM) [16]. However, when it comes to IoT scenarios [17] in which the vulnerable IoT devices are threatened by distributed cyberattacks, the rigid prestored information is technically less maintainable for updating, while the spare firmware space or specific hardware is usually unavailable.

Recently, the rise of blockchain technologies [18, 19] inspires researchers for brand new solutions. With the excellent features of openness, decentralization, and tamper resistance, the blockchain techniques have been used as the underlying security fabric for a bunch of emerging service provisioning and sharing systems [20–29]. These approaches leverage the blockchain to release service information so as to ensure that the clients can obtain services correctly. Unfortunately, these blockchain-based schemes usually have low throughput and high service latency problems and take little consideration of necessary information updating as well as the legality validation of the numerous service providers. Even worse, few of them are designed for IoT scenarios and thus bring about unaffordable computing and storage costs to most existing IoT devices.

Motivated by the situations mentioned above, in this paper, we proposed a novel blockchain-based secure service provisioning mechanism to protect the lightweight clients from insecure exogenous service codes from untrustworthy edge servers in the edge transparent computing scenario. We leverage the blockchain to maintain all the validity states of the off-chain services and edge servers dynamically updated by the arbitration cloud merchants, to help the lightweight clients get rid of untrusted or discarded services through the provider identification and service verification. Besides, the specific smart contracts [30] are introduced and can be triggered to verify the validities of the edge servers and service codes on behalf of the lightweight clients according to the on-chain transactions, thereby reducing the direct costs of these IoT devices. Furthermore, a consortium blockchain with the proof of authority consensus mechanism [31, 32] is employed to achieve a high throughput and low latency further. Finally, we demonstrate the security of our approach and then test it comprehensively. The evaluation results show that our approach protects the lightweight clients from untrusted edge service providers and undependable service codes effectively with acceptable latency and affordable costs.

Summarily, the major contributions of our work are threefold:

(1) We design a blockchain system to maintain the appendable and tamper-resistant validity states of the off-chain services and edge servers dynamically declared by the arbitration cloud merchants, to help the lightweight clients get rid of insecure or deprecated services by the means of provider identification or service verification.

(2) We not only introduce smart contracts, which can be triggered by the lightweight clients to help them check the validities of the acquired services and edge servers according to the transactions on chain for reducing the costs of these IoT devices, but also employe the efficient consortium blockchain with the proof of authority consensus engine for ensuring the high throughput and low latency of the entire system.

(3) We demonstrate the security of the proposed approach, implement a prototype based on the Ethereum project [33], and evaluate its effectiveness and efficiency in the IoT-oriented edge transparent computing environment.

The rest of this paper is organized as follows. Section 2 gives an introduction to some related work and shores up our choice of blockchain technique for protecting lightweight clients from insecure service in network computing scenarios. In Section 3, we propose a blockchain-based secure service provisioning mechanism for lightweight clients in network computing scenarios. And then, we discuss the security of the proposed approach and evaluate it in experiments in Section 4. Finally, Section 5 concludes this paper and describes possible enhancement.

2. Related Work

In this section, we introduce some existing approaches about secure service sharing mechanisms which can be applied to the network computing environments.

To defend against the threatening service codes loading from the remote servers via the open network, Kuang et al. [12] proposed a security-enhanced service sharing approach for local terminals in network computing by using the integrity checking technique. This approach deploys the checking procedures together with static hash results of services on the local firmware and checks the integrity of acquired service codes from the Internet. Therefore, the terminals are secured as any unmatched suspicious service code would be discarded without execution. Furthermore, the software engineers of Intel Cooperation [13–16] proposed a series of integrity-checking-based secure methods on the UEFI (Unified Extensible Firmware Interface) firmware collaborating with the dedicated TPM hardware. However, these security approaches become unpractical in the IoT scenarios. For one thing, the static information prestored in local device is rigid and technically less updatable, especially in IoT scenarios. For another, the requirements of extra supports of firmware or specific hardware are usually unavailable for lightweight IoT devices.

In recent years, the emergence and fast growth of blockchain technologies [18, 19] also contribute to the development of service sharing techniques and meanwhile indicate a new way to secure the local terminals from the threats of untrusted extraneous services. As a decentralized ledger built upon peer-to-peer (P2P) structure, blockchain eliminates the need of trusted third parties and has the features of decentralization, trustworthiness, and anonymity. According to the permissions of blockchain nodes, current blockchains can be divided into three types: the public blockchain (which is an open public system that can be partaken by any entities), the private blockchain (which is totally controlled by a single entity), and the consortium blockchain (which is maintained by several privileged entities with limited permissions to normal participants). As the soul of blockchain techniques, there exist several consensus algorithms; typically, the Proof of Work (PoW) is a very fair but costly hashrate-based algorithm fitting for public blockchains, while the Proof of Stake (PoS) is a stake-based algorithm, and the Proof of Authority (PoA) is an efficient and economical authority-determined algorithm often used in consortium blockchains. Besides, the smart contract is another important part of the blockchain. It is a set of promise codes that may be triggered for automatic execution when deployed on the blockchain. And the transactions of execution results will be generated and verified by all blockchain miners so that they will be appended on the blockchain trustworthily. Obviously, the booming of smart contracts makes the blockchain a functionally rich technology. Based on the outstanding features of blockchain technologies, some people started to use blockchain for content sharing. Kishigami et al. [20] proposed a digital content distribution system based on the blockchain. The content owner named licensor shares the data content with licensees over the Internet and all the transactions are recorded on the blockchain. Using blockchain for content distribution can guarantee certain security. With the support of such blockchain-based platforms, users can obtain rich services, while service developers can also control the deliveries of content-sharing services. This is the purpose of most schemes using blockchains for content distribution because a decentralized platform always gives users more freedom. Fotiou and Polyzos [21] presented a decentralized name-based security mechanism that aims to secure content distribution on the blockchain architecture. They leveraged Hierarchical Identity Based Encryption (HIBE) to solve the problem of content storage and verification. And the data content was divided into many small parts for flexible management. Similarly, Decent [22] also uses a similar method for managing data in chunks. It splits the data into multiple pieces before sending it to consumers. However, this system does not take the rationality of data into account, such as whether the data is tampered or not. These approaches that utilize the decentralization characteristics of the blockchain for service sharing ignored the importance of service reliability. Apparently, due to the lack of appropriate security mechanism, the clients are exposed to risks as they may receive unexpected malicious services. Some relevant solutions based on blockchain techniques were proposed to improve the reliability of service sharing. Xu et al. [23] proposed an integrity-checking-based blockchain approach to improve the security of data sharing. They transmitted the personal data in an off-chain manner and stored the corresponding hash value on the blockchain. However, it depends on users' own subjective judgment to decide whether the obtained services are secure or not, thereby causing many subjective controversies. Zhou et al. [24] proposed a protocol named CSSP (cleanroom security service protocol) based on the consortium blockchain to provide network software services. Instead of using the PoW algorithm, it uses an arbitration node to mediate and record transactions which saves a lot in mining. However, these approaches are

