Tracing Source and Security Audit of Entity Behavior in Core Business of Power Grid

Huafei Yang¹, Kun Liu¹,², Rui Xu¹,², Mengxian Chen³,*, Xiaojin Chi³, Yiting Cai³, Teke Jiang⁴

¹Nari Group Corporation / State Grid Electric Power Research Institute Nanjing, China
²Nari Information Communication Science and Technology Co. Ltd. Nanjing, China
³State Grid Zhejiang Electric Power Co., Ltd. Wenzhou Power Supply Company Wenzhou, China
⁴Dept. of Development, Open Security Research Lab Co. Ltd. Shenzhen, China

*Corresponding author e-mail: monicaxian@126.com

Abstract. Log-based security audit plays a key role in the safe operation of the system and evidence collection. Log auditing can effectively detect relevant threats and provide warnings in time, it can also trace the traces of entity behavior through logs. The log audit of the power grid is currently handled by the audit center. The audit center records the entire operation process and provides audit record query. But the research on the audit center system found that the audit center lacks security protection for the logs. Aiming at the above problems, a log anti-tampering system based on block chain technology is proposed, which uses block chain storage structure and underlying cryptography technology to ensure that the log cannot be tampered with. The test results show that after the log is protected by this method, it takes a lot of cost to tamper with the log without being detected, which effectively ensures the security of the log.

1. Introduction

As the core business of the power grid continues to grow and develop, the subsystems involved are diversified and complicated, and the number of people involved has also multiplied. The core business of the grid faces intrusions and attacks from outside the organization, as well as internal violations and unauthorized operations. The entity behavior in core business of power grid is audited by the audit center. The functions of the audit center mainly include seven parts: audit display, data analysis, abnormal monitoring, command channel, tool audit, data model and integration interface. The function audit display can query the specific function operations of the user by using the user account, the operation date, the center name, including the session ID, the user account, the user terminal IP, the center name, the operation time, the function operation description, and the execution result. Therefore, high-efficiency entity behavior tracking can be traced through the audit display [1]. However, due to the lack of security protection for the log itself, the criminals can falsify the log by forging or deleting some operational traces, which leads to low credibility of the entity behavior’s traceability. At present, the log anti-tampering has not paid enough attention, so the anti-tamper application of the log is still immature. The block chain technology has anti-tampering characteristics and has been verified in many transaction
payment fields. Therefore, this paper proposes a log anti-tampering scheme based on block chain technology. The scheme utilizes the irreversible modification of block chain technology to solve the problem of poor traceability of user entity behavior tracking in the core business of power grid, and utilizes the non-repudiation of block chain to do a good security audit of the core business system of the grid [2, 3].

2. The status quo of entity behavior’s traceability

2.1. The Significance of Entity Behavior Audit
After illegal operations by lawless elements, certain traces will be left in the log system. By tracing the source behavior, you can discover system-related security vulnerabilities and identify relevant offenders. Some employees caused accidents after illegal operation, the traceability of the physical behavior of the log system is traced to identify the responsible person, and statistics can be used to determine which operations are prone to violations and to continuously improve the system.

2.2. Log Data is Vulnerable to Attack
The log data is attacked mainly in the following aspects:
- By deleting the relevant log system data, the purpose of hiding behavior traces is achieved;
- Tampering with relevant operational records and modifying the violation operations to conform to the operational specifications;
- When the business needs to prove the workload through the operation record, it falsifies the record to deny its unfinished work.

It can be concluded that entity behavior’s traceability has great value and significance. The log not only suffers from external threats, but also faces malicious tampering from within. And malicious falsification from within is more threatening because internal personnel generally have high operational authority.

