Blockchain-Based Secure Time Protection Scheme in IoT

Kai Fan, *Member, IEEE*, Shangyang Wang, Yanhui Ren, Kan Yang, *Member, IEEE*, Zheng Yan, *Senior Member, IEEE*, Hui Li, *Member, IEEE*, and Yintang Yang, *Member, IEEE*

***Abstract*—Internet of Things (IoT) has been developed rapidly to make our life easier. In many IoT applications (e.g., smart homes, healthcare, etc.), all the IoT devices should be synchronized in time. However, some malicious nodes located in the IoT network can influence the time synchronization, which may interrupt the IoT system and lead to serious accidents. Therefore, it is critical and challenging to guarantee the accuracy and consistency of time during the time synchronization among all the IoT devices. In this paper, we propose a blockchain-based scheme to assure the security during time synchronization in IoT. Specifically, a publicly verifiable ledger is utilized to record and broadcast time, which can minimize many attacks from external environments. The use of multiple time sources can avoid the vulnerabilities caused by the centralized generation of accurate time. Moreover, the decentralized structure of this scheme has the advantage of adapting the changes of network topology. By employing an improved practical Byzantine fault tolerance consensus mechanism, time synchronization can be implemented efficiently. At last, the analysis results show that our proposed scheme can achieve the desired security with high efficiency.**

***Index Terms*—Blockchain, Internet of Things (IoT), security, time synchronization.**

I. Introduction

With the development of information technology, the Internet of Things (IoT) has been integrated into our lives [1]. The IoT is based on the Internet of the computer and utilizes information sensing devices (e.g., RFID, infrared sensors, global positioning systems, etc.) to connect any items with the Internet according to agreed protocols, so as to realize intelligent identification, localization, tracking, monitoring, and management, construct a kind of Internet covering all things in the world, and provide all kinds of information services for supply chain enterprises over the Internet [2], [3]. It means that IoT allows the objects collection and data exchange. In many IoT applications, such as smart grid and healthcare system [4]–[6], time synchronization is a critical and fundamental issue for data exchange and system operation. For example, the real-time health records of patients must be uploaded to a server in a chronological order [7]. Otherwise, incalculable losses may be caused due to wrong information provision to medical staffs. According to the above-mentioned facts, setting up a global system time in IoT is crucial for the collaboratively accomplishment of the tasks in IoT [8].

Although a number of time synchronization methods have been proposed, many of them focus on the problem of improving precision and/or efficiency of time synchronization [9], [10]. However, few of them consider the security issues in the time synchronization process, which are very important but also challenging due to the relatively open IoT environment where many malicious nodes or devices exist in the network. Some normal nodes may utilize a wrong time to achieve time synchronization due to a lack of ability to verify the correctness of the messages received. Even worse, these normal nodes would broadcast the wrong time to other nodes. The result is that a small number of attackers can affect the clock synchronization of the whole network [11]. Therefore, it is desirable to design a secure time synchronization scheme for IoT systems.

Toward this problem, several approaches have been proposed in the state-of-the-art. One approach is based on constructing trees in the networks as in the flooding time synchronization protocol (FTSP) [12]. This type of method consists of centralized protocols, which relies on some reference nodes [13]. In particular, the root of the tree is time source, which is responsible for sending packets to the descendant nodes in the tree. However, such methods may not be robust against faults in nodes, especially the reference nodes. It may have wrong effects on all subsequent nodes once an error occurs during the time synchronization.

Another approach can be defined as distributed time synchronization protocols [14]. This form of network topology attains the synchronization through data exchange among neighbor nodes. Different from the one-way transmission of information in tree networks, distributed network ensures that

Manuscript received June 15, 2018; revised September 8, 2018; accepted September 25, 2018. Date of publication October 5, 2018; date of current version June 19, 2019. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802300, in part by the National Natural Science Foundation of China under Grant 61772403 and Grant U1401251, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2017JM6004, in part by the Fundamental Research Funds for the Central Universities, and in part by the National 111 Program of China under Grant B16037 and Grant B08038. An earlier version of this paper was accepted at the Blockchain 2018 Conference. *(Corresponding author: Kai Fan.)*

K. Fan, S. Wang, Y. Ren, Z. Yan, and H. Li are with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China (e-mail: kfan@mail.xidian.edu.cn; 390313624@qq.com; 1043671593@qq.com; zyan@xidian.edu.cn; lihui@mail.xidian.edu.cn).

K. Yang is with the Department of Computer Science, University of Memphis, Memphis, TN 38152 USA (e-mail: kan.yang@memphis.edu).

Y. Yang is with the Key Laboratory of the Ministry of Education for Wide Band-Gap Semiconductor Materials and Devices, Xidian University, Xi'an 710071, China (e-mail: ytyang@xidian.edu.cn).

Digital Object Identifier 10.1109/JIOT.2018.2874222

the nodes can interact with multiple neighbors within communication ranges. This approach has advantages in terms of robustness and scalability, as adding and removing nodes does not change the system behavior and does not cause any overhead, such as rebuilding the tree. However, in distributed systems, the transmission of time is not public. That is, it cannot be verified. Once some of the nodes tamper with time information, it will affect the process of time synchronization of the whole system.