only imperfect mitigations for security problems. When it comes to our blockchain-based service provisioning scheme, it implemented an off-chain service delivery, dynamic on-chain verification mechanism, to help the lightweight IoT clients get rid of insecure services and service providers, without the participation of traditional trusted third party.

The approaches mentioned above are mainly designed for the desktop environment. When it comes to the IoT scenarios, although blockchain and smart contracts have been introduced for improving the security of IoT systems [34–36], there are few practices using blockchains for achieving secure service sharing due to the limitations on the hardware and software in IoT environments. Boudguiga et al. [25] used blockchain as a platform to provide service updates for IoT devices. There are three entities in this system: manufacturer node, user node, and innocuousness checking node. Before manufacturers providing update service for clients, the innocuousness checking nodes will download the updates from the blockchain to check the innocuousness. And then they will respond with a message indicating whether the update is problematic or innocuous. The clients will not be allowed to download the update until more than half of the checking points prove that the current update is innocuous. This approach also makes use of an arbitrator node to ensure the reliability of services. Usually, these authority-determined consensus algorithms (i.e., the PoA algorithm) are used in the consortium blockchain which is maintained by several privileged entities. There also exist some typical studies which used the PoA-based consortium blockchain in the IoT scenarios [26]. Undoubtedly, the success of the PoA-based consortium blockchain is quite inspiring.

Except for security, efficiency is another important issue for service sharing on the IoT platform. Due to the limitations of hardware resources, IoT devices are not capable of performing too many service tasks. We can refer to some effective desktop methods in the IoT scenarios. The works [21, 22] reduced the pressure on a single data transfer by delivering content in chunks. Herbaut and Negru [27] divided regions on the blockchain by smart contracts; each contract manages a part of the edge users and content providers. This approach reduces the burden of content transfer on a single service node. Sharma et al. [28] proposed an edge-cloud architecture implemented as the blockchain system for service sharing in IoT environment. In this approach, the close fog nodes are responsible for service delivery for IoT devices. And all the services are stored in the blockchain cloud, thereby achieving the low-cost service access control. A similar approach is also proposed by Dorri et al. [29]. However, these architectures did not improve the performance from the perspectives of blockchain itself as well as the consensus mechanism. On the contrary, our platform took advantage of the PoA-based blockchain which achieves the high throughput and low latency of the entire system.

In conclusion, comparing with the existing solutions, our blockchain-based service provisioning scheme implements an off-chain service delivery and dynamic on-chain verification mechanism to help the lightweight IoT clients get rid of insecure services and service providers, without the participation of traditional trusted third party. Our approach uses smart contracts to help the lightweight IoT clients check the validities of the acquired services and corresponding edge servers which significantly reduces the costs on the side of IoT devices. Besides, our system employs the efficient consortium blockchain with the PoA consensus engine which achieves the high throughput and low latency of the entire system.

3. Blockchain-Based Secure Service Provisioning System

In this section, we provide an overview of the secure service provisioning framework and then detail it in terms of its validity management and verification businesses.

3.1. Overview of the Model. The blockchain-based secure service provisioning framework builds on the edge transparent computing model and is working in on-chain and off-chain collaboration mode, as shown in Figure 2.

It consists of both the legacy entities of edge transparent computing and several new entities of blockchain system.

The Legacy Entities of Edge Transparent Computing

(i) *Cloud Service Provider* (CSP): The CSP is the powerful cloud-tier service provider which provides the trusted service codes to ESPs in an off-chain manner. There exist several CSPs which belong to different organizations in the system and each CSP may consist of several cloud servers.

(ii) *Edge Service Provider* (ESP): The ESP is the off-chain weak service provider close to the LCs. It is able to cache the service programs from the CSPs and deliver them to the LCs when requested. The ESPs and their services are not always dependable. Devices such as laptops and routers are usually acting as the ESPs in practice.

(iii) *Lightweight Client* (LC): The LC is the terminal which is eager to request and execute the service codes from the service providers. The LCs are abstractions of the physical IoT devices.

The Entities of Blockchain System

(i) *Arbitration Node* (AN): The ANs are privileged nodes in the consortium blockchain and maintain a distributed ledger together which records smart contracts and transactions of the validities of ESPs and service codes. The ANs are responsible for initiating transactions of validities, verifying the candidate block, and executing smart contracts. All the ANs work in the PoA consensus mode in which each AN packages and broadcasts new block in turn while the others vote to reach a consensus according to the plurality (more than 50%) rule. In our approach, each CSP acts as an AN in the blockchain network (the *AN*s are deployed on legacy cloud servers in practice).

