Make Rental Reliable: Blockchain-Based Network Slice Management Framework with SLA Guarantee

Xinyi Luo, Kaiping Xue, Jian Li, Ruidong Li, and David S. L. Wei

The authors propose a blockchain-based network slice management framework consisting of a slice committee and three protocols: slice, audit, and dispute.

Abstract

To provide customized and high-quality network services under limited network resources, the 5G introduces the network slicing technology that divides physical networks into several logically independent virtual networks, improving the performance of network utilization. The slice management should satisfy the chief concerns of network operators and slice tenants who are the two most important participants, i.e., slice allocation for operators and Service Level Agreement (SLA) guarantee for tenants. However, for slice allocation, traditional centralized schemes cannot well support multi-operator slicing due to the lack of trust. And for SLA guarantee, existing solutions only provide global SLA based on game theory but cannot handle each dispute between operators and tenants. To solve the problems, we propose a blockchain-based network slice management framework consisting of a *slice committee* and three protocols: *slice*, *audit*, and *dispute*. With the help of the decentralization and reliability of blockchain, the proposed scheme achieves collaborative slice management among multiple operators with SLA guarantee. Through security and performance analysis, we prove that the proposed scheme can defend against possible dishonest behaviors of entities in the system, and is practical in terms of performance.

Introduction

Mobile communication technology has become a catalyst for social digitization, and future mobile communications are supposed to meet diverse communications needs simultaneously. For example, one business customer may require highly reliable services, while another may require high-bandwidth communication or extremely low latency [1]. The 5G technology aims to provide different mixing capabilities simultaneously to meet such diverse needs. Therefore, 5G introduces network slicing to divide a shared physical network infrastructure into multiple logical networks that could be used independently by different users [2, 3]. Such a design allows network customers to customize network slices according to their demands while enabling network operators to maximize the utilization rate of network resources [4].

Network slicing has two fundamental participants: network operators and slice tenants. Operators are resource providers that divide physical network infrastructure into virtual slices, and tenants are network customers of slices and should pay the operators. To enable network slicing, the slice management module is indispensable for collecting and matching resources and demands, monitoring slice status, guaranteeing Service Level Agreement (SLA), and charging tenants. For this purpose, Samdanis *et al.* [5] proposed the slice broker that acts as the management module. However, this kind of centralized management framework cannot sufficiently support multi-operator scenarios and is also prone to a single point of failure.

With the emergence of blockchain applications, researchers have introduced blockchain into network slicing. Blockchain plays a vital role in guaranteeing the fairness of slice orchestration and SLA. The decentralized consensus establishes trust among operators and achieves collaborative slice management, and the capacity to lock assets and reliable execution of smart contracts solve the "who-pay-first" contradiction between operators and tenants. Besides, the tamper-resistant storage of blockchain also provides an effective tool for audit and arbitration, providing fundamental conditions for correct billing and SLA guarantee. In the existing blockchain-based slice management schemes, blockchain-based orchestration [6–8] have been thoroughly discussed. Also, game theory has been utilized to enable global SLA guarantee [9, 10].

However, the issues of billing correctness and single SLA dispute remain unsolved. Specifically, existing billing schemes rely on usage reports from operators or tenants for charging without considering dishonest participants who may provide false reports. Besides, some game theory-based schemes only guarantee SLA from a holistic perspective but cannot handle every dispute between operators and tenants. To solve the problems, we propose to collect usage reports from both operators and tenants and then automatically audit their consistency through the smart contract. For consistent reports, the contract will transfer tokens to pay the bill. While for inconsistent reports, we design the slice committee mechanism and utilize the end-to-end network test to check which report is correct. Also, we design a

Digital Object Identifier:
10.1109/MCOM.001.2200665

Xinyi Luo, Kaiping Xue (corresponding author), and Jian Li are with the University of Science and Technology of China, China; Ruidong Li is with Kanazawa University, Japan; David S. L. Wei is with Fordham University, USA.

0163-6804/23/$25.00 © 2023 IEEE

dispute protocol for tenants to claim dissatisfaction with the service quality.

To summarize, we propose a secure and fair network slice management framework based on blockchain that provides correct billing and SLA guarantee. To further enhance the performance and security of our proposed framework, we introduce the Merkle hash tree (MHT) and verifiable random function technology. We prove the security and effectiveness of the proposed slice management framework by analyzing the possible dishonest behaviors in the system and prototyping the proposed framework for performance analysis.