To assure the security of time synchronization in IoT, we employ the blockchain technology as the underlying structure. Benefit from the tamper-proof and consensus mechanism of the blockchain, malicious nodes cannot disguise the error time into a time to synchronize in the network. The publicly verifiable distributed ledger ensures that any device in the network can take the initiative to synchronize the time and ensure that the synchronization time is correct. In consortium blockchain, nodes that want to join the union must be authenticated. In other words, all the nodes involved in transferring time are friendly. Under this setting, even if the device that needs to synchronize time is malicious, we can also guarantee the security of the time synchronization. In addition to preventing malicious nodes from propagating error time, how to find the nodes that use the error time in time is also one of the problems to be solved in the future [15], [16].

The main contributions of this paper can be summarized as follows.

1) We propose a secure scheme based on blockchain to solve the problem of time announcement in IoT. The distributed ledger brings convenience to verify and synchronize time in the network.
2) The design of multiple time sources can avoid single point of failure.
3) By analysis and comparison, the superiority on security, overhead, and convergence time of the scheme is displayed.

The rest of this paper is organized as follows. In Section II, we review the related work of blockchain and problem statement about time synchronization. The overview of our scheme is presented in Section III. And in Section IV, the detailed designed algorithms of secure model are introduced. Section V analyzes the security of the system in detail and shows the results of performance evaluation. Finally, Section VI concludes this paper and illustrates future extensions.

II. Related Work

In this section, we review the related work about time synchronization and introduce the blockchain.

A. Precision Time Protocol

There are various time synchronization protocols in IoT system [17]–[22]. Precision timing protocol (PTP) is also called IEEE 1588 Precision clock synchronization protocol. It is designed for local systems, which require accuracies beyond those using network time protocol (NTP). IEEE 1588 PTP protocol draws on NTP technology, with the advantages of easy configuration, fast convergence and less network bandwidth, and resource consumption. The nodes in the PTP domain perform clock synchronization according to a certain master–slave relationship. The master–slave relationship is relative. The node device that releases the time is called the master node. One device may synchronize the clock from the upper node device at the same time and then distribute the clock to the lower node device. The basic principle of synchronization includes the recording of time information sent and received. Moreover, a timestamp for each piece of information is attached. With time records and timestamp, the slave node can calculate the offset, compensate time, and synchronize to the master clock. However, it is the timestamp that can be utilized by malicious nodes in network. If slave nodes need to go through multiple hops to synchronize to the master node, they cannot resist the attacks from malicious timestamps. Similarly to other distributed systems, Maroti *et al.* [23] presented a robust FTSP for low communication bandwidth, which scales well for medium sized multihop networks and is robust against topology changes and node failures. The protocol especially tailored for applications requiring stringent precision on resource limited wireless platforms. However, the security of data usage is not satisfactory and malicious timestamps can easily attack the system. Liu *et al.* [24] investigated the time synchronization and localization problems in underwater sensor networks solving long propagation delay and transmission delay, low bandwidth, energy constraint, mobility, etc. Zhao *et al.* [25] investigated fixed-time and finite-time synchronizations of multilinks complex network. However, these schemes do not take full account of the risk of data security. Dowling *et al.* detailed a secure authenticated version of NTP, called ANTP, to protect against desynchronization attacks. It is designed to minimize server-side public-key operations by infrequently performing a key exchange using public key cryptography, then relying solely on symmetric cryptography for subsequent time synchronization requests [26]. Symmetric cryptography is the core of scheme security, so in fact, it also determines the scheme's ability to resist attacks is limited. Therefore, we need provide a solution to avoid the effects of malicious timestamps.

B. Blockchain

Blockchain technology was originally used as a decentralized and de-trusted infrastructure for encrypting digital currency [27]. Blockchain is linked together by hash values on the block. This kind of chain structure is usually used to verify and store data. It can be said that blockchain uses distributed consensus algorithms to generate and update data, uses cryptography to ensure data transfer and access security, and uses smart contracts consisting of automated scripting code to program and operate data. The data in blockchain ledger has the characteristics of nontampering and publicly verifiable. Moreover, these properties of blockchain provide possibility to accomplish the time synchronization in IoT [28]–[31]. All of the devices in the network can attain the time recorded in the public ledger. This method can minimize the effects from malicious nodes because the time source need not send messages

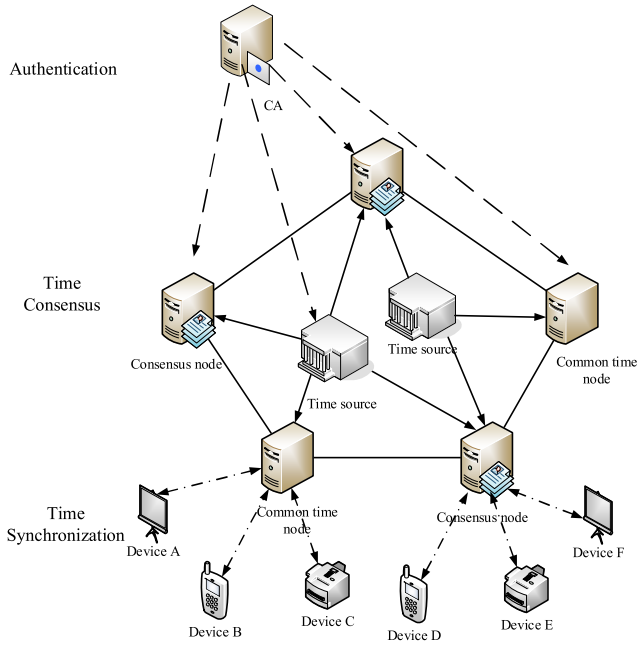

Fig. 1. Architecture of system.