(ii) *Lightweight Node* (LN): The LN is the less privileged entity which is only allowed to read the information on the blockchain and trigger the smart contract

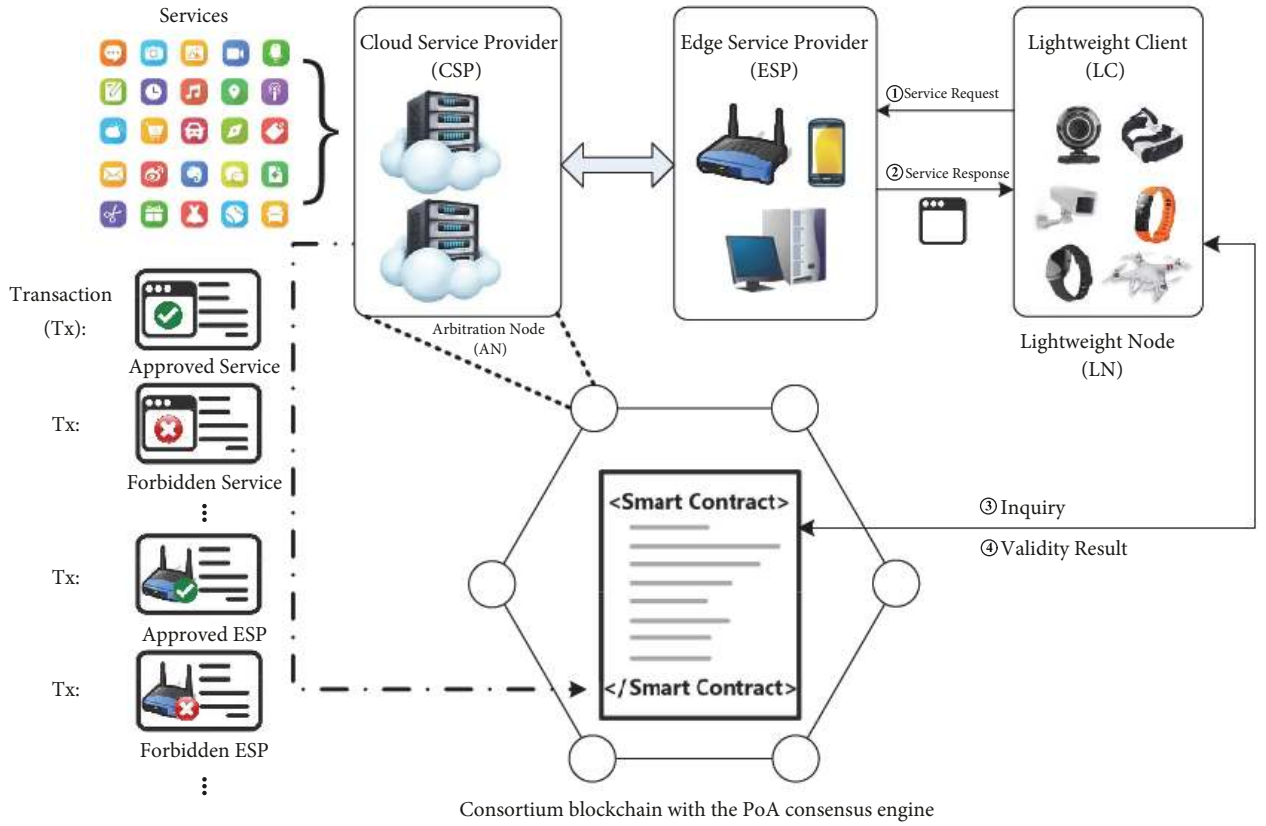


Figure 2: The secure service provisioning in edge transparent computing.

to query the validity states about the ESPs or the acquired service codes. Each LC is also an LN in our blockchain network.

In this framework, the LCs mainly request and obtain service codes from close ESPs. For protecting the LCs from untrusted ESPs and undependable services, they are allowed to trigger the smart contract (oracle smart contract, SC_O) deployed on the blockchain to figure out the validity states of the current serving ESPs or the acquired service codes according to existing validity transactions, so that the LCs can determine whether to execute the service programs or not. Besides, to keep the validity states of the ESPs and service codes updated, the ANs would continually append the new transactions of validity into the blockchain.

In our approach, the service business is off-chain while the security business is on-chain, which helps to achieve the security with low-performance overheads. Additionally, the blockchain is implemented as a consortium blockchain with the PoA consensus engine for performance reasons as well.

Next, we describe the major businesses of our approach, namely, the validity maintenance and verification in detail.

3.2. Validity Maintenance Business. The ANs keep the validity states of the ESPs and service codes updated by continually appending the new transactions of validity into the blockchain. The transaction structures are given as follows:

(i) The validity transaction of ESP: $Tx_E = \langle ESP_{ID}; V; C; T\rangle$,

where ESP_{ID} is the MAC address of the ESP, V is the validity state, C is the comments, and T is the timestamp.

(ii) The validity transaction of service: $Tx_S = \langle S_{NAME}; S_{HASH}; V; C; T\rangle$,

where S_{NAME} is the service name, $S_{HASH} = hash_{Keccak\text{-}256}$ (service codes), V is the validity state, C is the comments, and T is the timestamp.

By appending new Tx_E and Tx_S with corresponding validity states to the blockchain, the ANs can declare new legal ESPs and service programs, discard the existing ESPs and service programs when necessary (e.g., bugs discovered), update the service version, or even declare malicious ESPs and service programs. In addition, all the transactions and corresponding addresses will also be stored into the public database of a maintainer smart contract (SC_M) synchronously to make the transactions efficiently searchable for the smart contract SC_O (the on-chain address of SC_M is embedded in SC_O).

Notice that we reasonably assume that all the *AN*s can obtain service codes and necessary information about *ESP*s which are engaged in the transactions; meanwhile, the *ESP* authentication and service security testing are out of the scope of this work. Besides, the MAC address-based identification used in our case is an exemplary method

which can be replaced or combined with other identification mechanisms. And defending against identification spoofing attacks such as MAC spoofing are complementary to our work.

