ChinaRxiv

AI translation · View original & related papers at
chinarxiv.org/items/chinaxiv-202009.00117

A Blockchain-Based Scalable Digital Forensics Model (Postprint)

Authors: Sun Jingchao

Date: 2020-09-28T00:00:00+00:00

Abstract

Blockchain technology can enable electronic evidence to break free from the constraints of centralized institutions, thereby improving the efficiency and trustworthiness of evidence storage. To address the issues of poor scalability and usability of existing blockchains in electronic forensics, we design a hybrid on-chain and off-chain blockchain-based electronic evidence collection model based on Delegated Proof of Stake-Practical Byzantine Fault Tolerance. First, block producers are elected through voting via the Delegated Proof of Stake method, and then synchronization is performed exclusively among these block producers using the Practical Byzantine Fault Tolerance mechanism. Given that blockchain technology itself is not suitable for storing large-capacity data, a hybrid on-chain and off-chain approach is adopted for data storage. Security and efficiency analyses of the proposed model are conducted, demonstrating that the model can simultaneously satisfy security and performance requirements and effectively support electronic forensics.

Full Text

Preamble

Vol. 38 No. 3 *Application Research of Computers* (ChinaXiv Cooperative Journal)
Accepted Paper: Research on Scalable Digital Forensics Model Based on Blockchain

Author Affiliation: School of Criminal Investigation & Forensic Science, People's Public Security University of China, Beijing 100032, China

Abstract: With the decentralized anti-tampering feature of blockchain technology, digital evidence is being freed from the constraints of centralized institutions, and the efficiency and credibility of evidence deposition have been

improved. To address the shortcomings of weak scalability and availability in existing blockchain-based digital forensics applications, this paper proposes a parallel storage blockchain digital forensics model that combines Delegated Proof of Stake (DPoS) with Practical Byzantine Fault Tolerance (PBFT). First, block producers are selected through a voting process based on delegated proof of stake, and then the Practical Byzantine Fault Tolerance mechanism is used for synchronization exclusively among these block producers. Given that blockchain technology itself is not suitable for storing large-capacity data, the model employs a hybrid on-chain and off-chain storage approach. Security and efficiency analyses of the proposed model demonstrate that it simultaneously satisfies both security and performance requirements, making it well-suited for digital forensics applications.

Keywords: blockchain; digital evidence; digital forensics; consensus algorithm

0 Introduction

Digital evidence represents a novel form of evidence that has emerged as a prerequisite condition for the development of electronic computer technology. While academic interpretations of the concept of digital evidence vary widely, its profound impact is undeniable, permeating numerous aspects of daily life [1]. However, the inherent susceptibility to tampering has hindered both the practical application of digital evidence and in-depth theoretical research [2], ultimately limiting its evidentiary capacity [3].

In 2008, Nakamoto [4] constructed a decentralized electronic currency—Bitcoin —and elaborated in detail on its underlying blockchain technology, characterized by decentralization, immutability, and traceability. Blockchain technology has since attracted significant attention from various sectors [5][6], yielding abundant research achievements in domains such as network public opinion [7], information protection and prediction [8], and open resource acquisition [9]. Current academic research on digital evidence predominantly focuses on its formation and authenticity verification [10, 11], objective acceptance [12], and certification standards [13], with certain accomplishments in applying blockchain to construct digital forensics models. Huang et al. [14] proposed a blockchain-based cloud computing forensics model that employs Merkle trees for evidence preservation and an improved consensus algorithm to reduce block generation time. Hou et al. [15] presented a complete blockchain-based electronic evidence system and described a batch packaging method for digital evidence to improve deposition efficiency. Cebe et al. [16] constructed a blockchain infrastructure for vehicular network forensics, providing comprehensive digital evidence services for accident investigations. Bonomi et al. [17] developed a blockchain-based chain of custody for physical evidence, ensuring auditable integrity and owner traceability. Ryu et al. [18] proposed a blockchain-based digital forensics framework for IoT environments to address the heterogeneity and distributed char-

acteristics of IoT environments as well as the centralization issues in existing forensic investigations.