The rest of our article is organized as below: we introduce existing schemes related to network slice management, and provide basic backgrounds. We detail the proposed slice management framework. We discuss the security and performance of the proposed scheme. Finally, we conclude the article.

Related Work

For network slicing management, Samdanis *et al.* [5] introduced the broker that collects resources from operators and assigns them to tenants according to their demands. Based on this architecture, Rost *et al.* [11] proposed a hybrid architecture that enables the coexistence of dedicated and shared slices to implement different objective functions dynamically, maximizing spectral efficiency. Ksentini *et al.* [12] proposed another framework based on eDECOR and introduced a two-level scheduler for radio resource allocation. These schemes perform well in single-operator scenarios. However, with further development, a network slice is supposed to consist of resources from multiple operators and those centralized broker-based schemes cannot provide solid trust among operators.

To solve the problem, researchers introduced blockchain to establish decentralized network slice management architecture for the multi-operator demand. Togou *et al.* [6] proposed a basic blockchain-based architecture called DBNS that utilizes smart contracts to execute the pre-established bidding-based allocation strategy. Subsequent studies improve performance and security in various aspects. For instance, Hewa *et al.* [7] proposed SNSB that optimizes the slice orchestration strategy to guarantee SLA based on the Stackelberg game model. Reference [8] adopts the DAG algorithm for selecting the optimal slice and proposes several security mechanisms, including authentication and handover. Reference [9] designs a fair incentive mechanism for slice trading. Considering the public characteristic of blockchain, He *et al.* [10] proposed NetChain that protects privacy by combining blockchain with a trusted execution environment (TEE). References [9] and [10] also adopt game theory to design incentive mechanisms for SLA guarantee. However, they only realize a long-term and global SLA but cannot handle each dispute between an operator and a tenant. For a single dispute, Zhou *et al.* [13] proposed a blockchain-based witness model to handle SLA disputes of cloud service. Zhou's scheme utilizes the smart contract to randomly select a group of witnesses to verify the current service level to handle users' disputes. However, this scheme relies on users' initiative and cannot provide automatic SLA audits. To solve those problems, we propose a blockchain-based network slicing framework that enables both automatic audit and user dispute of SLA, guaranteeing fairness and efficiency.

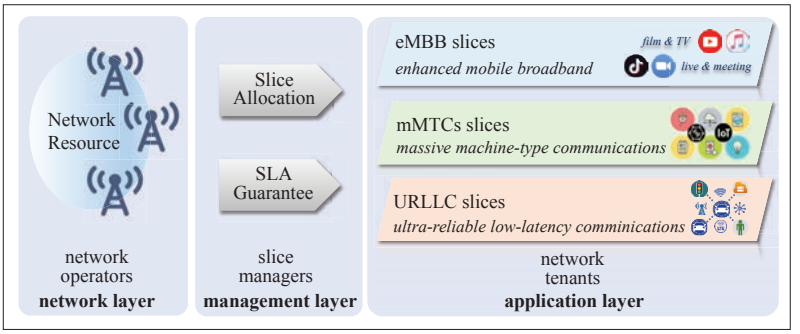

FIGURE 1. The overall architecture of 5G network slicing.

Preliminaries

This section introduces two critical technologies of our work, i.e., the concept of 5G network slicing and smart contract.

5G Network Slicing

3GPP defines the network slicing technology that enables operators to create virtual networks and customize them to provide optimized solutions for different communication scenarios with diverse needs, such as eMBB, mMTCs, and URLLC, as Fig. 1 shows. Using cloud computing and virtualization technology, operators can efficiently orchestrate the shared physical network resources into logical network slices according to different demands.

A significant problem that operators are concerned about is slice allocation, i.e., how to divide network resources will achieve maximum benefit. For tenants, one of their biggest concerns might be the SLA guarantee, i.e., whether the provided slice can meet the pre-agreed service quality [14]. We can acquire a network slice's service level (e.g., latency, throughput, or jitter) through the end-to-end network test. Besides, correct billing is a concern for both parties. Our scheme combines the slice test with smart contracts to provide SLA guarantees for a network slice.