attached timestamp to the slave nodes to synchronize. On the other hand, the blockchain can be classified as public blockchain, consortium blockchain, and private blockchain. Public blockchain, just as its name implies, is public to all users in system. The data stored in ledger is visible to all nodes. An overly open environment provides convenience for malicious nodes. The private blockchain is used only by private organizations. Similarly, the accounting, reading, and writing permissions of the blockchain are specified by organization rules. Therefore, the consortium blockchain has become the only choice in our scheme.

In this scheme, the consortium blockchain represents an open but limited network environment. Its consensus process is controlled by some preselected nodes, which resist a number of attacks from outsiders and insiders. In addition, it is convenient to increase or decrease the consensus nodes while keeping the network robust. If a normal node, which stores blockchain ledger collapses, the devices nearby still can facilitate time synchronization. What they need do is reconnecting with the other nodes, which own a ledger.

III. System Architecture

As shown in Fig. 1, a time synchronization system consists of five modules: 1) CA; 2) time source; 3) consensus node; 4) common time node; and 5) IoT devices.

A. CA

CA is both a system administrator and an identity management agency. Its main responsibility is to verify the identities of the nodes that apply to join the federation and promptly remove malicious nodes from the system to enhance the security of system. This ensures the honesty of nodes existing in system to the utmost extent. CA itself is not involved in the process of time synchronization. Authorized nodes will obtain a digital certificate including the only identity sequence and corresponding public key for signature. When the devices are ready to access to system, the certificate contributes to verifying the legitimacy of node to be connected.

B. Time Source

As the sponsor of the consortium blockchain, the privileges it owns are highest. Simultaneously, as the initial point of the time synchronization process, time source owns the most accurate time in network. In addition to the function of calculating the time, the time source requires regular broadcast of accurate time. The system allows multiple time sources to broadcast accurate time. Authorized authenticated time sources can be applied to join the system for time synchronization. Under normal circumstances, the accurate time issued by different time sources should be the same and the time nodes can be synchronized according to the actual situation. In fact, different time sources are independent in order to resist the attacks from external environment. Therefore, the process of time synchronization will not be affected even if some time sources break down.

C. Common Time Node

This kind of node can be regarded as the center of a small area in network topology. A number of physical equipment nearby facilitates clock synchronization through it. A ledger recorded the time issued by time source is stored in the server of the time node. At the same time, a physical clock is running with this node. It is the clock that measures the time offset between the latest time recorded in the ledger and the current time. In order to ensure the security and improve the efficiency of time synchronization, a black list is created in time node. The data written on the list are EID and IP address of malicious device. Time node can refuse to provide service for these malicious devices on the list.

Specifically, three parameters are essential for a device that needs to carry out time synchronization. Namely, the latest updated time on the ledger, the clock offset of the node and the network transmission time. There is no doubt that the first two parameters can be obtained from time node directly. As for the third data, it is dynamic in real time based on the network communication status. If a device is ready to synchronize time, a request is made and sent to the node, which possesses a time ledger. The request is equivalent to a probe to detect the network congestion. This promotes to obtain the current message transmission time in system.

D. Consensus Node

In our scheme, consensus node is an enhanced common time node. In other words, in addition to completing all the work of a common time node, the consensus nodes still need to reach consensus to clarify which blocks can be written to the ledger. Other time nodes determine whether to update local ledgers and time based on consensus results. The system randomly selects consensus nodes from the active common time nodes according to the consensus mechanism. Before selecting consensus nodes, the time source will broadcast a query

message. Therefore, only the common time nodes that respond in a timely manner and have enough devices connected will have the opportunity to become a consensus node before the deadline. Each view, namely, each round to add block, the consensus nodes are different.

Time source sends current accurate time blocks to consensus nodes that are responsible for accounting at this view. Consensus nodes reach consensus through the practical Byzantine fault tolerance (PBFT) protocol. If the consensus is reached, all time nodes will add blocks to the ledger and attaches the signature generated by its own private key. This mechanism conduces to minimize the cost of maintaining network security, maximize network performance, minimize the cost of running the network (bandwidth, CPU, etc.).

The items included in the block are as follows.

1) *Current Hash:* The SHA256 hash of the current block including block header and block body.
2) *Previous Block Hash:* The hash value of previous block, which is used to connect and verify.
3) *Consensus Node ID:* The unique identity of consensus node, which is used to verify.
4) *Consensus Node's Signature:* The digital signature of the consistent node is used to ensure the source and authenticity of the data.
5) *View ID:* Current view number.
6) *Current Time:* Accurate time to be synchronized.
7) *Time Difference:* Time difference between previous local time and accurate time.