3.3. Validity Verification Business. For security purpose, when LCs request and obtain service codes from ESPs, they can trigger the oracle smart contract (SC_O) with the corresponding indices of the current ESPs or the service codes, to query the corresponding validity states. The workflow of the secure service provisioning is as follows:

(1) *LC* initiates an off-chain service request $\langle S_{NAME} \rangle$ to a close ESP.

(2) The ESP returns service codes to the *LC* in an off-chain manner.

(3) *LC* (i.e., *LN*) calculates the hash value of the service codes by the Keccak-256 algorithm and then triggers SC_O with a vector $\langle S_{NAME};\ S_{HASH};\ ESP_{ID} \rangle$.

(4) SC_O invokes SC_M for corresponding on-chain records about $\langle S_{NAME} \rangle$ and $\langle ESP_{ID} \rangle$ and then compares them with the received ones. Matching a valid record means the corresponding *ESP* or service codes are secure while matching an invalid record means the opposite. Note that an invalid result will also be given if there is no record related to $\langle S_{NAME} \rangle$ or $\langle ESP_{ID} \rangle$. Finally, SC_O outputs the result in the form of the on-chain transaction so that *LC* can make decisions accordingly.

The process of validity verification is shown in Figure 3.

3.4. Case Study. This section demonstrates an example of our approach to help people understand how it works concretely.

Assume there exists a blockchain-based secure service provisioning system which includes the following validity transactions in the blockchain:

$$\mathrm{Tx}_{E1} = \langle \textit{00-50-56-C0-00-08};\ 1;\ \text{Legitimate node};\ \textit{1539450834} \rangle,$$
$$\mathrm{Tx}_{S1} = \langle \text{Servie\_1.3};\ \text{7d7b084c0e330d734986a3a5884ad2c2af23a72e90ea06e8691849c64bbc64f9};\ 0;\ \text{Legitimate service};\ 1539454312 \rangle. \quad (1)$$

Then, we assume that *LC* initiates an off-chain service request <Servie_1.3> to a close ESP. Then, the ESP returns service codes to *LC* in an off-chain manner. After receiving the service codes from the ESP, the *LC* calculates the hash value of the service codes by Keccak-256 algorithm and then triggers SC_O with a vector before execution:

$$\langle \text{Servie\_1.3};\ \text{7d7b084c0e330d734986a3a5884ad2c2af23a72e90ea06e8691849c64bbc64f9};\ \textit{00-50-56-C0-00-08} \rangle. \quad (2)$$

SC_O invokes SC_M for the latest on-chain records about <Servie_1.3> and <*00-50-56-C0-00-08*> and then compares them with the received vector from the *LC*. Since the valid records Tx_{E1} and Tx_{S1} are found in the blockchain, SC_O outputs the valid result in the form of on-chain transaction (*cf*. Figures 4(a) and 5(a)). Finally, the *LC* finds the result transaction and believes that corresponding *ESP* and service codes are secure.

On the contrary, if the vector submitted to SC_O from the *LC* is <Servie_1.3; 56608f2ed0cdcf51ba6a99b2718aab4d2-e74ff78acdfa64ee8290 37be50b2cef; *E0-94-67-D4-1C-7D*>, SC_O outputs the invalid result in the form of on-chain transaction (*cf*. Figures 4(c) and 5(b)) because no valid record can be found in the blockchain. Therefore, the *LC* finds this result transaction and then denies the service codes from the unreliable *ESP*.

4. Analysis and Evaluation

This section demonstrates the security of our approach and then analyzes the experimental results in terms of effectiveness and efficiency.

4.1. Security Analysis

4.1.1. Threat Model. We assume that the adversary can set illegal ESPs to provide malicious or vulnerable service codes for attacking clients. Besides, the benign ESPs may also provide illegal services, e.g., outdated unpatched codes, due to the improper maintenance, thereby putting clients at risks. However, the adversary can neither compromise the majority of arbitration nodes to tamper the blockchain system nor forge digital signatures without corresponding private keys, which is the basic security assumption of general blockchain network commonly accepted. Note that defending against identification spoofing attacks such as MAC spoofing attacks on *ESP*s are out of the scope of this work as the MAC-based identification used in this approach is only an exemplary method which can be replaced or combined with other advanced mechanisms.

4.1.2. Analysis. Since the *ESP*s and service codes are not always reliable, our security mechanism makes use of the smart contract to check the latest validity states of edge

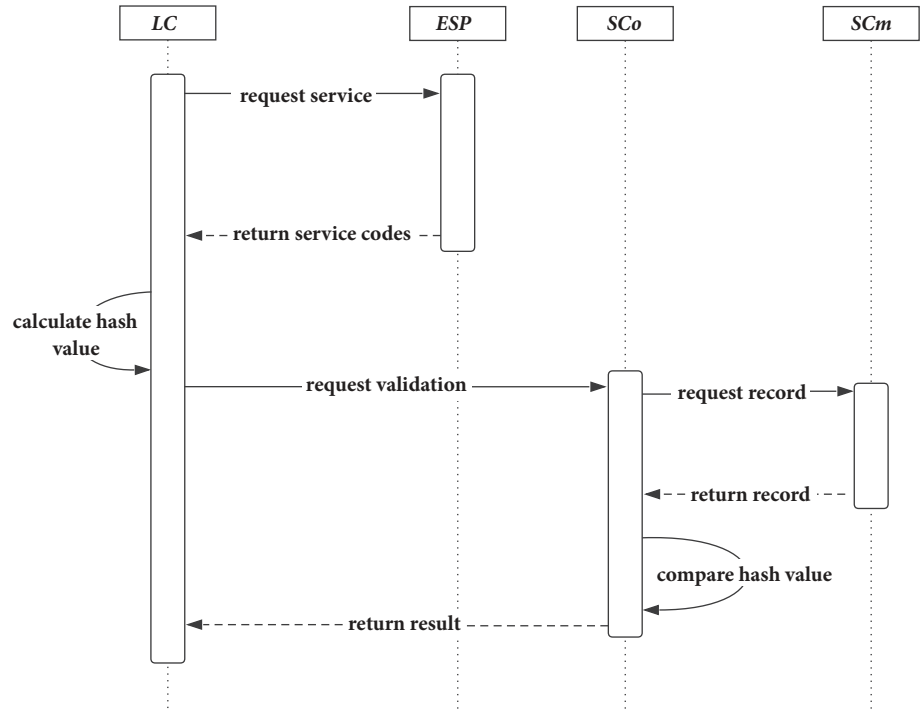
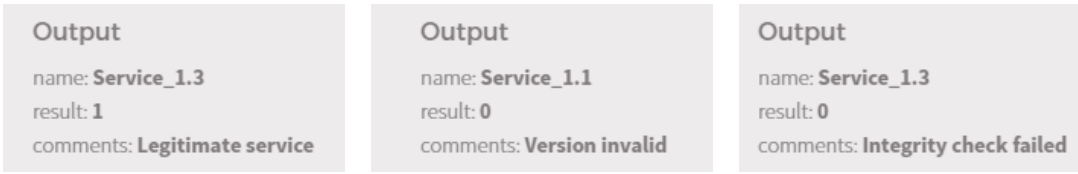