While blockchain-based digital forensics applications in various scenarios have achieved certain results, most existing approaches adopt Byzantine fault tolerance algorithms for consensus. As the number of nodes increases, communication costs rise dramatically, affecting scalability in terms of users and transaction volume as well as overall system availability. To address these limitations, this paper designs a blockchain-based digital forensics model that combines on-chain and off-chain storage with a hybrid Delegated Proof of Stake-Practical Byzantine Fault Tolerance consensus mechanism. Block producers are first selected through delegated proof of stake voting, and then only these producers synchronize using the Practical Byzantine Fault Tolerance mechanism. To overcome blockchain' s inherent unsuitability for storing large-capacity data, the model employs a hybrid on-chain and off-chain storage approach.

1.1 Blockchain Technology

Blockchain technology implements a distributed, replicated ledger across peer-to-peer networks, originally developed for Bitcoin cryptocurrency. A blockchain consists of a series of blocks containing transaction records, with transactions within each block ordered chronologically. Upon receiving a sufficient number of transactions, block-creating nodes package them to create new blocks. Once packaging is complete, a consensus process begins to persuade other nodes to include the block in the blockchain. Blockchain utilizes cryptographic digital hashing and signature techniques, with all transaction content being public, including transaction addresses and specific content.

1.2 Consensus Mechanism

The consensus mechanism constitutes a core component of blockchain. For instance, Bitcoin' s consensus mechanism is Proof of Work (PoW), which grants block creation rights through solving cryptographic puzzles. This mechanism offers significant security on-chain (tolerating up to 50% malicious nodes) but at the cost of substantial computational and time resources. Consensus mechanisms have undergone considerable development, with several mainstream schemes currently employed in blockchain systems:

Proof of Work (PoW): In PoW, nodes must solve mathematical puzzles to reach consensus. No computational shortcuts exist for solving these puzzles; only exhaustive enumeration can produce hash results meeting the required criteria. Blockchains relying on PoW consensus suffer from slow transaction confirmation and high resource consumption.

Proof of Stake (PoS): The most common alternative to PoW is Proof of Stake. In PoS, the right to create new blocks depends on who holds more cryptocurrency, with a deterministic algorithm selecting nodes based on their

token holdings. Consequently, miners need not expend costly computational resources to create blocks; the probability of a miner being selected to create a block depends on the proportion of currency they hold in the system.

Delegated Proof of Stake (DPoS): DPoS is a variant of PoS where nodes vote to select witnesses. Witnesses are nodes chosen through voting to verify transactions, with each node able to vote for trusted witnesses. Voting power is determined by the number of shares each voter holds. Top-tier witnesses are responsible for verifying transactions and creating blocks, earning rewards for fulfilling their duties. DPoS is designed as a democratic voting implementation aimed at protecting blockchains from centralization and malicious use through voting and election processes. Cryptocurrencies using DPoS include Lisk, Steem, EOS, and BitShares.

Practical Byzantine Fault Tolerance (PBFT): PBFT is an algorithm tolerant of Byzantine faults, featuring low algorithmic complexity and good practicality in distributed systems. It can operate in asynchronous environments containing Byzantine faults, guaranteeing distributed system availability even when up to one-third of participants are compromised.

2.1 On-Chain and Off-Chain Combined Digital Forensics Model

The blockchain-based digital evidence model is illustrated in Figure 1. [Figure 1: see original paper]

The specific steps are as follows:

a) Employ forensic techniques to secure digital evidence and record time and geographical location.

b) Generate hash digests of the collected electronic data using public keys.

c) Sign the generated hash digest, recorded time, and location information with a private key.

d) Select accounting nodes through a consensus algorithm.

e) Selected accounting nodes perform comprehensive sorting of data based on multiple factors including block size and transaction waiting time.

f) Accounting nodes package the sorted data into new blocks.

g) Accounting nodes broadcast the generated new blocks.

h) Representative or miner nodes verify the correctness of broadcast block data. If approved by the majority, accounting nodes add the generated

blocks to the main chain and upload original data to the database; if not approved by the majority, they deny its validity.

The model employs a hybrid on-chain and off-chain storage approach for two reasons. First, due to inherent blockchain limitations, evidence may be too large for efficient storage on-chain. Second, and more importantly, if evidence were stored on-chain, every node in the blockchain network could access it, whereas in reality only authorized nodes should obtain evidence. Therefore, the model stores only evidence hashes on-chain, with specific evidence stored in an evidence database.

The evidence database is a conventional database or file repository. Considering the diversity of electronic data, a non-relational database is appropriate. Original digital evidence is stored together with an identifier ID, which is obtained as a hash value of the evidence combined with a nonce to ensure uniqueness. This database is distributed and managed by trusted entities. Evidence logs are stored on the blockchain, implemented through blockchain technology to store for each piece of evidence its ID, hash digest, record time, location information, description, submitter (creator) identity, and complete ownership history (including current time). Although the evidence itself is not stored on-chain, the ID can verify that evidence has not been tampered with as long as it is generated using a robust cryptographic hash function.

