International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 613-626

https://doi.org/10.70517/ijhsa46455

Research on sensitive data discovery and optimization algorithm based on trusted execution environment

Junfang Sun[1,*]
[1] State Grid Qinghai Information & Telecommunication Company, Xining, Qinghai, 810000, China
Corresponding authors: (e-mail: 13001041161@163.com).

Abstract Under the cloud computing environment, traditional security mechanisms are difficult to effectively protect the confidentiality and integrity of sensitive data. Aiming at the security protection of sensitive data in cloud storage environment, this paper proposes an optimization algorithm for sensitive data discovery and integrity checking based on Intel SGX trusted execution environment. Methodologically, the SGX-based trusted execution environment framework is constructed, the integrity verification scheme combining the multi-branch path tree (MBT) data structure and bilinear pair algorithm is designed, and the third-party verification organization and blockchain network are introduced to realize data integrity verification. The data verification in challenge-answer mode is realized through the smart contract mechanism, and the file version verification is optimized by combining the version sequence number parameter. The results show that when the number of files reaches 1900, the total execution time of this paper's scheme is reduced by 51.29% and 32.76% compared with the B-PDP and SA-PDP schemes, respectively. Tests based on the MNIST dataset show that the time overheads of the storage and validation phases are 0.728ms and 0.291ms, respectively. The overall performance of the Trusted Execution Environment network reaches 95.48% of the original Fabric, with only a 6.42% increase in latency and a 5.03% decrease in throughput. The conclusion proves that the scheme can significantly improve the efficiency of sensitive data verification under the premise of guaranteeing security, and provides an effective guarantee for data security in cloud storage environment.

Index Terms Trusted Execution Environment, Intel SGX, Sensitive Data, Integrity Verification, Multi-branch Path Tree, Blockchain

I. Introduction

With the development of mobile Internet and mobile terminal technology, smart phones and Internet of Things (IoT) have begun to integrate into every corner of people's daily lives. While enjoying the convenience brought by these technologies, they also bring a lot of security risks. While they provide rich functions, they also provide opportunities for information leakage and malware attacks, and various security problems are becoming more and more prominent [1], [2]. Most of the research reports in recent years show that the overall trend of sensitive information leakage problem is increasing year by year [3], [4]. For example, sensitive data leakage jeopardizes users' privacy and personal information security, and also seriously affects operators' core secrets, competitiveness in the same industry, and market reputation [5], [6]. At present, the security problem of sensitive data has aroused widespread concern from all walks of life.

There are more ways of sensitive data flow through the whole data life cycle, covering multiple links such as data generation, analysis, statistics, transfer, and expiration, etc., and sensitive data leakage is most likely to occur in the process of data flow to low-control environments, such as the production environment to the test environment [7]-[9]. Therefore, identifying sensitive data throughout the data lifecycle in order to obfuscate sensitive data has become a top priority [10]. Early identification of sensitive data by manual means, by the database administrator based on personal experience to find and determine the sensitive data, the method is time-consuming and laborious, and there is a high probability of missing sensitive data, after the introduction of intelligent technology and application models, the probability of missing sensitive data is reduced, but due to the semantic fuzzy characteristics of unstructured data, the differences in the classification of the sensitive data class in different environments, and the sensitive data dynamic changes, the sensitive data identification technology still needs to be improved [11]-[15]. The construction of trusted execution environment is an effective measure to protect sensitive data. Migrating sensitive data and applications to the trusted execution environment, establishing physical isolation from the ordinary execution environment, and preventing them from being directly read or modified are widely used in the fields of secure payment, identity authentication, digital copyright management, secure communication, data sharing, encryption key management, etc. [16]-[19].