Figure 3: The workflow of validity verification.



(a) Legal service (b) Invalid version (c) Illegal service

Figure 4: The checking results of service given by the smart contract.

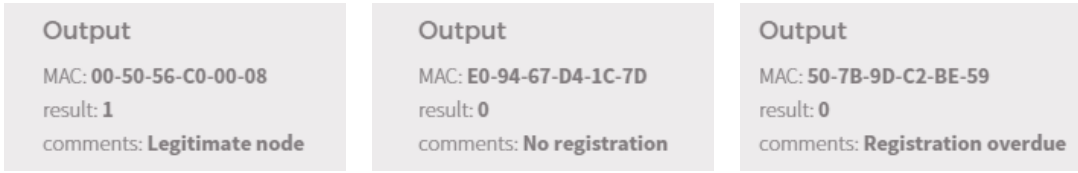
servers and service codes recorded in the form of transactions on the blockchain, so as to help the lightweight clients get rid of illegal service providers and avoid running the unknown or discarded service codes, thereby mitigating the risks. Obviously, the security of our approach mainly depends on the correctness of validity transactions and the proper executions of smart contracts.

For the validity transactions, since every transaction is publicly checked and maintained by all the distributed arbitration nodes, according to the basic security assumption of blockchain network, it is almost impossible to tamper existing transactions in blocks or package incorrect transactions into new blocks because, in the PoA consensus mechanism, the adversary can hardly compromise the majority of arbitrators (more than 50% *AN*s), which are deployed on well-maintained cloud servers. Besides, with the aid of the digital signature technique integrated into the blockchain, the adversary is unable to add malicious transactions with forged digital signatures of legal arbitration nodes because the adversary does not have corresponding private keys. Therefore, the validity transactions are trustworthy in our approach.

When it comes to the smart contracts, just like the ordinary transactions in the blockchain, they are also publicly verified and will be executed by all the arbitration nodes. Since the codes of the smart contracts are designed to be immutable, they cannot be modified after deployment even by the creators. Besides, all the execution results given by smart contracts are verified and packaged as transactions within the blocks by all arbitration nodes; therefore, these results are tamper-proof as well.

Consequently, according to the analysis above, the validity transactions of service codes and *ESP*s are trusted, and the smart contracts would be executed correctly. Therefore, the IoT clients can obtain the trusted results for security decisions, thereby getting rid of illegal service providers and insecure service codes effectively.

Additionally, as a blockchain-based approach, our security facilitates work in a decentralized P2P manner without relying on a single trusted third party and thus is more robust



(a) Legal *ESP* (b) Illegal *ESP* (c) Revoked *ESP*

Figure 5: The checking results of the *ESP*s given by the smart contract.

Table 1: The specifications of the testing devices.

Parameter	Cloud (arbitration) node (Virtual cloud server)	Edge node (Virtual laptop)	IoT node (Virtual IoT client)
CPU frequency	3.4 GHz	2.6 GHz	512 MHz
CPU core	Quad-core	Dual-core	Single-core
Network	100 Mbps	100 Mbps	100 Mbps
RAM	16 G	8 G	256 M
ROM	1 T	512 G	4 G
OS	CentOS 6.0	Fedora 12	Ubuntu Mate

against security problems like the single point of failure which can be caused by distributed denial of service attacks that often happen in IoT scenarios [17].

4.2. Experimental Evaluations. In this section, we conducted experiments to evaluate the effectiveness and efficiency of our system. We simulated cloud nodes on a single physical machine; each of them acts as an arbitration node in the consortium blockchain, and they have the highest power as miners. Also, we simulated several edge nodes to serve client nodes. There also exists virtual IoT client to request service. The details of the arbitration node, edge service node, and client node are listed in Table 1.

We use the *Ethereum Geth_1.8.11* which supports the PoA consensus mechanism to implement the consortium blockchain-based approach.

4.2.1. Effectiveness. To test the effectiveness, we simulated 20 edge nodes and 6 of them are set to provide the wrong service to the clients. Besides, we simulated 10 arbitration nodes on the consortium blockchain. When receiving the service codes, the client will calculate the corresponding hash value and then submit the result together with the identification of the edge node to the oracle smart contract SC_O for checking. Then, SC_O will query the corresponding service hash in SC_M. Finally, it puts the result on the block which can be referred by the *LC*. Therefore, the IoT device can decide to abandon the service or start to use it. For visualization purpose, we use Ethereum-Wallet's graphical interface to show the feedback from the smart contract, and Figure 4 consists of the screenshots of corresponding information of the service. As we can see, SC_O returns the query result with the help of SC_M, which includes the service name (version), the check result, and the comments. Figure 4(a) shows that the service is a legitimate service, because the hash value of the service is consistent with what the IoT device provides. And the contract will return "1" to confirm the legitimacy of the service. Figure 4(b) shows that the version of the service is invalid, and the IoT device finds "0" as the result. The edge node may not be malicious, but it has not updated the service so that the contract will identify this service as an expired service. Figure 4(c) indicates that the service provided by the edge node is completely unreliable because the integrity checking failed, and the data being transmitted is likely to be malicious and must be deprecated by the IoT node.