The network can be decomposed into two node groups: validator nodes and lightweight nodes. Validator nodes primarily function to store blockchain copies, verify transactions, create and propose blocks, and add them to the chain (i.e., participate in the consensus protocol). Such nodes must be preventively authorized as validators in a permissioned blockchain. Lightweight nodes can be considered chain clients, as they only issue transactions and rely on validators to add and verify their transactions.

2.2 Forensics Algorithm Design Based on Delegated Proof of Stake-Practical Byzantine Fault Tolerance

The principle of Delegated Proof of Stake allows nodes holding stakes to vote for block creators. This voting approach enables stakeholders to delegate block creation rights to representatives they support rather than creating blocks themselves. Figure 2 [Figure 2: see original paper] shows the flowchart of the block-producing node algorithm in the forensics model. As illustrated, if selected representatives fail to generate blocks normally, they are dismissed and stakeholders choose new nodes to replace them. DPoS fully utilizes shareholders' voting power to reach consensus in a fair, democratic manner.

a) Stake-holding nodes vote, and high-vote nodes become representative nodes.

b) Representative nodes collect data, package it, and create blocks.

c) If block creation succeeds within the given time, the block is broadcast to other nodes for verification; otherwise, the representative node is dismissed.

Representative nodes have numerous obligations. If a representative node is incompetent or crashes, other nodes can revoke their votes, causing it to lose representative status. Voting rankings are recalculated each cycle, and among the top-voted representative nodes, the system first shuffles the order, then sequentially produces blocks. Representative nodes can claim rewards after fulfilling their duties. Every coin holder has voting and candidacy rights, with voting power related to the amount of currency held and holding duration. Having representative nodes selected through votes to package transactions ensures both election efficiency and decentralization.

In DPoS, stakeholders select witnesses responsible for generating and adding blocks to the blockchain. Voting for witnesses is a continuous process, and a reputation scoring system helps stakeholders better evaluate witness quality. Failure to generate blocks during allocated time slots results in witnesses being skipped, negatively impacting their reputation. Consequently, witnesses are motivated to perform at the highest standards or risk losing their positions.

The block-producing node algorithm in the forensics model only determines who can send blocks to the root chain and when. Before a block enters the root chain, it must be possible to achieve consensus among all validators. To this end, the Practical Byzantine Fault Tolerance algorithm is introduced into the node verification process of the forensics model.

PBFT consists of five phases: request phase, pre-preparation phase, preparation phase, commit phase, and reply phase. Figure 3 [Figure 3: see original paper] illustrates how PBFT operates, where node 0 is the primary node and node 3 is a faulty node. The primary node forwards messages from the client to the other three nodes. With node 3 having crashed, a message undergoes five phases to achieve consensus among nodes, concluding when the client receives replies from nodes. PBFT ensures nodes maintain a common state and take consistent actions in each consensus round, achieving the goal of strong consistency and thus serving as an absolutely final consensus protocol.

Based on these principles, the forensics model node synchronization algorithm is designed, with its specific workflow shown in Figure 4 [Figure 4: see original paper].

1) Client A requests execution of state machine operation B by sending a message to the primary server. The operation to be executed is represented by the request, with timestamp ensuring exactly-once execution of client requests. Request timestamps are totally ordered, so later requests have higher timestamps than earlier ones. The client signature is represented by σ_c.

2) The selected primary node accepts the request.

3) The primary node validates the client message, verifying the correctness of the client message signature. If invalid, the request is discarded; if valid, the request is assigned a number. The format is $\langle\langle PREPREPARE, v, n, m\rangle\rangle_{\sigma_p}$, where v represents the view number of the sent message, n is the message number assigned by the primary node, m represents the message content, and σ_p represents the primary node' s signature.

4) The primary node packages the pre-prepare message and broadcasts it to other replica nodes.

5) Replica nodes validate the broadcast message from the primary node, verifying the correctness of the primary node' s signature, whether n is within the specified range, and whether different requests with the same sequence number under the same view have been accepted. If invalid, the request is discarded; if valid, the process proceeds to the next step.