E. Devices

The devices are the normal smart devices, such as RFID terminals, smart home appliances, pilotless automobiles, smart phones, and so on. All of them have the requirement for time synchronization. At the same time, many sensors are integrated in them including the sensor for timing. The local time provided by sensor is used directly outside the period of time synchronization. If it is necessary for smart devices to reduce the offset between local time and absolute time, they can read the blockchain ledger and make requests to the time node on their own initiative. Moreover, update their own local time to facilitate time synchronization eventually. The advantage of this method is the terminals in network have autonomy to decide when to synchronize or whether to synchronize, which also improves the network congestion significantly.

IV. Algorithm Design

In this section, we describe the detailed algorithms used in our proposed scheme.

A. Initialization of System

In Algorithm 1, it is easily to understand the initialization of system. First, CA needs to issue digital certificate to all nodes. Moreover, the time source must exist. That is to say, we choose a node as time source. It can be connected to other nodes and its main responsibility is to transmit current time to consensus nodes of each view. All nodes are initialized as normal time nodes. Next, consensus nodes are selected from normal time nodes randomly. In the process of initialization, time nodes are still in the initialization state. Therefore, the rules selecting consensus nodes are not suitable here. It is the reason that we choose consensus nodes randomly.

Algorithm 1 Initialization of Timing System

```
if the node is time source then
verifyNode()
sendMesToConsNode()
else
nodeType ← common time node
end if
selectConsNode() ← random
```

Algorithm 2 Selection of Consensus Node

```
input: a set N of time nodes in the network
input: T, the periodicity of selection
input: q, the quantity of devices that the consensus node need to be connected
if currentTime()%T == 0 then
foreach n ∈ N
    numOfDevConted ← submit()
consensusNodes ← compare(q)
end if
currentConsNode ← choose(N)
broadcast() ← currentConsNode
verifySig() ← normal time node
confirmBlock()
```

B. Selection of Consensus Node

The process of selecting consensus nodes is essential in our scheme. It is these consensus nodes that accomplish the tasks of adding block. Namely, they update the latest and most accurate time generated by time source. In Algorithm 2, we know that the selection is held on a regular basis. For each view, a large number of nodes participate in the system operation at this time. Whether nodes are active or not is the criterion of choice. If it is active, we can think that it can be selected as a consensus node. Therefore, only the active time nodes that have enough devices connected can become consensus nodes. A group of consensus nodes is selected at this time. Each view, the system will generate random numbers through natural sources such as white noise and generate a number of consensus nodes in all selected time nodes according to the result of random number generation. The number of consensus nodes is usually between 10 and 16. After the accounting node broadcasts message to other time nodes, they will verify the signature and confirm the new block. Eventually, the latest time is successfully written into the blockchain ledger.

C. Process of Time Consensus

The time source sends the accurate time to the consensus nodes. The consensus nodes set the data of accurate time to the cache state and wait for the consensus result. Then according to the PBFT protocol, if 1/3 of the consensus nodes recognize the time to be authentic, they will write it to the ledgers. Otherwise, the data is discarded. In addition to the accurate time t_{time}, the consensus node also records the offset time t_{off} of system delay after receiving the accurate time. At last, the

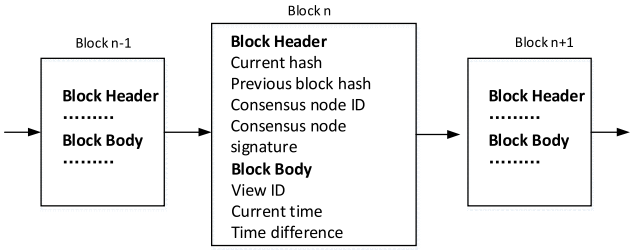

Fig. 2. Constitution of block.

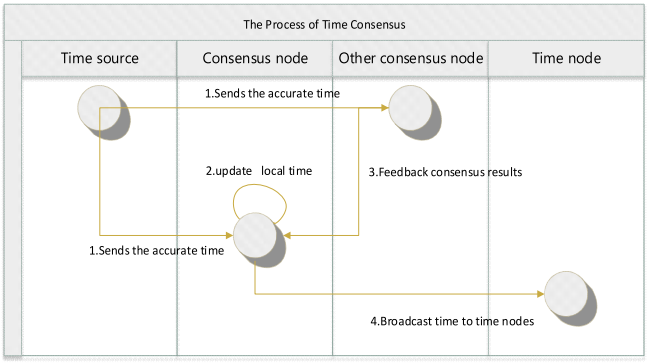

Fig. 3. Process of time consensus.

Algorithm 3 Process of Time Consensus

select consensus nodes < 13;
if consensus nodes is active **then**
  time source connect the consensus nodes;
else
broadcast reselection of new consensus nodes
return line 1
  end if
if the connection between consensus and time source exists **then**
verify() $\leftarrow$ *consensusnode*
else
  consensus $\leftarrow$ *sendtime*() $\leftarrow$ *timesource*
  $t_{time} \leftarrow$ *cachestate*() $\leftarrow$ *consensusnode*
  $t_{off} \leftarrow$ *sendOffsetToDev*() $\leftarrow$ *consensusnode*
  $t_{delay} \leftarrow$ *dectTransTime*()
  updateTime$(t_{time} + t_{off} + t_{delay})$
end if
if the consensus is reached **then**
  $t_{time} \leftarrow$ *writeinledger*()
  timenodes $\leftarrow$ *sendtime*() $\leftarrow$ *consensusnodes*
else
  $t_{time} \leftarrow$ *discard*() ;
end if

nodes detect the network congestion status and calculate the messages transmission time t_{delay}. The sum of three parameters is latest local time t_{local}. Other time nodes get accurate time and update ledger from consensus nodes. It is worth mentioning that latest local time synchronization with accurate time is not affected by consensus speed. This mechanism greatly guarantees the accuracy of time synchronization. The process is shown in Fig. 3.