3. Defense method for log audit security issues
In response to the security problem of log auditing, there are mainly three methods to protect log security audit:

3.1. Backup Log Data
Regularly back up the logs to the backup server. When the audit log is needed, calculate the hash value of each log file (save one log file a day), and use these hash values to build a merkle tree. When the backup log and system log calculate the merkle roots are the same, it indicates that the log has not been tampered. If they are not the same, then the tampering log is found by traversing the merkle tree [4]. This method can find out where the log file has been tampered and can repair the log. However, its hardware cost is high, and once the backup server is attacked or has problems, the credibility of the log system cannot be verified. The protection of the backup server also introduces a new problem. To generate a merkle tree, you need to perform in times of hashing (n is the number of log files to be compared, in times of hashing only generates the value of the leaf node, in order to generate the merkle tree, you need to do n-1 hash operations for the node). For comparison, 2n times of hashing operations are required, when n is large, it takes a lot of time to check the log data each time. If you need to tamper with the log file, you only need to change the backup log data and the database log data to the same. You do not need to use complex calculations to achieve undetected discovery when tampering with the log. Getting access to the backup server and log database is a simple matter.

3.2. Backup Log Hash
By backing up the hash value of each log, when the log audit is required, the hash value of each log is compared with the hash value of the backup. If they are the same, the corresponding log has not been tampered with. If not, the log has been tampered with. This method can accurately locate which log has
been tampered with, but as the number of log files increases, the hash of each record increases the hardware storage overhead. And once the hash value of the backup is also tampered with, the tampering of the log system will not be found. The hash value backed up here becomes a new protection object, and tampering with a single log does not affect other log data. The log data needs to be hashed n times, and n is the number of logs to be compared.

3.3. Sign the Log File

By performing hash calculation on the log file, and then encrypting the hash value by the asymmetric encryption algorithm as the signature of the file, the log hash1 is calculated by the log hash, and then the hash information is decrypted to obtain the hash2, and the values of hash1 and hash2 are compared. If they are not the same, the log has been tampered with. Although this method does not require a backup system, once a cryptographic key is possessed, it is easy to forge a log file again, and tampering with a single log file does not affect other log file signatures. This is easy to do for internal managers.

The above three methods, in which the backup log data method hardware cost is high, and the protection of the backup server has become a new problem; the backup log hash method also introduces a new protection object, that is, the log hash value; The signature method is easily forged by a person with relevant permissions inside. And in these three methods, tampering with a single log file will not affect other log files. Although these three methods have strengthened the protection of the log system to a certain extent, these protections are not sufficient for the growing attack technology.

4. Log Protection Design Based on Block chain

Block chain is a decentralized protocol that combines existing cryptography and consensus algorithms to store data securely [5-7]. For example, the most widely used digital currency transaction data is now used to ensure that transaction information cannot be tampered with. The basic storage structure is a chronological chained data structure. The transaction confirmation on the block chain is completed by all nodes on the block chain. The consensus algorithm ensures that the nodes reach consensus without centralized management and the consistency of the main chain data. However, when the decentralized block chain technology is applied to the log tamper-proof, the decentralized redundancy backup and the centralized management concept of the log are contrary [8], which is not conducive to the storage, audit and analysis of the log. Therefore, it is a wise choice to take advantage of the block chain tamper-proof feature while ensuring the centralized nature of the log, and it is necessary to abandon the consensus algorithm of the current block chain [9] and the multi-node storage method. Focus on the application of block chain data structured storage concepts and the use of cryptography.

4.1. Business Processes

The traceability of user behavior involves two roles, one is the user and the other is the log reviewer. The system needs to be responsible for collecting and processing the logs generated by the user behavior, and obtaining multi-dimensional key value data, which are stored in chronological order. In the log library, the key value data is then presented to the log reviewer through the list. The system also needs to ensure that the log is stored securely and has tamper-proof features. The specific method is to store the log to the log library in a block chain manner, and the log blocks are linked to each other to form a block chain, and the modification of the log is triggered by the whole body. Tampering with the log requires refactoring the entire log block chain to be undetected by the auditing system. The encryption and decryption operations used by the system are performed in the encryption machine, and the encryption and decryption keys are also stored on the encryption machine. The business process is shown in Fig. 1:
4.2. System Framework
The system framework is shown in Fig. 2, where the log collection uses the interface of the audit center system. The system uses the daily log as a block, so the system will create a new log block every morning, and the new block will be completed by the block generation module. When the auditor needs to query the entity behavior, it is necessary to use the audit module to verify whether the log has been tampered with or not by entering the start date and end date of the log block to be verified. After the log verification is successful, the auditor can use the visual interface provided by the audit center to trace the entity behavior, and the multi-dimensional review of the entity behavior can be performed through the filtering function.