The widespread application of cloud computing technology has driven a fundamental change in the data storage model, with organizations and individual users increasingly choosing to store their data in the cloud for cost-effectiveness and convenience. However, the open and shared nature of cloud storage environments also brings unprecedented security challenges, especially in the protection of sensitive data. When a user entrusts data to a cloud service provider, the physical control of the data is transferred and the user loses direct control over data access, and the asymmetry of this trust relationship becomes the core issue of cloud storage security. While traditional encryption techniques can protect data confidentiality to a certain extent, plaintext data is still exposed to potential security threats during data processing and computation. Malicious cloud service providers or insider attackers may access user data through privileged software, and existing software-level security mechanisms are often unable to defend against attacks from the operating system or hypervisor level. In addition, data integrity verification in cloud environments faces many challenges, including inefficient verification, high communication overhead, and difficulty in supporting dynamic data updates. Most of the existing data integrity verification schemes are based on cryptographic techniques, and although they have security guarantees in theory, they often face practical problems such as high computational complexity and high storage overhead in actual deployment, especially in large-scale data environments, where the performance bottlenecks of the traditional verification schemes are becoming more and more obvious.

Based on the above analysis, this paper proposes to use Intel SGX trusted execution environment technology to build an optimization scheme for sensitive data discovery and integrity verification. The research is divided into four levels: firstly, establish a trusted execution environment framework based on SGX to provide an isolated protection environment for sensitive data processing through a hardware-level security mechanism; secondly, design an integrity verification mechanism that combines a multi-branching path tree data structure and a bilinear pair algorithm to improve the efficiency and accuracy of data verification; and thirdly, introduce the blockchain technology and a smart contract mechanism to build a decentralized Third, blockchain technology and smart contract mechanism are introduced to construct a decentralized data integrity verification system, and efficient data verification is achieved through the challenge-answer mode; finally, the version control mechanism is combined to optimize the integrity verification in dynamic data update scenarios to ensure the practicability and scalability of the scheme.

II. Trusted execution environment construction based on SGX

In recent years cloud storage has received widespread attention from researchers and IT vendors, and many applications use cloud storage to store data. However, both users and vendors have expressed concerns about the security and privacy of cloud storage, the core of cloud storage security is the security and privacy of distributed file system. In cloud storage, since the physical medium for storing data does not belong to the cloud storage user, when the user hands over the data to the cloud storage provider, it is not the corresponding user who has priority access to the data, but the cloud storage provider. At the same time, in the cloud storage platform the user's data is in a shared environment, how to ensure the user's data privacy is a great concern.

II. A. Smart Contracts and the Hyperledger Fabric

II. A. 1) Definition of Smart Contract Behavior

Blockchain is a decentralized digital transaction ledger, which is jointly maintained by the nodes in the blockchain network under the constraints of a consensus agreement. The consensus protocol ensures that honest nodes in the blockchain network can reach consensus even when malicious nodes interfere, and the consensus process is the process of nodes verifying and updating the ledger, and the result of the consensus is that the system provides a unified ledger to the public. Transactions in a blockchain are stored in an ever-growing ordered list of “blocks”, each of which also includes state metadata, creation timestamps, the Merkle hash of the transaction, the hash of the previous block in the chain, as well as smart contract code and data [20].

In this paper, we propose a trusted verification mechanism for contract behavior to guarantee that the contract behavior meets expectations, and we study how to construct a real-time metric and protection method for contracts with reference to the definition of behavior in software behavioral science. The basic definition of smart contract behavior contains the following elements:

(1) The set of blockchain system states $S=(s_0,s_1,\dots,s_n)$, with the original state of the blockchain being s_0.

(2) The set of individual behavioral units A, where a behavioral unit a is an atomic operation that constitutes a certain behavior.

(3) A single contract's behavior set B, where the behavior α is formed by concatenating all behavior units a.

(4) The expected behavior rule set $P(X)$, where the expected behavior rule set specifies the rules to be followed when the contract behavior meets expectations, where X is the security condition to be met by the behavior α.

Contract behavior is defined as the subject's use of a function to operate on an object, i.e.,:

$$\begin{aligned} A = \{ & action = (s)applies(f)to(obj)in(env) \mid s \in subjects \\ & f : functions, obj : objects, env : environment \} \end{aligned} \tag{1}$$

where $subjects$ is the subject set, $functions$ is the function set, $objects$ is the object set, and $environment$ is the environment state. Where the subject mainly refers to the contract, the function set mainly refers to the built-in function operations (initialization, invocation, deletion, etc.) of the contract, the object set mainly refers to the blockchain ledger or external libraries, etc., and the environment mainly includes the sandbox, etc., and the contract behavior is defined as the contract reads or writes to the blockchain ledger through the built-in functions.