Smart Contract

A smart contract is a computer program deployed to the blockchain and can be reliably executed by blockchain miners. Ethereum is the first blockchain that supports smart contracts, and users can deploy and invoke contracts by sending transactions to the blockchain network. Miners will collect transactions to execute relevant codes and upload the results into the blockchain. There are two important limitations of smart contracts for our scheme. First, since all the miners should execute each line of code, some heavy tasks are improper to be directly implemented through smart contracts, including the network test in our scheme. Second, blockchain cannot directly generate random numbers due to the lack of external information. Our design of the slice committee and adoption of the verifiable random function (VRF) are to solve these issues.

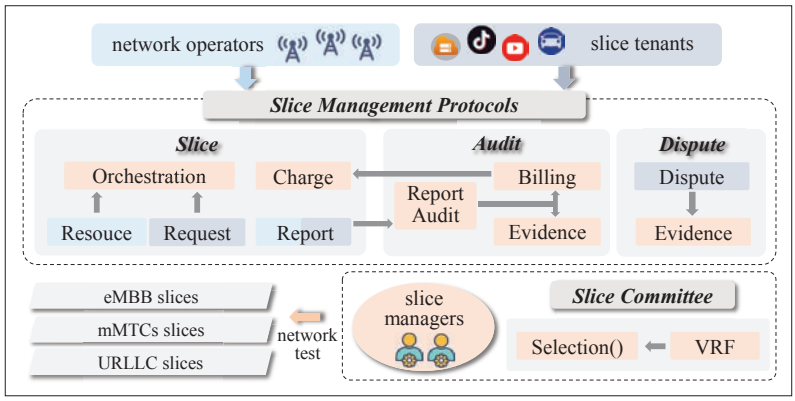

FIGURE 2. System model of the proposed slice management framework.

Proposed Slice Management Framework

In this section, we first introduce the system model and security assumptions and then provide an overview of our design. Subsequently, we explain our design of the slice committee and the three slice management protocols.

System Model and Security Assumptions

The proposed slice management framework consists of one slice committee and three slice management protocols, i.e., *slice*, *audit*, and *dispute*. Together, they provide a reliable slice management framework that enables correct billing and SLA guarantee. As shown in Fig. 2, the framework is implemented through four contracts, i.e., slice contract, audit contract, dispute contract, and slice committee contract, and details will be given later. Besides, there are three entities in the system, i.e., network operators, slice tenants, and slice managers.

- *Network operators* provide physical network resources and transfer resources to network slices to server slice tenants. They publish the resource information through the slice contract and provide slices according to the slice orchestration result.
- *Slice tenants* publish their requests to the blockchain by invoking the slice contract and wait for appropriate choices. Then, they might rent one or more slices as needed and pay the operators regularly.
- *Slice managers* are blockchain miners who undertake the slice management duty by executing the management contracts. The contracts are not required to be deployed on public blockchains. Instead, network operators and tenants with strong abilities can establish their own consortium blockchain for slice management. There are many options for the underlying blockchain, e.g., proof of Work (PoW) Ethereum, proof of Stake (PoS) Ethereum, or Hyperledger Fabric, and slice managers may determine a choice after a careful consideration.

We separately consider the security of slice service and the management framework. For the slice service, we mainly concentrate on two problems: dishonest operators and "dine-and-dash" tenants. Specifically, a dishonest operator may provide unqualified slices that do not meet the agreed service level. And a "dine-and-dash" tenant may refuse to pay usage fees to operators after using the slice. And for the management framework, the slice managers are blockchain miners, most of whom are honest. However, some might try to deceive others for their own benefits. For example, some dishonest operator miners may collude to provide false audit (the *audit* protocol), resulting in charging more fees from tenants.

Overview of the Proposed Solution

We utilize the blockchain to propose a secure and fair network slice management framework that provides correct billing and SLA guarantee. Our main idea is to periodically collect usage reports from operators and tenants and then audit them by checking whether they are consistent. A consistent result means the usage report is correct, and the bill can be directly performed according to it. If not consistent, it means one of them is cheating, and we should determine which is valid through off-chain methods, e.g., checking the current slice condition through a pre-defined test interface. However, in some cases, the above audit process might lag and cannot promptly determine if there is a quality decline. For instance, the report uploading period is relatively long. Therefore, we propose a way for tenants to claim the quality descent proactively.