6) The format is $\langle\langle PREPARE, v, n, d\rangle\rangle_{\sigma_i}$, where v represents the view number of the sent message, n is the message number assigned by the primary node, d is the message digest, i is the replica node number, and σ_i represents node i' s signature information.

7) Each replica node sends this message to other nodes, including the primary node.

8) Each node validates the received prepare message, verifying the correctness of the replica node signature, whether n is within the specified range, and whether different requests with the same sequence number under the same view have been accepted. If invalid, the request is discarded; if valid, the request is recorded.

9) When a replica node has cumulatively received $2f + 1$ valid prepare messages from different nodes, it packages a commit message and signs it. The format is $\langle\langle COMMIT, v, n, d\rangle\rangle_{\sigma_i}$, where v represents the view number of the sent message, n is the message number assigned by the primary node, d is the message digest, i is the replica node number, and σ_i represents node i' s signature information.

10) Each replica node sends commit messages to other nodes, including the primary node.

11) Each node validates the received commit message, verifying the correctness of the replica node signature, whether n is within the specified range, whether d is correct, and whether the view, sequence number, and signature match those of the prepare message. If invalid, the request is discarded; if valid, the request is recorded.

12) When a replica node has cumulatively received $2f + 1$ valid commit messages from different nodes, it executes the client request.

13) The reply message is packaged and signed, with format $\langle\langle REPLY, v, t, c, i, r\rangle\rangle_{\sigma_i}$, where v is the current view number, t is the timestamp of the corresponding request, i is the replica number, r is the result of executing the requested operation, and σ_i represents node i' s signature information.

14) Each replica node sends reply messages to the client.

15) The client validates the received prepare message, verifying the correctness of the replica node signature. If invalid, the request is discarded; if valid, the request is recorded.

16) When a replica node has cumulatively received $f + 1$ valid confirmation messages from different nodes, the result is confirmed as valid; otherwise, the request is broadcast to all replicas.

17) The client sends operation execution requests to replicas, and all non-faulty replicas execute the same operations in the same order. Since replicas are deterministic and start from the same state, all non-faulty replicas send replies with identical results for each operation. When the client receives $f+1$ identical result replies from different replicas, the result is considered valid.

PBFT is based on the State Machine Replication method, where state machines are replicated across different nodes in a distributed system. Each replica maintains service state and implements service operations. Let R represent the replica set, assuming $|R| = 3f + 1$, where f is the maximum number of potentially faulty replicas. The challenge in state machine replication is ensuring non-faulty replicas execute identical requests in the same order. PBFT uses a primary-backup technique for ordering, where replicas move through a series of configurations called views. In each view, one replica serves as the primary while others are backups, with the primary selected as $p = v \mod |R|$, where v is the view number. The primary determines the order for executing client requests, assigning a sequence number to each request and sending this assignment to backup replicas. The primary may experience various issues: assigning the same sequence number to different requests, stopping sequence number assignment, or leaving gaps between request numbers. Therefore, when the current primary fails, backup replicas check the sequence numbers assigned by the primary and trigger a view-change to select a new primary.

3.1 Security Analysis

The security of the proposed model is analyzed under the following threat scenarios.

Scenario 1: Malicious nodes exist and create a minority fork. Under normal operation, block producers take turns generating blocks at fixed intervals. With a minority of malicious nodes (less than 1/3 of the total), malicious nodes produce blocks slower than honest nodes. The chain produced by the honest majority will always be longer than the minority chain. According to

the longest chain principle, the minority fork created by malicious nodes will not be recognized.

Scenario 2: Nodes produce duplicate blocks. One case involves offline minority nodes attempting to produce their own forked chain, but since the minority fork will be shorter than the majority chain, it will not be accepted by mainstream nodes. Another case involves online nodes producing two or more alternative blocks during their assigned time slot. In this situation, the next scheduled producer can choose to build upon any of these alternatives, which does not affect construction of the longest chain. Therefore, the number of alternative blocks attempted is irrelevant.

Scenario 3: Network fragmentation. This scenario typically occurs under abnormal network connectivity, where the longest chain will be produced by the largest minority of nodes. Upon network recovery, the smaller minority will naturally switch to the longest chain, restoring clear consensus. During network anomalies, two forks may have equal length. In such cases, the smaller branch will break the tie when rejoining the network. The total number of block producers is odd, making sustained ties impossible, while shuffling of block producer order randomizes production sequences to ensure that even if two forks have equal numbers of producers, the forks will grow at different rates.