Step 1: Time sources are independent of each other. In principle, there is only one time source that can synchronize time in one view. The time source generates regularly the standard time and sends the time to the consensus nodes selected by the system in current view.

Step 2: When the consensus node receives the standard time, it will instantly calculate the latest local time and set the data of accurate time to the cache state. The time synchronization process and the consensus process are separate. Even if the consensus speed is slow, it will not affect the accuracy of node synchronization time.

Step 3: Upon receiving enough acknowledgment information, the consensus node updates the local time according to the time in caching and writes the accurate time and the time offset between previous local time and accurate time to its ledger.

Step 4: Other time nodes get accurate time from consensus nodes and update local ledger.

Algorithm 4 Time Synchronization Process

send request to time node nearby
if the connection between device and time node exists **then**
  verify() $\leftarrow$ *timenode*
  if device is in black list **then**
    refuse to serve
  else
    $t_1 \leftarrow$ *readLedger*() $\leftarrow$ *device*
    $t_2 \leftarrow$ *sendOffsetToDev*() $\leftarrow$ *timenode*
    $t_3 \leftarrow$ *dectTransTime*()
      updateTime$(t_1 + t_2 + t_3)$
  end if
else
  broadcast to reconnect a new time node
  return line 1
end if

D. Process of Time Synchronization

The pseudo code of time synchronization is presented in Algorithm 4. After the preparation of system initialization and selecting consensus node, smart devices can send requests to synchronize. First, the connection between time nodes and requestors must exist. If not, the requestor needs to search for a time node nearby and establish the connection. Then the devices will read the ledger and obtain the latest time t_1. In addition, a request is sent to time node. When the time node receives the request, he/she knows that some devices want to update their time. Then, he/she will return the real-time offset records t_2. At last, another function of the request is to detect the network congestion status and calculate the messages transmission time t_3. Moreover, the sum of three parameters is current time. Smart devices will utilize this time to update their local time and accomplish the process of time synchronization.

V. Security Analysis

This section, we will focus on the discussion of security analysis of our system model. Analyzing the overall security of the system, we can start with several types of attack.

A is a node, B is a ledger server, K is defined as a public–private key, m is defined as a signed message.

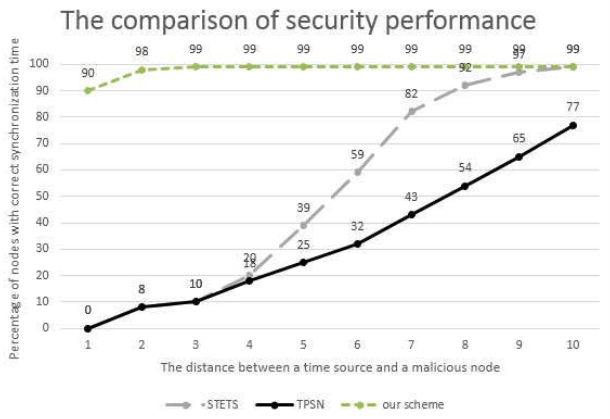

Fig. 4. Comparison of security performance.

Authentication Stage:

$$\frac{\text{B Received } m_1 \text{Signed With } K_a^{-1}, \text{ x in } m_1, \text{ B IsTrustedOn } K_a}{\text{B Can Prove(A Says x)}} \tag{1}$$

Reception Stage:

$$\frac{\text{A Receives } m_2 \text{ Signed With } K_b^{-1}; \text{ A Can Prove(K Authenticates B)}}{\text{A Can Prove(B Says } m_2)} \tag{2}$$

$$\frac{\text{A IsTrustedOn B; A Can Prove(B Says } m_2); \text{ A Can Prove(B IsTrustedOn } m_2)}{\text{A Can Trust } m_2}. \tag{3}$$

If the adversary wants to send false accurate time to common time nodes, it will be judged as false information due to the lack of authentication information of ledger server. Time stamps can be used to determine the time of message generation. So that the system can resist identity disguise and binding attack.

Time node can refuse to provide service for these malicious devices based on a black list, which is created in time node. The data written on the list are EID and IP address of malicious device, which is used to prevent DDoS of malicious device.

It may cause the worst effect if the neighboring nodes of time source are malicious nodes in the process of time synchronization. Therefore, we assume that some neighboring nodes of time source are malicious. Then analyze the security performance of our scheme through simulation results. We compare the results with STETS [22] and TPSN [21]. The result is shown in Fig. 4.

Many neighboring malicious nodes will continuously broadcast the wrong time to the subsequent nodes, which makes more and more nodes in the system synchronize with the wrong time. The closer the node is to the time source, the more the number of affected nodes will be. When the nodes adjacent to the time source are malicious nodes, the correct accurate time will be difficult to synchronize. Moreover, it is difficult to detect and correct errors in time. Yet our scheme has done well. The main reason for this result is that we choose blockchain for time synchronization. Only the authorized node can participate the alliance. Moreover, a unique key pair is issued for signature, which ensures the correct time can be written in the block. Even if the malicious node is selected as the consensus node and the time in the time ledger is tampered with, the time in the ledger is publicly verifiable. It is possible to prevent malicious nodes from tampering with time.