The contract will also check the information of the edge node. Figure 5(a) shows that the edge node is legitimate, and the result is "1." And we can see that the MAC address is also recorded. Figure 5(b) shows an illegal edge node that has not been registered on the blockchain. If the registration of an edge node is past due, it cannot be accepted and Figure 5(c) shows such information in this case.

According to the results of the experiments, our smart contracts correctly record and send back the details of the *ESP*s and service codes. The system is considered effective and the security is assured.

4.2.2. Performance. To measure the performance of our system, we conducted comparative experiments on the blockchain using the consensus algorithm of the PoA and the PoW (with the mining difficulty 0x131072), respectively. We simulated 100 IoT devices to request services from edge servers. To be more practical, we enforced the IoT clients following the Poisson probability distribution ($\lambda = 0.2 \times n$) to initiate requests. There are 20 edge nodes to provide 87 kinds of services with corresponding information recorded in the blockchain. We also simulated 10 cloud servers (i.e., arbitration nodes) whose major tasks are mining blocks and updating the validity states of various services. Through the experiment, we recorded experimental results to show the

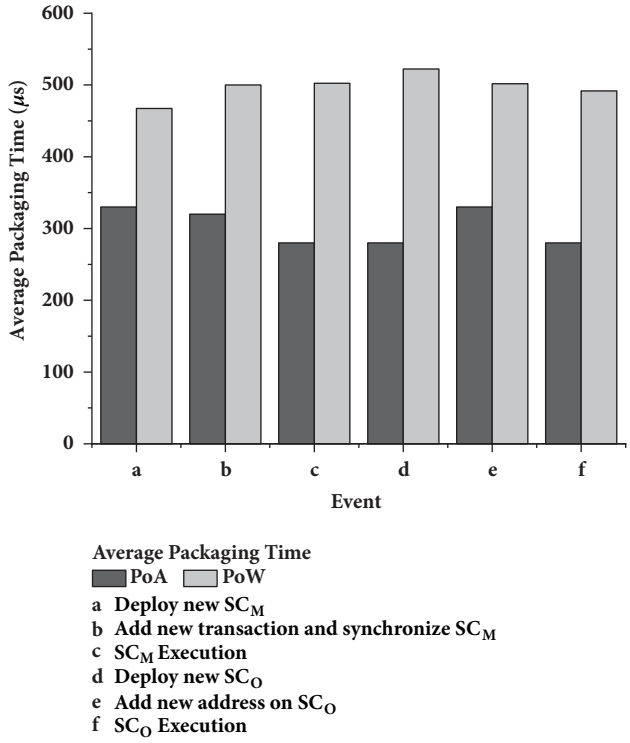


Figure 6: The average packaging time of each transaction.

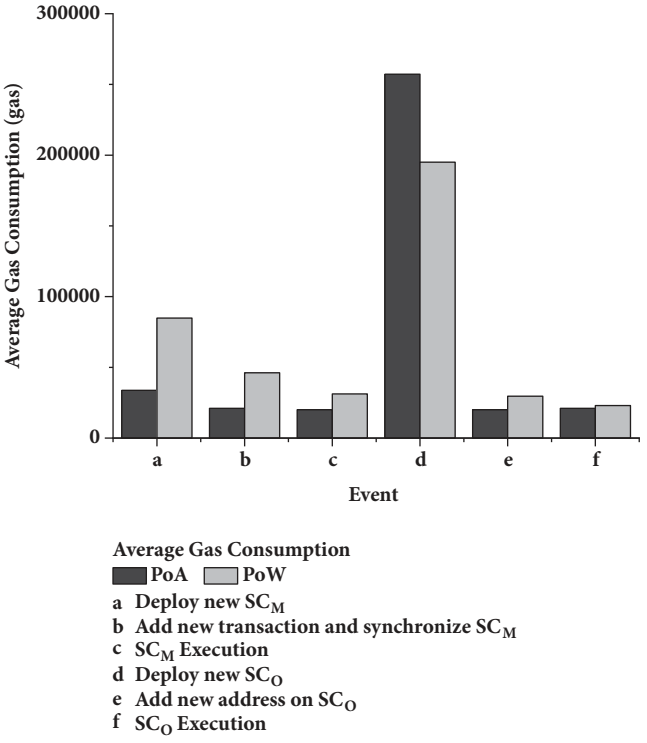


Figure 8: The average gas cost of each process.

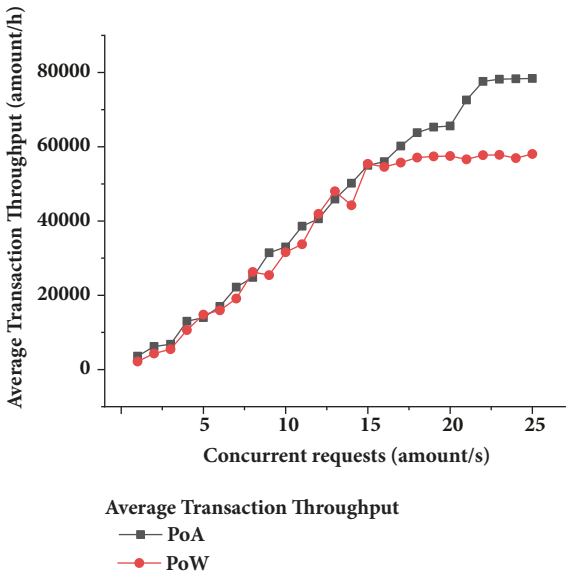


Figure 7: The throughput of transactions.