Scenario 4: Majority producer collusion. If most producers collude, they can create many forks, each growing under majority confirmation. However, in this case, the last irreversible block algorithm will cause one fork to revert to the longest chain, meaning the longest chain is still determined by the minority of honest nodes. Majority producer collusion cannot persist for long, as stakeholders will eventually vote to replace these corrupt nodes.

Scenario 5: Long-range attack. Long-range attacks refer to attackers creating a fake chain longer than the main chain in a short time. Such attacks may occur in proof-of-stake consensus methods but are prevented in delegated proof-of-stake because when users sign transactions, they base their actions on perception of multiple blocks. Blocks on isolated forks are not recognized as they lack confirmation from stakeholders and thus cannot replace the main chain.

Scenario 6: Double-spending attack. Double-spending attacks can occur in various scenarios, such as when communication failures cause blockchain reorganization and previously excluded transactions are discovered. The DPoS algorithm used in the model can monitor network health, promptly detect witness anomalies, and immediately identify any communication losses, thereby largely avoiding double-spending attacks.

3.2.1 Consensus Latency

Assuming the total number of nodes in the network is N and message propagation uses the gossip protocol, the number of messages that need to be propagated is N^2. The consensus waiting time can be estimated using formula (1), which

calculates the time required to propagate block b among validators.

The corrupted formula appears here: $N2N2()/cppppcclatencySSrSrNR = ++ ppSpScSprcrRpScSppSR$

Where S_{pp} is the size of the pre-prepare message, S_p is the size of the prepare message, S_c is the size of the commit message, r_p and r_c are the retry counts for prepare and commit messages, and R is the bandwidth of the slowest communication channel between two validator nodes. Although S_p and S_c are constant, the pre-prepare message carries block b, so S_{pp} typically depends on block size. Since R is usually a constant determined by the infrastructure connecting validator nodes, waiting time can be reduced by adjusting block size and message retry counts.

Block size is the sum of transaction sizes plus the block header size (a constant). In the proposed forensics model, the block header size is constant. The actual number and types of transactions in a block depend on many factors, including block period T and the specific transaction set sent within a given time period. Therefore, different blocks typically have different sizes, but unless network conditions are extremely poor or retry counts are excessive, consensus speed can meet processing requirements.

3.2.2 Blockchain Growth Rate

Since each block has a fixed-size header, the more blocks created, the larger the space occupied by block headers relative to transactions in the blockchain—i.e., the greater the space overhead. The block header size overhead, or the total size of block headers at any time t, can be obtained from formula (2):

The corrupted formula appears here: $HsHs()HtOHtsT = ()Itt()()()()mgmtxItStsoverheadtstx \subseteq= ++\sum gs12[,]tt21()()mmStSt - 122112(;)(;)()HtxIttttGttsstxT \subseteq -=+\sum$

Notably, this value depends only on the number of blocks in the chain at time t, not on the number of transactions. Assuming $tx(t)$ is the set of transactions included in the blockchain at time t, the total blockchain size at time t is calculated by formula (3):

The corrupted formula appears here: $()()()()mgmtxItStsoverheadtstx \subseteq= ++ \sum$

Where s_0 is the size of the genesis block. Therefore, the growth rate over time interval $[t_1, t_2]$ is given by formula (4):

The corrupted formula appears here: $12[,]tt21()()mmStSt-122112(;)(;)()HtxIttttGttsstxT \subseteq -=+\sum$

Clearly, blockchain growth speed over time depends primarily on transaction rate. Another factor affecting growth rate is block period T, which influences block header overhead and thus the first term in equation 4. Assuming new evidence is created and deleted n times per year, even with large-scale evidence

collection (1 million per year) and transfer (10 million per year), the annual growth rate remains at the gigabyte level, which is acceptable given today' s storage capacities.

3.2.3 Comprehensive Comparison with Existing Digital Forensics Models

This section analyzes the proposed model' s performance through comparison with existing blockchain-based digital forensics models and traditional digital forensics approaches, as shown in Table 1 .

The comparison reveals that blockchain-based forensics models offer better security and anti-tampering characteristics by eliminating dependence on central authorities. However, previous forensics models mostly employed Byzantine fault tolerance-based consensus mechanisms, resulting in poor scalability and throughput performance. The proposed scheme combines delegated proof of stake with traditional Byzantine mechanisms, enhancing scalability and throughput while retaining the advantages of blockchain forensics models over traditional approaches, thereby better meeting the needs of high-concurrency scenarios.