The security analysis results of several attacks are shown in Table I.

TABLE I
LIST OF SYSTEM RESISTANCE ATTACK

SCHEME	IDENTITY DISGUISE	DDOS	BINDING ATTACK	MALEVOLENT TAMPERING
Our scheme	Y	Y	Y	Y
TPSN	Y	N	Y	N
STETS	Y	N	Y	N

VI. PERFORMANCE ANALYSIS

A. Overhead Analysis

In this section, we evaluate the overhead of our scheme by calculating the number of exchanging messages in the process of time synchronization. We assume N is the total number of nodes in blockchain. M_{blk} is used to represent the number of all messages to synchronize all nodes in our scheme. In the process of synchronization, time source sends a new time to consensus node at first. After verifying the signature, the consensus will broadcast the new message to all nodes. In other words, every node receives one message in the synchronization process. As a result, the number of exchanging messages in blockchain is N

$$M_{blk} = N. \tag{4}$$

Therefore, we use the same method to calculate the number of messages to synchronize all nodes in STETS. Nodes are divided into backbone nodes and passive nodes. Backbone nodes are similar to the nodes in our scheme. N is the number of backbone nodes. In addition, M_{STEST} represent the number of messages to synchronize all nodes. In STEST, every backbone node needs to complete three information exchanges to synchronize. However, the passive nodes are ignored since they only receive messages and do not broadcast messages. Thus, the total number of exchanging messages in STEST is obvious

$$M_{\text{STEST}} = 3N. \tag{5}$$

Compared with the traditional time synchronization method, obviously, the overhead of system can be reduced according to the analysis.

B. Storage Overhead Estimation

In our scheme, we have redesigned a new block for time blockchain. Considering the synchronization efficiency and storage overhead, the block size should be as small as possible on the foundation of security. Otherwise, the huge overhead of storing blockchain ledger may impose a burden on the time node. Through estimation, we can calculate the size of time block. Table I shows the size of individual structure of time block.

TABLE II
COMPONENT SIZE OF BLOCK

Structure Name	Size in Bytes
Current Hash	32
Previous block hash	32
Consensus node's signature	256
Consensus node ID	4
View ID	4
Current time	4
Time Different	16

TABLE III
STORAGE OVERHEAD ESTIMATION

Space Occupied by Time Blocks Generated in a Period of Time				
Time interval(min)	Per day	Per month	Per year	Per 10 years
2	244.69KB	7.169MB	87.22MB	872.2MB
5	97.876KB	2.867MB	34.888MB	348.88MB
15	32.625KB	978.75KB	11.629MB	116.293MB
30	16.313KB	489.375KB	5.815MB	58.145MB

According to Table I, we know the time block is 348 bytes. In order to estimate the storage overhead, we still need to obtain the synchronization frequency. The higher the time synchronization frequency, the more time nodes in blockchain network can update time on ledger. As a result, devices in IoT can also calibrate local time with less time offset. However, higher sync frequency means more blocks will be generated. Considering the continuity of the time synchronization, the ledger will become huge. In short, the improvement of time accuracy will bring higher storage overhead. The following formulas are used to estimate time nodes' storage overhead.

$S(B \times ft)$ used to calculate the data growth size in time t.

S represents the hard disk space required for a certain number of time blocks. B represents the block size, 348 bytes. ft represents the number of blocks added on blockchain in time t.

First, we assume time source sends absolute time to consensus node every 10 min. Based on this, we perform the following calculations.

1) $S(348 \times 6) = 2088$ bytes ≈ 2.039 KB per hour.
2) $S(348 \times 6 \times 24) = 50\,112$ bytes $= 48.938$ KB per day.
3) $S(348 \times 6 \times 24 \times 365) =$ bytes ≈ 17.444 MB per year.

The results are optimistic. New added blocks occupy storage space at the megabytes (MB) level each year. With current storage capabilities, it can be said these storage spaces almost do not burden time nodes. Using the example shown earlier as guide, the frequency of time synchronization is dynamically adjusted. Table II represents the total size of data in per day, per month, per year, and per ten years. It is obvious that the data sizes generated are at level of MB even if time source updates time every 2 min. It means that storage of blockchain ledger will not be bottleneck of this system.