For the above purpose, we design three protocols: *slice*, *audit*, and *dispute*. The *slice* protocol collects network resources and slice requests and runs the orchestration algorithm for matching resources and requests. It also maintains usage reports uploaded by operators and tenants for audit. The *audit* protocol checks the consistency of reports from operators and tenants and charges the tenants if consistent. If not, it will check the current service condition of the slice through the provided test interface to determine which one is correct for billing. Moreover, the *dispute* protocol is used for tenants to proactively claim the quality decline of a slice. The protocols are mainly implemented by smart contracts. However, considering the expensive cost to invoke the test interface and the smart contract requiring all the miners to execute each line of code, it is impractical to execute all the protocols through contracts directly. Therefore, we design the slice committee for executing network tests, reducing the cost of slice management significantly.

Slice Committee

We can implement the protocols through smart contracts directly; however, the audit and dispute protocols require executors to test slice quality through network tests. As a result, direct implementation through smart contracts will cause prohibitive costs. To solve the problem, we propose the *slice committee* to perform the network test. For convenience, we divide time into slots, and a fixed amount of slots form an epoch. A slice committee is a group of miners randomly selected from all the miners, and a particular one acts as the leader (also chosen randomly). Since on-chain random generation is a challenge, we adopt the Chainlink verifiable random function (VRF) [15] technology which is particularly designed as a smart contract random generator. We can call the Chainlink VRF to acquire a secure random number in any contract to support the subsequent function. The leader is responsible for invoking contracts when needed, as smart contracts can-

not automatically start without invocation. Based on the slice committee, only miners in the committee will invoke the test interface for network tests, and other miners will verify whether the result is provided by valid committee members. Other miners will accept it only when more than 2/3 committee members give the same test result, which will be proved to be secure later.

Specifically, the management framework consists of four smart contracts: committee contract, slice contract, audit contract, and dispute contract. The first one is used to select the slice committee, and the latter three are used to implement the *slice*, *audit*, and *dispute* protocols, respectively. During each epoch, the current leader invokes the audit contract for auditing and charging at each slot. Furthermore, at the last slot, the leader invokes the committee contract to select the committee and leader for the next epoch. The details of the three protocol contracts will be introduced later. Besides, in case the leader does not invoke the contracts as requested, an inactive leader will be punished according to the pre-determined rules, e.g., be banned from the committee or even from making profits by providing resources for a while.

Slice Management Protocols

The proposed architecture consists of three slice management protocols: *slice*, *audit*, and *dispute*. Among them, *slice* protocol undertakes slice allocation, and *audit* and *dispute* protocols are for SLA guarantee. Roughly speaking, *audit* protocol is executed by slice managers to monitor slice quality, and *dispute* protocol is invoked by tenants who found that the slice's quality has decreased.

Protocol: Slice. The *slice* protocol is responsible for slice orchestration and is implemented by the slice contract consists of five functions: `Resource`, `Request`, `Orchestration`, `Report`, and `Charge`. The former three are for slice allocation: operators and tenants separately invoke the `Resouce` and `Request` functions to upload the network resources and slice demands information. Then the `Request` function invokes `Orchestration` to divide available resources into needed slices.

- `Resource`. When an operator wants to join the system or change information, it invokes the `Resouce` function to upload its resource and other information. Then, the current slice committee validates and determines whether to allow the operator to join. If successfully joined, the corresponding resource will be recorded in the *resource* object for the following slice orchestration.
- `Request`. A tenant can request slices by invoking the `Request` function to provide his/her request and a certain amount of deposit. The `Request` function will automatically invoke the `Orchestration` function for matching requests and resources. Besides, the tenant should lock some *deposit* into the slice contract for the following charging and can also withdraw the redundant deposit later.
- `Orchestration`. This function runs optimization algorithms to appropriately match resources and requests according to several factors to maximize the effectiveness of the network slicing. There are lots of studies that propose excellent orchestration algorithms. For instance, [10] proposes a bilateral evaluation mechanism for slice orchestration.

The `Report` and `Charge` functions are used for billing. Operators and tenants invoke `Report` to upload usage reports periodically. Then slice managers audit the reports and charge tenants according to the audit result through the `Charge` function.