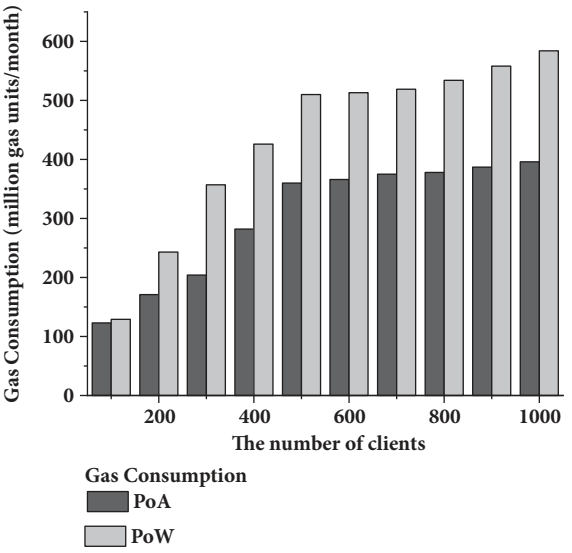


Figure 9: The gas cost under the different numbers of *LC*s in one month.

performance of the system in terms of system delays (see Figure 6), throughput (see Figure 7), and gas consumption (see Figures 8 and 9).

The packaging time is a measure of the speed of block output; to some extent, it determines how fast the system can complete transactions. As shown in Figure 6, the packaging time delay for each event is almost the same (about 500us when using the PoW and 300us when using the PoA). We can see that the packaging time under the PoA mechanism is only about 60% of that under the PoW, which shows the benefit of the adoption of the PoA in our approach. In the case of large scale throughout, the platform based on the PoA mechanism will have better performance. The experiment result shows that, in general, the packaging time for each event is small and acceptable. In particular, it is negligible to the clients in the service provisioning process.

We also conducted an experiment to measure system throughput by studying the relationship between concurrent service requests and transaction output speed. As shown in Figure 7, below a certain amount of concurrent service

requests, given a fixed period of time, the output speed of transactions increases at a certain rate along with the increase of concurrent service requests. But when the amount of concurrent requests is over 22 per second, the curve starts to converge, and the output speed of the transaction gradually tends to a stable value. The system throughput cannot increase indefinitely because it is limited by the speed of blocks creations and the capability of each block. And our maximum transaction throughput is approximately 80000 per hour. At its best, the platform can complete about 80000 transactions per hour, i.e., about 22 transactions per second, which is a relatively high and stable throughput large enough for service business. When it comes to the PoW-based approach, the transaction throughout curve converges earlier (since the concurrent request amount is 16 per second) and the maximum throughput is about 55000 per hour, which is obviously inferior to our PoA-based approach.

In the Ethereum-based blockchain, each mining node (i.e., *AN*s in our case) participating in the network will perform the blockchain protocol. With the creation of a transaction, a certain amount of gas will be charged. The gas price is the unit price of gas (e.g., 1 ether ≈ 210 USD) set by the initiator of the transaction, and the total cost of the transaction is cost (ether) = gas × gas price. Therefore, we also logged the average gas consumption of all the events in our approach, seen as shown in Figure 8. When using the PoA mechanism, event (a) costs about 33800 gas units. Events (b), (c), (e), and (f) cost about 21000 gas units per transaction. Event (d) costs around 257000 gas units. Similar results were observed from the experiment using the PoW mechanism. As we can see, the gas usage of each execution event is almost the same. But the deployment of SC_O costs much gas units than other events. We believe the reason is that the codes of SC_O are more complex than those of the others. Besides, as for the setup events, events (a) and (d) only happen once in the initialization while events (b) and (e) occur when new validity state of service codes or an *ESP* is appended. The gas price in this experiment is 0.02 ether per million gas units; therefore, it costs about 0.0058 ethers (≈ 1.2 USD) to deploy the smart contracts which are necessary to make our system functional. And as for normal transactions, we can record 1000 transactions with the cost of only 0.042 ethers (≈ 8.8 USD). We can see that the cost of our platform is relatively small and acceptable.

Besides, we also simulated n ($n = 100, 200, 300, \ldots, 1000$) IoT devices to request services from edge servers within a month and recorded the gas consumption. We also enforced the IoT devices following the Poisson probability distribution ($\lambda = 0.2 \times n$) to initiate requests. In Figure 9, the x-coordinate represents the number of *LC*s increasing from 100 to 1000 with the increment of 100. The y-coordinate represents the monthly gas consumption in the certain number of *LC*s. As shown in Figure 9, when the number of clients is 100, the monthly gas consumption is about 125 million gas units (≈ 5.25 USD per client) under the PoA, compared with 130 million gas units (≈ 5.46 USD per client) under the PoW. We can see that the gas consumption maintains a slow and stable growth along with the increase of clients. When the number of clients is 1000, the monthly gas consumption of the system under the PoA consensus mechanism is about 400 million gas units (≈ 1.68 USD per client), compared with 580 million gas units (≈ 2.436 USD per client) in the PoW-based one. Besides, the gas consumption of both the PoA and PoW relies on a stable growth, which means adding IoT devices does not impose a great overhead; therefore, our system has a low consumption and sufficient scalability.

5. Conclusions

In this paper, we proposed a novel blockchain-based secure service provisioning mechanism to protect the lightweight clients from insecure exogenous service codes from untrustworthy edge servers in the edge transparent computing scenario. We introduce the blockchain to keep all the validity states of the off-chain services and edge service providers for helping the IoT terminals get rid of undependable services through edge servers' identification and service verification. Besides, we develop and deploy the smart contracts that can be triggered by the lightweight clients to check the validities of both the service codes and edge servers according to the transactions on chain, thereby reducing the direct overheads of these resource-constrained IoT devices. Additionally, to ensure the high throughput and low latency, we adopt the efficient permissioned blockchain together with the PoA consensus engine. The security analysis and the evaluation results show that our approach protects the lightweight clients from untrusted edge service providers and undependable service codes effectively, and the validation latency is acceptable while the overheads are affordable to IoT devices.