C. Convergence Time

Considering that the consensus process and the convergence process are two independent processes. The results provided by the simulation show that when the number of nodes and devices increases, it leads to the increase of the time convergence for our scheme. We assume that the normal neighboring

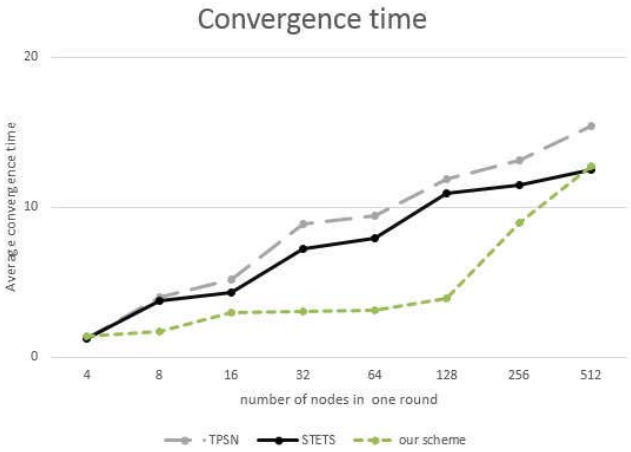

Fig. 5. Comparison of convergence time.

nodes can only synchronize with four nodes, while in our scheme time nodes can synchronize with the 200 nodes or devices at most simultaneously. When the number of nodes is less than 200, the distributed time synchronization protocol requires multiple hops between nodes to complete the time synchronization. Moreover, the more hops are needed, the longer the time it takes. Ideally, our scheme only need one round to achieve time synchronization, which makes our scheme converge faster. When the time node and devices are excessive, the broadcasting efficiency of time node will become one of the key factors limiting convergence time. If the number of nodes requiring synchronization time in a round is more than 200, the time node needs another round to synchronize time for the remaining nodes. The convergence time becomes longer and the advantage becomes less obvious. The result is shown in Fig. 5.

VII. CONCLUSION

In this paper, we propose a secure scheme based on blockchain to solve the problem of time announcement in IoT. In this distributed network, a closed blockchain to record and broadcast time is utilized, which minimize attacks from external environments. The use of multiple time sources can effectively avoid centralization. Even though one or more time sources are crashed, devices in network can still synchronize accurately. Moreover, this scheme has the advantage of adapting the changes of network topology. We have designed the block structure in order to send few messages in process of time synchronization, which reduces communization overhead. By employing PBFT consensus mechanism, time synchronization can be implemented efficiently. Our future work will focus on how to improve the accuracy of time and reduce the offset as much as possible. Besides, the consensus mechanism is also a direction of improvement. Time synchronization scheme can be more effective and secure by the consensus mechanism, which balances efficiency and security.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[2] E. Xu, Z. Ding, and S. Dasgupta, "Target tracking and mobile sensor navigation in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 1, pp. 177–186, Jan. 2013.
[3] G. Xu, W. Shen, and X. Wang, "Applications of wireless sensor networks in marine environment monitoring: A survey," *Sensors*, vol. 14, no. 9, pp. 16932–16954, 2014.
[4] S. Sengupta, S. Das, M. Nasir, A. V. Vasilakos, and W. Pedrycz, "An evolutionary multiobjective sleep-scheduling scheme for differentiated coverage in wireless sensor networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1093–1102, Nov. 2012.
[5] Z. Su, Y. Hui, and Q. Yang, "The next generation vehicular networks: A content-centric framework," *IEEE Wireless Commun.*, vol. 24, no. 1, pp. 60–66, Feb. 2017, doi: 10.1109/MWC.2017.1600195WC.
[6] Q. Yang, A. Lim, S. Li, J. Fang, and P. Agrawal, "ACAR: Adaptive connectivity aware routing for vehicular ad hoc networks in city scenarios," *J. Mobile Netw. Appl. Archive*, vol. 15, no. 1, pp. 36–60, Feb. 2010, doi: 10.1007/s11036-009-0169-2.
[7] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "MedBlock: Efficient and secure medical data sharing via blockchain," *J. Med. Syst.*, vol. 42, p. 136, Aug. 2018.
[8] K. Fan *et al.*, "Secure time synchronization scheme in IoT based on blockchain," in *Proc. IEEE Blockchain Conf.*, Jul./Aug. 2018.
[9] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1352–1364, Jun. 2011.
[10] Y. Liu, J. Li, and M. Guizani, "Lightweight secure global time synchronization for wireless sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Shanghai, China, Apr. 2012, pp. 2312–2317.
[11] R. Poovendran, C. Wang, and S. Roy, *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks* (Advances in Information Security). New York, NY, USA: Springer, 2007, pp. 395–408.
[12] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd ACM Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 39–49.
[13] K. S. Yildirim and A. Kantarci, "Time synchronization based on slow-flooding in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 244–253, Jan. 2014, doi: 10.1109/TPDS.2013.40.
[14] J. He, J. Chen, P. Cheng, and X. Cao, "Secure time synchronization in wirelesssensor networks: A maximum consensus-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 4, pp. 1055–1065, Apr. 2014.
[15] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.
[16] L. Li, K. Ota, and M. Dong, "When weather matters: IoT-based electrical load forecasting for smart grid," *IEEE Commun. Mag.*, vol. 55, no. 10, pp. 46–51, Oct. 2017.
[17] L. Schenato and F. Fiorentin, "Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks," *Automatica*, vol. 47, no. 9, pp. 1878–1886, 2011.
[18] R. Solis, V. S. Borkar, and P. R. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *Proc. 45th IEEE Conf. Decis. Control*, 2006, pp. 2734–2739.
[19] M. Akhlaq and T. R. Sheltami, "RTSP: An accurate and energy-efficient protocol for clock synchronization in WSNs," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 3, pp. 578–589, Mar. 2013.
[20] W. Dong and X. Liu, "Robust and secure time-synchronization against Sybil attacks for sensor networks," *IEEE Trans. Ind. Informat.*, vol. 11, no. 6, pp. 1482–1491, Dec. 2015.
[21] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, Los Angeles, CA, USA, Nov. 2003, pp. 138–149.
[22] T. Qiu, L. Chi, W. Guo, and Y. Zhang, "STETS: A novel energy-efficient time synchronization scheme based on embedded networking devices," *Microprocessors Microsyst.*, vol. 39, no. 8, pp. 1285–1295, 2015.
[23] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," Inst. Softw. Integr. Syst., Vanderbilt Univ., Nashville, TN, USA, Rep. TR #: ISIS-04-501, pp. 1–15, 2004.
[24] J. Liu, Z. Wang, J.-H. Cui, S. Zhou, and B. Yang, "A joint time synchronization and localization design for mobile underwater sensor networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 3, pp. 530–543, Mar. 2016, doi: 10.1109/TMC.2015.2410777.
[25] H. Zhao *et al.*, "Fixed-time synchronization of multi-links complex network," *Mod. Phys. Lett. B*, vol. 31, no. 2, 2017, Art. no. 1750008.
[26] B. Dowling, D. Stebila, and G. Zaverucha, "Authenticated network time synchronization," in *Proc. USENIX Security*, 2016, pp. 823–840.
[27] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[28] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. Security Privacy Workshops (SPW)*, 2015, pp. 180–184.
[29] M. Swan, "Blockchain thinking: The brain as a decentralized autonomous corporation," *IEEE Commun. Soc. Mag.*, vol. 34, no. 4, pp. 41–52, 2015.
[30] Z. Zheng, S. Xie, H. N. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," Working Paper, 2016.
[31] K. Fan, Y. Ren, Y. Wang, H. Li, and Y. Yang, "Blockchain-based efficient privacy preserving and data sharing scheme of content-centric network in 5G," *IET Commun.*, vol. 12, no. 5, pp. 527–532, Mar. 2018.