- `Report`. Operators and tenants invoke the `Report` function to upload the usage report of slices periodically. The reports are used for billing and auditing in the *audit* protocol. For lightening the on-chain payload, blockchain miners compress reports into two Merkle hash trees and only store the tree roots in the contract. A Merkle hash tree is a binary tree that takes the hash of reports as leaf nodes, and the hash of every two adjacent nodes is their parent. It is often used to check the consistency of two data sets quickly.
- `Charge`. The `Charge` function is automatically invoked by the audit contract to charge tenants by transferring the required assets from the deposit locked by the tenant to the operator's address. When the balance is insufficient, the operator will suspend the slice service until the tenant replenishes the deposit or terminate the service after waiting a long time.

Protocol: Audit. The *audit* protocol supervises slice quality and processes billing and is implemented by the audit contract. At each slot, the leader invokes the `ReportAudit` function to audit the usage reports uploaded during the last slot. The `ReportAudit` function first compares whether the usage reports uploaded by operators and tenants are consistent. If consistent, it will invoke the `Billing` function to charge the tenant according to the usage report. If not consistent, it should invoke the `Evidence` function to check which report is true.

- `ReportAudit`. This function checks two usage report sets in the form of MHT that are provided by operators and tenants separately. It compares their consistency and invokes `Billing` or `Evidence` contract as needed.
- `Billing`. Given a set of usage reports, the `Billing` function invokes the `Charge` function to transfer tenants' deposits to operators according to each usage report.
- `Evidence`. The `Evidence` function handles inconsistent usage reports. When members of the slice committee find the invocation of `Evidence`, they conduct off-chain investigations (e.g., checking the current service quality from end-to-end network test) to check which report is correct and upload the result to the contract. When receiving the same result from 2/3 members, the corresponding report will be regarded as correct (and become the input of the `Charge` function).

Protocol: Dispute. The dispute protocol is designed for tenants to claim that a slice does not satisfy the negotiated SLA. For some slices, the report uploading period may be set to a relatively long time, and the audit protocol cannot timely discover the quality decline of slices. Therefore, the dispute protocol acts as a supplement to the SLA guarantee. The dispute protocol is implemented in the dispute contract that consists of two functions: `Dispute` and `Evidence`. The

During each epoch, the current leader invokes the audit contract for auditing and charging at each slot. Furthermore, at the last slot, the leader invokes the committee contract to select the committee and leader for the next epoch.

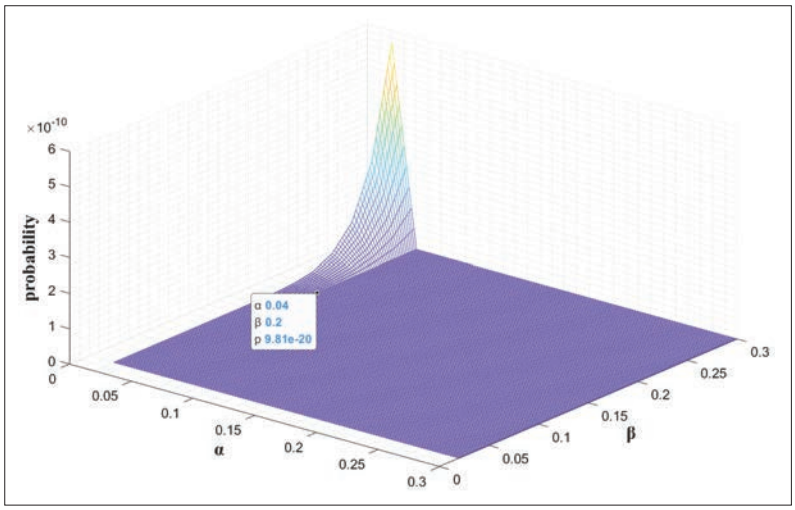

FIGURE 3. Probability that the system accepts a wrong test result with different α and β.