Contract behavior sequence is defined as the sequence of system calls triggered by the contract during the occurrence of a certain behavior, i.e:

$$\alpha = a_1, a_2, \ldots, a_n \quad a \in system\ call \tag{2}$$

where α represents a behavior of a smart contract, and $a_1, a_2, \ldots, a_n$ denote the sequence consisting of system calls triggered during the execution of this behavior.

The functionality of a contract is realized through ordered code segments, and if the code is executed in an incorrect order, the functionality will be faulty. Thus a particular function of a contract has a fixed sequence of behaviors. System calls have a better property in detecting whether the behavior is as expected or not, and it is only necessary to determine whether a contract is trustworthy or not by the difference between the system call sequence and the expectation.

Behavioral trustworthiness is defined as behavior that meets expectations, i.e:

$$\forall_i \in (1 \sim n), \forall_\alpha \in \alpha.\ Actual\left(s_0, \alpha, a_i\right) = Expected\left(s_0, \alpha, \left(P(a_i), a_i\right)\right) \tag{3}$$

where $Actual\left(s_0, \alpha, a_i\right)$ denotes the actual execution result of the behavior α, and $Expected\left(s_0, \alpha, \left(P(a_i), a_i\right)\right)$ denotes the expected outcome of the execution of the behavior α under the condition that the expected behavioral rules are satisfied. If the equation holds, it means that for any system call in the behavior α the expected behavior rule is satisfied, i.e., the contract behavior meets the expectation and thus is trustworthy. Whether a contract's behavior is plausible or not is determined by the satisfaction of the system calls as its behavioral units with respect to the expected behavioral rule $P(X)$. Trustworthy contract behavior not only discusses the condition satisfaction of each behavior unit, but also considers its execution order, as well as the contract's own integrity, execution environment, and so on, to satisfy the condition.

II. A. 2) Hyperledger Fabric

Hyperledger Fabric is an open-source permissioned distributed ledger technology platform customized for business environments, and the open source community of Fabric is committed to providing a solution that includes strict identity management and permission control mechanisms for enterprise-level applications, and effectively solves the problems of inefficient transactions and poor identity management of public blockchains through modular and plug-in architecture innovation [21].

Hyperledger Fabric's system architecture system empowers developers with a rich set of interfaces and tools that enable upper layer applications to fully utilize blockchain technology. From an application layer perspective, Fabric provides developers with the following key components and interfaces as follows:

(1) Identity management: Fabric is a permission-based blockchain, which means that participants need to be authenticated before they can join the network and perform operations, and the identity management service provides the functions of creating, managing, and verifying participants' identities.

(2) Ledger Management: Fabric uses a distributed ledger to record all transactions and status changes. The ledger management service is responsible for maintaining the integrity, availability and consistency of the ledger.

(3) Transaction Management. Transactions are the basic operations on the blockchain that are used to change the state of the ledger. The transaction management service is responsible for receiving, verifying, and executing transactions to ensure their legality and correctness.

(4) Smart Contract Management. Smart contracts are automatically executed business logic used to define and manage the behavior of assets on the chain, and the smart contract management service provides the deployment, invocation, and upgrading functions of the chain code.

II. B.SGX-based trusted execution environment

II. B. 1) Intel SGX Key Technology

Intel SGX is a new extension to the Intel architecture that adds a new set of instruction sets and memory access mechanisms to the original architecture. These extensions allow an application to implement a container, called an Enclave, to carve out a protected area in the application's address space, providing confidentiality and integrity protection for the code and data within the container from malware with special privileges [22]. The overall architecture of SGX is shown in Figure 1. The implementation of SGX requires hardware and software collaboration such as the processor, memory management component, BIOS, driver, runtime environment, and other hardware and software to work together. In addition to providing memory isolation and protection security attributes, the SGX architecture also supports remote authentication and sealing features, which can be used in the design of secure software applications and interaction protocols.