Kai Fan (M'16) received the B.S., M.S., and Ph.D. degrees in telecommunication engineering, cryptography, and telecommunication and information system from Xidian University, Xi'an, China, in 2002, 2005, and 2007, respectively.

He is a Professor with the State Key Laboratory of Integrated Service Networks, Xidian University. He has published over 70 papers in journals and conferences. His current research interests include IoT security and information security.

Shangyang Wang was born in Shandong, China, in 1991. He received the B.S. degree in information security from Xidian University, Xi'an, China, in 2016, where he is currently pursuing the master's degree with the State Key Laboratory of Integrated Service Networks.

His current research interest includes blockchain.

Yanhui Ren was born in Henan, China, in 1992. He received the B.S. degree in telecommunication engineering from the Changan College of Xidian University, Xi'an, China, in 2015, and the master's degree from the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, in 2018.

His current research interest includes blockchain.

Kan Yang (M'17) received the B.Eng. degree in information security from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2013.

He is an Assistant Professor with the Department of Computer Science and the Associate Director of the Center for Information Assurance, University of Memphis, Memphis, TN, USA. His current research interests include security and privacy in cloud computing, big data, Internet of Things, information-centric network, and distributed systems.

Dr. Yang was a recipient of the Outstanding Research Thesis Award from the City University of Hong Kong for his Ph.D. degree.

Zheng Yan (SM'14) received the D.Sc. degree in technology from the Helsinki University of Technology, Espoo, Finland.

She is currently a Full Professor with Xidian University, Xi'an, China, and a Visiting Professor and the Finnish Academy Research Fellow with Aalto University, Espoo. Her current research interests include trust, security, and privacy; data mining; mobile applications and services; and social networking and cloud computing.

Prof. Yan was a recipient of the number of Outstanding Leadership Awards for IEEE conference organization, the 2017 IEEE ComSoc TCBD Best Journal Paper Award, the Outstanding Associate Editor of 2017 for IEEE ACCESS, and the EU Eureka Excellence Award in 2017. She serves as an organizational and technical committee member for over 80 international conferences and workshops. She is an Associate Editor of the IEEE INTERNET OF THINGS JOURNAL, *Information Fusion*, *Information Sciences*, IEEE ACCESS, *Journal of Network and Computer Applications*, *Soft Computing*, IEEE BLOCKCHAIN NEWSLETTER, and *Security and Communication Networks*. She is a Founder Steering Committee Co-Chair of IEEE Blockchain conference. She is organizing and has organized over 10 conferences, such as IEEE Blockchain 2018, NSS/ICA3PP/IEEE CIT2017, IEEE TrustCom/BigDataSE/ISPA-2015, and IEEE CIT2014.

Hui Li (M'14) was born in Shaanxi, China, in 1968. He received the B.S. degree in radio electronics from Fudan University, Shanghai, China, in 1990, and the M.S. and Ph.D. degrees in telecommunications and information system from Xidian University, Xi'an, China, in 1993 and 1998, respectively.

He is currently a Professor with Xidian University. His current research interest includes network and information security.

Yintang Yang (M'17) was born in Hebei, China, in 1962. He received the Ph.D. degree in semiconductors from Xidian University, Xi'an, China.

He is currently a Professor with the Key Laboratory of the Ministry of Education for Wide Band-Gap Semiconductor Materials and Devices, Xidian University. His current research interests include semiconductor materials and devices, and network and information security.