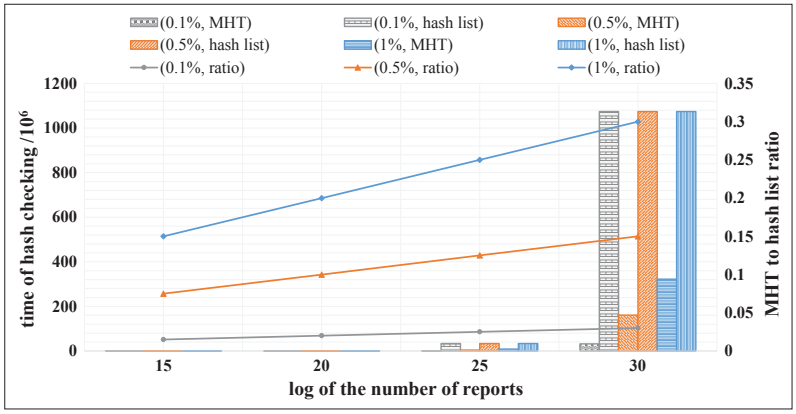

FIGURE 4. A comparison between MHT-based and hash list-based report check.

former is invoked by a tenant to provide the identity of the objective slice. Then it automatically invokes the latter to handle the dispute, and the process is the same as in the audit contract.

Security and Performance Analysis

Security Analysis

Assertion: A dishonest tenant may try to underpay for the slice but cannot succeed in our proposed framework.

Proof: On the one hand, a tenant could interfere with billing by lying about the usage report, for example, lowering the service level in the report. On the other hand, a common charge mode is use-then-pay, i.e., for each charge epoch, the tenant pays for the service at the end of the epoch; thus, a dishonest tenant may leave without paying the fee of the last epoch. In our proposed framework, the *audit* and *slice* protocols solve the two problems, respectively. By adopting `ReportAudit` to compare the usage reports uploaded by operators and tenants, slice managers can find a lying tenant or operator when inconsistency appears. Besides, tenants are required to lock enough deposit into the slice contract when requesting slices, and the `Charge` function will automatically charge tenants from the deposit. According to the reliability of blockchain smart contracts, tenants cannot succeed in defaulting on payments. □

Assertion: A dishonest operator may try to charge tenants more than they should or lower the service level of slices in use, but cannot succeed in our proposed framework.

Proof: Similar to tenants, a dishonest operator may try to charge tenants more than they should pay by providing forged usage reports, and this kind of misconduct can be detected and handled by the *audit* protocol in our framework. Besides, a dishonest operator may also lower the service level of slices to save cost or to spare network resources to create more slices, violating the SLA. This problem is solved by the *audit* and *dispute* protocol. A violation of SLA can be detected when the *audit* protocol finds an inconsistency of usage reports or a tenant executes the *dispute* protocol to proactively claim the decrease in service level. In conclusion, by combining *audit* and *dispute* protocol, SLA is strongly guaranteed. □

Assertion: A dishonest miner may try to provide false test results to disturb the reliable execution of the management protocols but cannot succeed in our proposed framework.

Proof: In our proposed framework, a group of miners is randomly selected to form the slice committee that executes the slice management protocols. Other miners are ordinary blockchain miners that obey the blockchain consensus. Since the underlying blockchain is considered secure, we can ignore ordinary miners and concentrate on the security of the slice committee. For simplicity, we assume that half of the miners are operators and others are tenants. And intuitively, dishonest operator miners tend to favor operators during the audit and tenant miners are on the contrary. Suppose there are a total of $N = 1000$ miners in the system and βN are dishonest, and the committee consists of αN miners. Only when the committee includes more than $2/3\alpha N$ dishonest tenant-miners or operator-miners will the system accept a wrong test result, and the probability is shown in Fig. 3. According to the result, the system might accept a false test result when $\alpha < 0.04$ and $\beta > 0.2$. And for other settings, the probability will stay at a tiny value close to 0, which is negligible in practice. □

Performance Analysis

To evaluate the performance of the proposed scheme, we implement slice management contracts through Ethereum Solidity. We mainly concentrate on the report check and network test. For report check, to explore the effect of adopting MHT, we also implement a report check function in the slice contract based on the hash list and compare their runtime. Furthermore, for the network test, we simulate 500 miners and 10,000 tenants randomly distributed in a specific area and estimate the test latency.