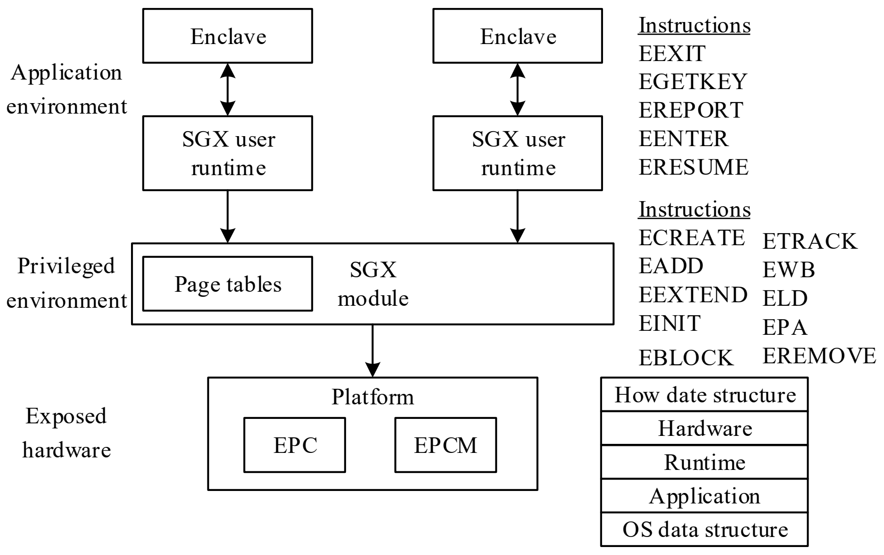


Figure 1: Architecture description of SGX

With SGX technology, developers can ensure the integrity and confidentiality of data code during application execution based on application code in trusted processors and secure execution areas. SGX technology mainly contains two core mechanisms, namely, isolated execution and remote authentication, as follows:

(1) Isolated execution mechanism. sGX technology allows (part of) the application code to be executed in a secure execution environment called Enclave. sGX technology can protect the code and data in Enclave from being tampered with and listened to by other application programs or high-level system software during the operation process. In order to realize the protection of application code and data from high-level system software attacks during operation, the application code, data, and Enclave-related key data structures in the Enclave are encrypted and stored in memory during operation. SGX technology does not affect the management and allocation of platform resources by the traditional operating system, and therefore realizes that the Enclave's access to the pre-preserved memory space (EPC) frames is not affected. Therefore, the page table for realizing frame mapping in the Enclave to the pre-reserved memory space (EPC) is still managed by the operating system.

(2) Remote Authentication Mechanism

In order to provide users with the function of verifying the integrity of the remote program, SGX technology provides a remote authentication mechanism, which is mainly accomplished by two commands, EGETKEY and REPORT, and the REPORT structural information mainly contains the identity information and some user data of Enclave A. The MAC value is generated by a report key, which is only valid for the target Enclave B. is only visible to the target Enclave B and the EREPORT command of the same platform. When Enclave B receives the REPORT message, it calls the EGETKEY instruction to obtain the key used to compute the MAC value of the REPORT structure, and then compares the recomputed MAC value with the MAC value of the received REPORT to confirm that Enclave A is running on the same platform. Once the trusted hardware part is confirmed, Enclave B then authenticates Enclave A's identity with the information in the REPORT. Finally, Enclave A verifies the identity of Enclave B in the same way to complete the mutual authentication within the platform.

II. B. 2) Trusted Execution Environment Framework

The overall architecture of the SGX-based trusted execution environment is shown in Figure 2. In the hardware layer, the CPU maintains and provides EPC, a physically isolated memory area, through the extended instruction set of Intel SGX technology and the EPCM structure. Based on the change of memory access semantics of Enclave and the protection of application address mapping relationship, these two functions together complete the protection of confidentiality and integrity of Enclave internal code and data. At the kernel layer, the SGX driver provides a large number of interface services for the application program, such as the creation and destruction of Enclave, trusted cryptography library, data sealing, remote authentication and other functions. In the application layer, user code and data are encrypted and decrypted in the secure zone by means of a trusted cryptography library, which ensures that the user code and data are in plaintext only in the secure zone and remain encrypted when stored in the cloud.