Next, we would like to establish a blockchain-based reputation system for the service providers according to the feedbacks from IoT terminals, so as to achieve a better trade-off among flexibility, availability, and security of service provisioning. Besides, service auditing and charging are also interesting issues that can be further studied.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China [Grant nos. 61632009, 61702561, 61702562, and 61472451], the Hunan Provincial Innovation Foundation for Postgraduate [Grant no. CX2015B047], and the Guangdong Provincial Natural Science Foundation [Grant no. 2017A030308006].

References

[1] C. MacGillivray and P. Gorman, "Connecting the IoT: The Road to Success," International Data Corporation (IDC) Report, 2018.

[2] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions," in *Internet of Everything*, Internet of Things, pp. 103–130, Springer Singapore, Singapore, 2018.

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

[4] Y. Zhang, K. Guo, J. Ren et al., "Transparent Computing: A Promising Network Computing Paradigm," *Computing in Science & Engineering*, vol. 19, no. 1, pp. 7–20, 2017.

[5] 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 Network*, vol. 32, no. 1, pp. 126–132, 2018.

[6] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, 2017.

[7] H. Guo, J. Ren, D. Zhang, Y. Zhang, and J. Hu, "A scalable and manageable IoT architecture based on transparent computing," *Journal of Parallel and Distributed Computing*, vol. 118, no. 1, pp. 5–13, 2017.

[8] 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 of Things Journal*, vol. 5, no. 3, pp. 1816–1829, 2018.

[9] 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. 981–1022, 2018.

[10] Y. Zhang, L. T. Yang, Y. Zhou, and W. Kuang, "Information security underlying transparent computing: Impacts, visions and challenges," *Web Intelligence and Agent Systems*, vol. 8, no. 2, pp. 203–217, 2010.

[11] G. Wang, Q. Liu, Y. Xiang, and J. Chen, "Security from the transparent computing aspect," in *Proceedings of the 2014 International Conference on Computing, Networking and Communications, ICNC 2014*, pp. 216–220, USA, February 2014.

[12] W. Kuang, Y. Zhang, Y. Zhou, and H. Yang, "RBIS: Security Enhancement for MRBP and MRBP2 Using Integrity Check," *Journal of Chinese Computer Systems*, vol. 28, no. 02, pp. 251–254, 2007.

[13] M. Wu, "Analysis and a Case Study of Transparent Computing Implementation with UEFI," *International Journal of Cloud Computing*, vol. 1, no. 4, pp. 312–328, 2012.

[14] M. Wu, "How to Make Transparent Computing Secure – Several Security Considerations in Transparent Computing Design and Implementation," in *Proceedings of the Workshop on Trusted Computing (Guangzhou) Presentation*, 2018.

[15] V. J. Zimmer and D. Wei, "UEFI Technical Updates and Platform Innovations," in *Proceedings of the Transparent Computing Summit (Shanghai) Presentation*, 2010.

[16] V. J. Zimmer, "Platform trust beyond BIOS using the Unified Extensible Firmware Interface," in *Proceedings of the 2007 International Conference on Security and Management, SAM'07*, pp. 400–405, USA, June 2007.

[17] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: mirai and other botnets," *IEEE Computer Society*, vol. 50, no. 7, pp. 80–84, 2017.

[18] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2008.

[19] Z. Zheng, S. Xie, H. Dai et al., "Blockchain Challenges and Opportunities: A Survey," *International Journal of Web & Grid Services*, 2016.

[20] J. Kishigami, S. Fujimura, H. Watanabe, A. Nakadaira, and A. Akutsu, "The Blockchain-Based Digital Content Distribution System," in *Proceedings of the 5th IEEE International Conference on Big Data and Cloud Computing, BDCloud 2015*, pp. 187–190, China, August 2015.

[21] N. Fotiou and G. C. Polyzos, "Decentralized name-based security for content distribution using blockchains," in *Proceedings of the 35th IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2016*, pp. 415–420, USA, April 2016.

[22] M. Michalko and J. Sevcik, *DECENT Whitepaper*, DECENT Foundation Documentation, 2015.

[23] X. Xu, C. Pautasso, L. Zhu et al., "The blockchain as a software connector," in *Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016*, pp. 182–191, Italy, April 2016.

[24] L. Zhou, G. Wang, T. Cui, and X. Xing, "Cssp: The Consortium Blockchain Model for Improving the Trustworthiness of Network Software Services," in *Proceedings of the 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, pp. 101–107, Guangzhou, December 2017.

[25] A. Boudguiga, N. Bouzerna, L. Granboulan et al., "Towards better availability and accountability for IoT updates by means of a blockchain," in *Proceedings of the 2nd IEEE European Symposium on Security and Privacy Workshops, EuroS and PW 2017*, pp. 50–58, France, April 2017.

[26] 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 Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3690–3700, 2018.

[27] N. Herbaut and N. Negru, "A Model for Collaborative Blockchain-Based Video Delivery Relying on Advanced Network Services Chains," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 70–76, 2017.

[28] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[29] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and Solutions," https://arxiv.org/abs/1608.05187.

[30] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 23rd ACM Conference on Computer and Communications Security, CCS 2016*, pp. 254–269, Austria, October 2016.

[31] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, In press, corrected proof, Available online 23 August 2017.

[32] S. D. Angelis, L. Aniello, R. Baldoni et al., "PBFT vs Proof-of-authority: Applying the CAP Theorem to Permissioned Blockchai," in *Proceedings of the Italian Conference on Cybersecurity*, pp. 1–11, 2018.

[33] https://www.ethereum.org/.

[34] E. F. Jesus, V. R. L. Chicarino, C. V. N. de Albuquerque, and A. A. Rocha, "A Survey of How to Use Blockchain to Secure Internet of Things and the Stalker Attack," *Security and Communication Networks*, vol. 2018, Article ID 9675050, 27 pages, 2018.

[35] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 8th ACM Cloud Computing Security Workshop, CCSW 2017*, pp. 45–50, ACM, TX, USA.

[36] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.