The result of the report check is shown in Fig. 4. We set the proportion of inconsistent reports as 0.1 percent, 0.5 percent, and 1 percent, and the total number of reports as 2^{15}, 2^{20}, 2^{25}, 2^{30}. Then we analyze the time of hash check for conducting a report check. As Fig. 4 shows, when the report size increases, the check cost of the hash list method grows sharply while the MHT-based method performs much better. When the

report size reaches 2^{30}, the hash list-based method should conduct 10^9 hash checks, while the MHT-based method needs only around 10^6. The less the inconsistency proportion, the more optimized the MHT method is compared to the hash list method. When there are 0.1 percent inconsistent reports, the cost of the MHT-based method is around 5 percent of the hash list method, proving the effectiveness of adopting MHT for the report check.

To evaluate the performance of the network test, we simulate 500 miners and 10,000 tenants randomly distributed in a specific area. The communication delay is proportional to the distance and is 100 ms between the two farthest nodes. The test concurrency of miners is set to 100, i.e., a miner could test 100 tenants at the same time. Then we set the committee size as 50 and test the average time cost of each miner with the proportion of inconsistency reports from 0.1 percent to 0.5 percent. The result is shown in Fig. 5a. We can find that the time cost of each miner increases with the inconsistency proportion increases and is less than 1 second, which is a relatively small cost compared to blockchain consensus.

We also fix the proportion as 0.03 and change the committee size to test the total execution delay of the `Evidence` function. The concurrency of tenants is set to 20. The delay consists of committee members' testing slice quality, uploading results, and other miners verifying signatures. The network test delay results from previous experiments, and the signature verification delay is represented by the throughput of executing the `Evidence` function, which is tested through Caliper. As Fig. 5b shows, the average delay of executing the `Evidence` function increases with the increase of committee size.

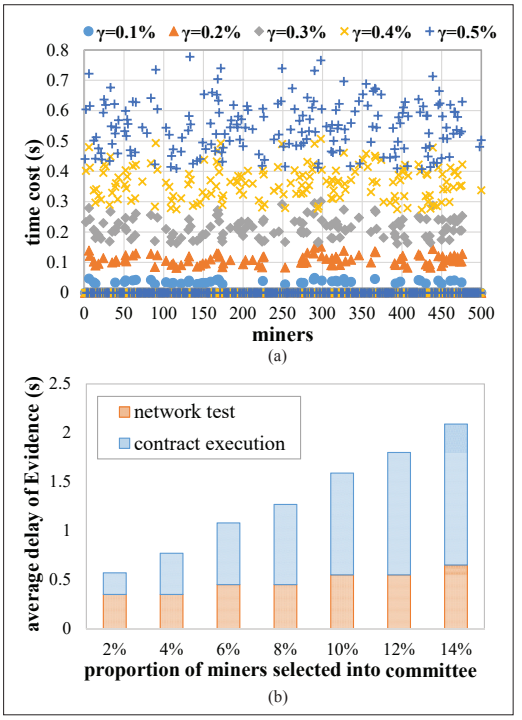

FIGURE 5. Time cost of the network test and `Evidence` function in *audit* protocol: a) time cost of network test; b) average delay of `Evidence`.

Conclusion

Aiming to enhance the security and fairness of network slicing, we introduced blockchain to propose a network slice management framework with correct billing and SLA guarantee. Our proposed framework consists of the *slice*, *audit*, and *dispute* protocols. Our main idea is to utilize blockchain's public and reliable storage to audit and compare the usage reports from operators and tenants. Besides, our scheme relies on network tests to handle inconsistency and disputes among operators and tenants. However, it is infeasible to directly implement the network test through smart contracts due to its relatively high cost. Therefore, we proposed the *slice committee* mechanism to handle the inconsistency and dispute for *audit* and *dispute* protocols. We analyzed the possible dishonest behaviors to prove security and conducted some experiments to show the practicability of the proposed scheme.