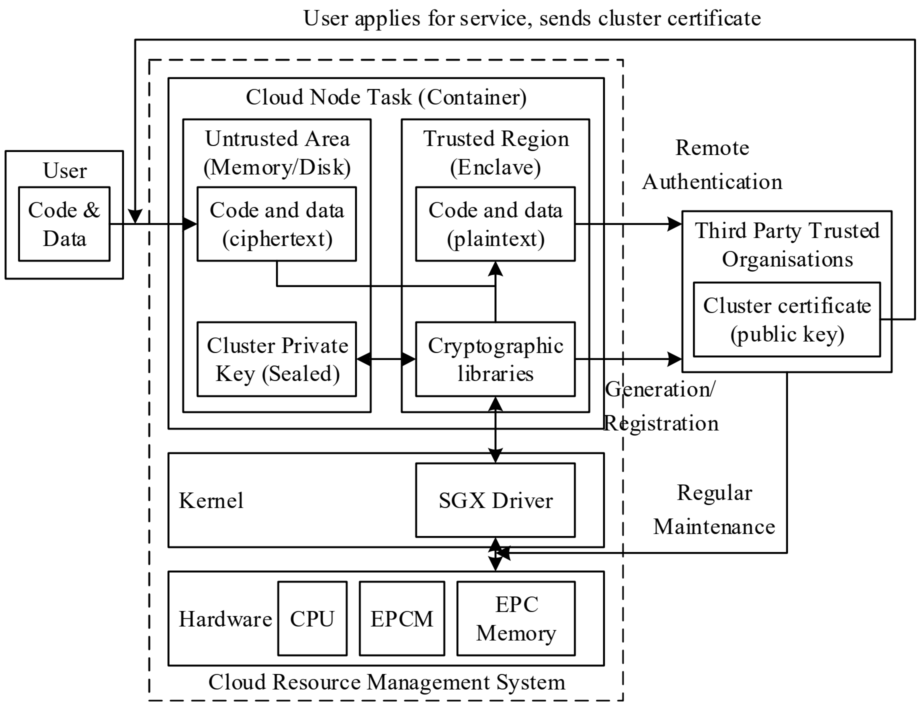

User applies for service, sends cluster certificate
Cloud Node Task (Container)
Untrusted Area (Memory/Disk)
Trusted Region (Enclave)
User
Code & Data
Code and data (ciphertext)
Code and data (plaintext)
Remote Authentication
Third Party Trusted Organisations
Cluster certificate (public key)
Cluster Private Key (Sealed)
Cryptographic libraries
Generation/ Registration
Kernel
SGX Driver
Regular Maintenance
Hardware
CPU
EPCM
EPC Memory
Cloud Resource Management System

Figure 2: The architecture of the trusted execution environment based on SGX

Its specific functions are as follows:

(1) The user's data is executed in the secure container Enclave provided by SGX technology, which can effectively prevent attackers from inside the cloud from using privileged software, ensuring confidentiality and integrity.

(2) The cryptographic library and sealing mechanism provided by the SGX driver generates cluster certificates, forming a set of secure and efficient data transmission encryption and decryption scheme through a third-party trusted organization to ensure that user code and data are always encrypted by the user-generated key in cloud transmission, and decrypted and executed only in the security zone.

(3) The security of the software and hardware environment of the cloud platform can be ensured through the authentication and maintenance of a third-party trusted organization. Meanwhile, remote authentication through the third-party trusted organization ensures the authenticity and effectiveness of the trusted environment of the cloud platform and solves the problems of false certification, result forgery and identity impersonation in the traditional trusted cloud.

(4) Through Enclave's isolation mechanism, secure data transmission scheme, and unified security standards and certification by the third-party trusted organization, it achieves the purpose of using the cloud platform's computing resources and not leaking information to the cloud platform, so that users can trust it.

(5) Only sensitive user code and data are put in the safe zone for execution to minimize the performance overhead caused by the use of SGX technology.

III. Sensitive data integrity verification program design

With the rapid development of the Internet era and the popularization of the mobile Internet, all kinds of computing platforms and cloud platform security issues have grown, and all kinds of malicious attacks threaten information security, but also lead to many enterprises and individuals suffer from the danger of privacy and property infringement. It is difficult to solve these problems simply by using software, and the trusted computing environment with hardware security chips as the root of trust provides a new way of thinking.

III. A. System Security Model and MBT Algorithm

III. A. 1) System security model

The system model designed in this paper contains three entities, i.e., user, cloud server and trusted auditor. The user is the data owner with the need to store personal data on the cloud server, which is provided and managed by the CSP and is capable of providing cloud storage services. The auditor can be a trusted third party organization that can check the correctness of the user's data stored on the cloud server. The system uses homomorphic verification technology that can compress the data blocks to a fixed length, so it is ideal for network environments with limited communication bandwidth.