4.3. Block Data Structure
Each log block is composed of a block header and a block body. As shown in Fig. 3, the block body is composed of log information, and each piece of data information of the block header is as follows:
Version number: Each day's log is a block, the version number is accumulated from 0, and the No.0 block is a creation block. The block header has only the version number, and other elements are empty.

Previous block signature information: performs hash256 operation on the entire block data of the previous block, get the hash value and then encapsulated into 256 bytes by PKCS1.5. Finally, RSA encryption is performed by the encryption machine to generate 256-byte encrypted information, which is the signature information. This will ensure the security of the information and ensure the confidentiality of the information.

Random array: The random array has a total of 32 elements. By performing a hash operation on the corresponding block of the random array, the obtained value is used as the leaf node of the merkle tree, and the rightmost leaf node of the merkle tree is the value obtained by performing the hash operation on the previous block of the current block.

Merkle root: The merkle root is the top node of the merkle tree. If the value of the merkle tree leaf node changes, it will eventually be passed to the merkle root. The leaf node of the merkle tree here consists of the random hash value of 31 blocks and the hash value of a block on the current block. For the 32 blocks behind the creation block, the leaf nodes start from the first block, and the insufficient leaf nodes are replaced with the hash of 0. The purpose of adding the merkle root to each block header is to randomly verify the block in the block chain when creating the block, improve the defense capability of the log system, and then generate the merkle root based on the verified block hash value.

4.4. Modular Analysis

As can be seen from the overall architecture diagram of the system of Fig. 2, the entire system mainly includes the following modules:

4.4.1. Log Acquisition Module. To track the traceability of the user's behavior, the log of the user's behavior must be event-driven. When the corresponding event occurs, the log system can store the log according to the pre-determined log format. In the massive behavior log, it facilitates the multi-dimensional review of the entity corresponding to the event. For example, when you want to query the behavior of a user, you only need to filter the user name to get all the operations of the user, and can display them in chronological order. The collection log is presented to the auditor in the form of a table. The auditor can filter all operations of a user and trace the source behavior through a visual interface. It is also possible to prove that the operation of the relevant personnel conforms to the specification through this interface, all the operations of the user are connected in chronological order, and all operations can be restored.

4.4.2. Block Generation Module. The system uses one day's log as a block, and a new block is generated in the 0 am, but before the newly generated block, it is necessary to make a corresponding contribution to the main chain to generate a new block [10]. The biggest difference with the block chain is that this workload needs to be fixed. If the block chain workload proof mechanism is not fixed, the generation of the latter block will take a lot of time over time. The application of the system is obviously not suitable. The process of generating new blocks is shown in Fig. 4:

Figure 3. Log block data structure diagram
Start

Get the version number

Generate a random array

Verify the block corresponding to the random array

- Passed verification or not

  - Yes
    - Calculate the signature information of the last block
    - Generate a merkle root with the random array and the previous block hash values as the leaf node
    - Package version number, random array, signature information, merkle root into block header
    - Start recording log
    - New block generated successfully

  - No
    - Stop creating new blocks and notify log auditors

Figure 4. New block generation process
4.4.3. Audit Module. The main task of the audit module is to verify whether the log has been tampered with. The system audit process is shown in Fig. 5. Before the auditor traces the source behavior of the user entity, the auditor needs to confirm whether the corresponding log has been tampered with. At this time, you need to use the audit module to verify whether the log of the corresponding date has been tampered with. The auditor simply enters the start date and end date and the system verifies the log for the corresponding date. The sub-functions of the audit module are described in detail as follows:

**Chain verification:**
- a) Calculate the hash value of the S block to get the hash1;
- b) Decrypt the signature information of the S+1 block to obtain hash2;
- c) Then compare whether the two hashes are the same;
- d) 4) If the same, recursive verification until the E+1 block is successful.