Acknowledgments

This work is supported in part by the Key Research and Development Program of Anhui Province under Grant No. 2022a05020050, the National Natural Science Foundation of China under Grant No. 61972371, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] K. Katsalis *et al.*, "Network Slices Toward 5G Communications: Slicing the LTE Network," *IEEE Commun. Mag.*, vol. 55, no. 8, 2017, pp. 146–54.
[2] C. Benzaïd *et al.*, "Trust in 5G and Beyond Networks," *IEEE Network*, vol. 35, no. 3, 2021, pp. 212–22.
[3] —, "AI-Based Autonomic and Scalable Security Management Architecture for Secure Network Slicing in b5g," *IEEE Network*, vol. 36, no. 6, 2022, pp. 165–74.
[4] I. Afolabi *et al.*, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Commun. Surveys & Tutorials*, vol. 20, no. 3, 2018, pp. 2429–53.
[5] K. Samdanis *et al.*, "From Network Sharing to Multi-Tenancy: The 5G Network Slice Broker," *IEEE Commun. Mag.*, vol. 54, no. 7, 2016, pp. 32–39.
[6] M. A. Togou *et al.*, "DBNS: A Distributed Blockchain-Enabled Network Slicing Framework for 5G Networks," *IEEE Commun. Mag.*, vol. 58, no. 11, 2020, pp. 90–96.
[7] T. Hewa *et al.*, "Blockchain-Based Network Slice Broker to Facilitate Factory-As-A-Service," *IEEE Trans. Industrial Informatics*, vol. 19, no. 1, 2022, pp. 519–30.
[8] I. H. Abdulqadder *et al.*, "SliceBlock: Context-Aware Authentication Handover and Secure Network Slicing Using DAG-Blockchain in Edgeassisted SDN/NFV-6G Environment," *IEEE Internet of Things J.*, vol. 9, no. 18, 2022, pp. 18,079–97.
[9] G. O. Boateng *et al.*, "Blockchain-Enabled Resource Trading and Deep Reinforcement Learning-Based Autonomous RAN slicing in 5G," *IEEE Trans. Network and Service Management*, vol. 19, no. 1, 2021, pp. 216–27.
[10] G. He *et al.*, "NetChain: A Blockchain-Enabled Privacy-Preserving Multidomain Network Slice Orchestration Architecture," *IEEE Trans. Network and Service Management*, vol. 19, no. 1, 2022, pp. 188–202.
[11] P. Rost *et al.*, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Commun. Mag.*, vol. 55, no. 5, 2017, pp. 72–79.
[12] A. Ksentini *et al.*, "Toward Enforcing Network Slicing on ran: Flexibility and Resources Abstraction," *IEEE Commun. Mag.*, vol. 55, no. 6, 2017, pp. 102–08.
[13] H. Zhou *et al.*, "A Blockchain Based Witness Model for Trustworthy Cloud Service Level Agreement Enforcement," *Proc. 2019 IEEE Conf. Comp. Commun. (INFOCOM)*, 2019, pp. 1567–75.
[14] X. Zhou *et al.*, "Network Slicing as A Service: Enabling Enterprises' Own Software-Defined Cellular Networks," *IEEE Commun. Mag.*, vol. 54, no. 7, 2016, pp. 146–53.
[15] L. Breidenbach *et al.*, "Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks;" https://research.chain.link/whitepaper-v2. pdf, White Paper, accessed on Sept., 2022.

Biographies

Xinyi Luo (lxy0213@mail.ustc.edu.cn) received her B.S. degree in Information Security from School of the Gifted Young, University of Science and Technology of China (USTC) in July, 2020. She is currently working toward the Ph.D degree in the School of Cyber Science and Technology, USTC. Her research interests include Network security and Cryptography.

Kaiping Xue [M'09, SM'15] (kpxue@ustc.edu.cn) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is a Professor in the School of Cyber Science and Technology, USTC. His research interests include future Internet architecture design, transmission optimization, and network security.

Jian Li [M'20] (lijian9@ustc.edu.cn) received his bachelor's degree from the Department of Electronics and Information Engineering, Anhui University, in 2015, and received his Ph.D degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2020. He is currently a research associate with the School of Cyber Science and Technology, USTC. His research interests include future Internet architecture design and quantum networking.

Rruidong Li [SM'07] (lrd@se.kanazawa-u.ac.jp) received his bachelor's degree in engineering from Zhejiang University, China, in 2001, and received his Ph.D degree from the University of Tsukuba in 2008. Currently, he is an associate professor in College of Science and Engineering, Kanazawa University, Japan. His research interests include big data networking, information-centric network, network security, and quantum Internet.

David S.L. Wei [SM'07] (wei@cis.fordham.edu) received his Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. He is currently a Professor of Computer and Information Science Department at Fordham University. His research interests include cloud computing, big data, and quantum networking.