4 Conclusion

In current digital forensics investigations, maintaining data integrity is performed independently by relevant central authorities. While this offers sufficient procedural convenience, malicious attackers could compromise the integrity of potential evidence if they target these institutions. Leveraging blockchain' s decentralized anti-tampering characteristics can free digital evidence from central authority constraints, improving both evidence deposition efficiency and credibility. Existing blockchain-based digital forensics models predominantly employ Practical Byzantine Fault Tolerance algorithms and their variants that consider Byzantine faults. However, strong consistency algorithms like PBFT suffer from high complexity and insufficient decentralization, making them more suitable for scenarios with relatively few nodes.

This paper proposes a blockchain-based digital forensics model combining on-chain and off-chain storage with a hybrid Delegated Proof of Stake-Practical Byzantine Fault Tolerance mechanism. Nodes elected through DPoS achieve synchronization via PBFT, while digital evidence data is stored using a hybrid on-chain and off-chain approach to improve the scalability and availability of blockchain-based forensics models. Security and efficiency analyses demonstrate that the proposed model reduces overall network energy consumption while ensuring network security and remains robust under various natural network interruption scenarios. Future work will focus on deployment in real application scenarios to further optimize the consensus algorithm, improve throughput and transaction rates, and adapt the model to larger-scale data usage.

References

[1] Liu Pinxin. Discuss of the positioning of electronic evidence—based on the current evidence law of China [J]. Studies in Law and Business, 2002 (4): 37-44.

[2] Chang Yi, Wang Jian. On the independent status of electronic evidence [J]. Legal Forum, 2004, 19 (1): 66-74.

[3] Xu Kangding. Analysis of basic problems of electronic evidence [J]. Law Review, 2002 (3): 94-99.

[4] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [EB/OL]. 2008. https://bitcoin.org/bitcoin.pdf

[5] Wang Chuang, Zhu Meijuan. Bibliometric analysis on the research hotspots of blockchain in China [J]. Journal of Intelligence, 2017 (12): 73-78.

[6] Shang Qi, Chen Hongmei. Empirical analysis of the patent intelligence on the innovation status of blockchain technology [J]. Journal of Intelligence, 2019, 38 (04): 27-32.

[7] Zhao Dan, Wang Xiwei, Han Jieping, et al. Research on the propagation characteristics and rules of network public opinion information in blockchain environment [J]. Journal of Intelligence, 2018 (9): 127-133.

[8] Tu Ben, Zhang Liyi, Chen Jing. Research on the prediction model of information protection based on private blockchain [J]. Information Theory and Practice, 2017 (10): 110-115.

[9] Kong Fanchao. Construction and management of open access resources based on blockchain [J]. Information Studies: Theory & Application, 2019, 42 (05): 157-162.

[10] Wang Minyan. The formation and authenticity of electronic evidence [J]. Law Science, 2017 (6): 185-194.

[11] Chu Fumin. Three aspects of the authenticity of electronic evidence—an analysis of criminal proceedings as an example [J]. Chinese Journal of Law, 2018 (4): 121-138.

[12] Liu Pinxin. Confirmation and probability: objective acceptance of electronic evidence [J]. Global Law Review, 2017 (4): 109-127.

[13] Zhou Xin. Research on the certification norms of criminal electronic evidence [J]. Law Review, 2017 (6): 158-166.

[14] Huang Xiaofang, Xu Lei, Yang Qian. Blockchain model of cloud forensics [J]. Journal of Beijing University of Posts and Telecommunications, 2017, 40 (6): 120-124.

[15] Hou Yibin, Liang Xun, Zhan Xiaoyu. Blockchain based architecture model of electronic evidence system [J]. Computer Science, 2018, 45 (6): 348-351.

[16] Cebe M, Erdin E, Akkaya K, et al. Block4Forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles [J]. IEEE Communications Magazine, 2018, 56 (10): 130-136.

[17] Bonomi S, Casini M, Ciccotelli C. B-coc: A blockchain-based chain of custody for evidences management in digital forensics [J]. ArXiv Preprint ArXiv: 1807.10359, 2018.

[18] Ryu J H, Sharma P K, Jo J H, et al. A blockchain-based decentralized efficient investigation framework for IoT digital forensics [J]. The Journal of Supercomputing, 2019, 75 (8): 4372-4387.

Note: Figure translations are in progress. See original paper for figures.

Source: ChinaXiv —Machine translation. Verify with original.