**Merkle Root Verification:**
- e) List the merkle roots that have a hash value of a leaf node in a block from the S block to the E block;
- f) Randomly pick a merkle root for merkle1;
- g) Calculate the merkle root corresponding to the random block of the merkle root to obtain merkle2;
- h) Compare the two merkles. If they are consistent, the verification is successful. If they are inconsistent, the log is tampered with, but the tampered log is not necessarily within the period set by the auditor, but the random block corresponding to the merkle has may be tampered with.

When an auditor examines a block within a certain period of time, in addition to chain verification of the block within the term, it is also verified by the corresponding merkle root to verify that at least one of the blocks is safe and has not been tampered with. This prevents the entire fragment from being tampered with. As shown in Fig. 6, the S block is the starting block that needs to be audited, and the E block is the end block that needs to be audited. If the attacker tampers the entire block from the S block to the E block, it cannot be audited. It is found that if the merkle root of one or several blocks in the leaf block from the start block to the end block is verified to be tampered with by the merkle root, it is easy to find out the fact that the log has been tampered with. In simple terms, the merkle root verification is to ensure that the block within the verification period has not been tampered with. As shown in Fig. 4, the hash value including any block of the S block to the E block is enumerated as the merkle root of the leaf, and then a merkle root verification is randomly selected. If the calculated merkle is consistent, the R block is not tampered with.
Chain verification from block S to block E+1

Whether the log has been tampered with

Merkle root verification passed?

The log has not been tampered with

Audit passed

Handling exceptions through internal processes

Figure 5. Audit process

Merkler root

Leftmost node

Rightmost node

Block S

Block R

Block E

Figure 6. Merkle root function
5. Program analysis

5.1. Security Analysis
All encryption and decryption work of this system is completed by the encryption machine. The encryption and decryption algorithm adopts pluggable design, which supports both international algorithms and Chinese national secret algorithms. The encryption and decryption authority of the encryption machine is bound to the audit system server, so only the audit system server can communicate with the encryption machine. The encryption machine administrator authority adopts the split management mechanism, and the administrator uses the IC card with the commercial password product model as the authority management physical medium and sets the password protection. There are a total of 6 administrators, each with only a portion of the backup key. Key preservation and encryption and decryption operations are reach financially secure level.

If someone gets the encryption and decryption call permission of the encryption machine, and the permission to read and write the log library. In order to modify a log block, it needs to reconstruct the entire log block chain. The steps to reconstruct the log block chain are as follows:

1) Modify the "previous block signature information" of the next block, and perform a hash operation and add one operation;
2) Modify the "merkle root" of the next block, and perform 2*32-1 hash operations and one comparison, which is equivalent to 32 decryption operations and 32-1 hash operations;
3) Continue to modify recursively until all blocks of signature information to regenerate.

Now simulate the daily log file size is 1G, if the modified block is m blocks from the latest block, the estimated time and test time are as follows.

5.1.1. Estimated Time. Reconstruction of the entire block requires: m * (1 + 2 * 32 - 1) = 64m times of hash operations, m times of encryption operations, and 2m times to modify the block header signature information.

5.1.2. Tested Time. The test time here is done by the test code and is calculated by the server, regardless of the manual modification of the data factor. The test result is shown in Fig. 7. The test result is in line with the estimated time. It can be seen from the figure that the time cost of the tampering increases exponentially over time. It explains that log block security becomes more and more secure as storage time increases.

![Figure 7. Time cost of linked list reconstruction](image)
5.2. Performance Test Analysis

The prototype system is built to test the implementation performance of the solution. Because the system performance bottleneck is at the server, the mainly tested is the server's responsiveness. The prototype system is configured with Xeon-E5 processor, 32G memory, CentOS7 operating system, log anti-tampering system using java language development, back-end core framework using Spring Boot, database connection pool using Alibaba Druid, cache framework using Redis, log management’s framework using SLF4J, and the data table framework using bootstrap table to provide a visualized entity behavior tracking source interface and a log security audit interface. The database uses mysql to store chunked log information. The front end is docked in B/S mode. Mainly provides log audit API. The log audit API requires two parameters to be entered, a start date and an end date. The performance test mainly tests the verification performance of the log block. Now assume that the daily log file size is 1G, and it is necessary to verify whether a certain m block has been tampered with. The estimated time and test time are as follows.

5.2.1. Estimated Time. Each audit requires two verifications, chain verification and merkle root verification.

Chain verification time:
M times of hash calculations, m times of decryption calculations and m times of comparisons;

Merkle verification time:
2*32-1 times of hash calculations, 1 comparison.

The total time is: m+63 times of hash calculations, m times of decryption calculations, and m+1 comparisons.

5.2.2. Tested Time. The time required for the log block verification is tested by the log audit API. The test result is shown in Fig. 8, and the test result is in accordance with the estimated time. As the audit period increases, the audit time will increase slowly. The time is within acceptable limits for monthly audits and quarterly audits.

![Figure 8. Performance of log verification](image)

6. Conclusion

In view of the problem that the log is easily falsified, this paper proposes and implements the log tamper protection based on the block chain technology. The method introduces the block chain storage format of the block chain technology and the underlying cryptography technology, and forms a block of daily logs, links the blocks with signature information to form a chain relationship, and uses the merkle root to make the chain generate complex random relationships. Changing the data of one of these blocks will affect the entire chain. When an attacker needs to tamper with a block, in addition to obtaining
encryption and decryption permissions and database operation permissions, it must spend a lot of time
to reconstruct the entire chain. According to the simulation test, the solution is applied to the grid core
business system, which has a large-scale system with hundreds of thousands of users. The log audit
performance and the tampering time cost are well balanced to meet the security and efficiency of the
grid core business system.

Acknowledgments
This work is partially supported by the State Grid Corporation Science and Technology Project Funded
"Key Technology Research on Trustworthy Identity Authentication of Grid Core Business"
(52110418001L).

References
[1] DUAN Juan, XIN Yang, MA Yu-wei. Research and Design of Security Audit Log System Based
on Web Application [J]. Netinfo Security, 2014, 10: 70 - 76.
[2] LIU Ao-Di, DU Xue-Hui, WANG Na, et al. Research Progress of Blockchain Technology and
Its Application in Information Security [J]. Journal of Software, 2018, 29 (7): 2092 - 2115.
[3] HAN Xuan, YUAN Yong, WANG Fei-Yue. Security Problems on Blockchain: The State of the
Art and Future Trends [J]. Acta Automatica Sinica, 2019, 1: 206 - 225.
[4] SU Di, LIU Zhu song, Faculty of Computer. New type of Merkle Hash Tree for integrity audit
scheme in cloud storage [J]. Computer Engineering and Applications, 2018, 54 (1): 70 - 76.
[5] SHAO Qi-Feng, JIN Che-Qing, ZHANG Zhao, et al. Blockchain: Architecture and Research
Progress[J]. Chinese Journal of Computers, 2018, 41 (5): 969 - 988.
[6] HUANG Jun-fei, LIU Jie. Survey on Blockchain Research [J]. Journal of Beijing University of
Posts and Telecommunications, 2018, 2: 1 - 8.
[7] HE Pu, YU Ge, ZHANG Yan-feng, et al. Survey on Blockchain Technology and Its Application
Prospect [J]. Computer Science, 2017, 44 (4): 1 - 7, 15.
[8] BI Ling-na, HAN Li-heng. Research on the Trap and Paradox of Decentralization Blockchain
Currency [J]. Journal of Hubei University of Economics,2019, 1: 47 - 53, 127.
[9] Song Taoyi, Zhao Yunlei .COMPARISON OF BLOCKCHAIN CONSENSUS ALGORITHM
[J]. Computer Applications and Software, 2018, 35 (8): 1 - 8.
[10] WANG Shuai, QIN Bo, CHEN Jinchuan, et al. Blockchain Estimation Model Based on PoW
Mechanism [J]. Journal of Cyber Security, 2018, 3 (3): 30 - 42.