In the system model, the user uploads the data to the cloud server and deletes the local copy of the data, which reduces the storage burden of the user. The user can also delegate the task of data integrity checking to an auditor, which is able to perform the task of data integrity checking instead of the user when the auditor passes the user's authorization authentication. Then, the said auditor sends an audit request to the CSP asking the CSP to perform an integrity check on the user's data in order to compute an evidence to return it to the auditor. Finally, the auditor verifies the correctness of this evidence and informs the user of the verification result. If the CSP passes the auditor's data integrity check then it means that the user's data is correctly stored on the cloud server. Otherwise, it indicates that the user's personal data has been corrupted.

In cloud storage service system, it is not only necessary to be sure about the integrity of the personal data in the cloud, but also to fulfill the user's request to update the data. In addition, since the cloud server is untrustworthy or semi-trustworthy and there is insecurity in dynamic operations, the possible forms of attacks on the system are as follows:

(1) Replay Attack and Forgery Attack. When the Hash function is of the form H(i), the value of its Hash function remains H(i) after updating the ith data block. If the CSP dishonestly updates the user's “data-block-tag”, then the CSP can use an outdated version of the “data-block-tag” to calculate the evidence. When the auditor wants to check the integrity of the user's cloud data, the CSP can use the above evidence to deceive the auditor.

(2) Deletion Attack: A CSP may delete a portion of a user's data block and then pre-calculate an aggregated “block-label”. When the auditor wants to perform a data integrity check, the CSP will use the pre-computed aggregated “block-label” to deceive the auditor.

III. A. 2) MBT and bilinear pairs

Each node of a Multi-branch Path Tree (MBT) has multiple child nodes, i.e., the out-degree is greater than or equal to 2. This data structure ensures the correctness of the data block at the storage location in the cloud and the verifiability of the dynamic updates, and for the same number of leaf nodes, the depth of the constructed MBT is much less than that of the MHT. In MBT, the authentication path refers to the set of nodes in the path from the target node to the root node, and the The auxiliary information, on the other hand, refers to the set of sibling nodes of all the nodes in that authentication path.

The depth of the MBT tree is set to 2 and the out-degree is n. Then the number of leaf nodes is n^2, which corresponds to the hash values of n^2 data blocks, respectively. The hash value of the intermediate node W_1, $h_{w_1} = H\left(H(m_1)\|H(m_2)\|\cdots\|H(m_n)\right)$, with the rank value n, and the root node R corresponds to a hash value of $h_R = H\left(h_{w_1}\|h_{w_2}\|\cdots\|h_{w_n}\right)$, with rank value n^2. The authentication path constructed by the cloud storage server for the data block m_2 is $\{H(m_2), W_1\}$, and the auxiliary information to authenticate this data block is:

$$\Omega_2 = \{H(m_1), H(m_3), \cdots, H(m_n), W_2, W_3, \cdots, W_n\} \tag{4}$$

Based on the leaf node corresponding to the data block m_2 contained in the challenge information and the corresponding auxiliary information, the hash value of the root node R' in the MBT can be computed as $h_{R'} = \{H(m_2), \Omega_2\}$. Then, it is compared with the true root node R hash value h_R to verify that the data block is correct in terms of location, and thus whether the cloud storage server has correctly performed the dynamic update operation.

Assuming that G_1 and G_2 are cyclic groups of order both prime q, and P is any generating element of the group G_1, the bilinear mapping $e: G_1 \times G_1 \to G_2$ from G_1 to G_2 satisfies the following properties:

(1) Bilinearity: the equality $e(P^a, Q^b) = e(P, Q)^{ab}$ holds for any group elements $P, Q \in G_1$ and variables $a, b \in \square_N$.

(2) Non-degeneracy: there exists $P \in G_1$ such that $e(P, P) \neq 1$ holds.

(3) Computability: for any $P, Q \in G_1$, there exists a polynomial time algorithm to compute $e(P, P)$.

III. B. Sensitive data integrity verification program

III. B. 1) Design of integrity checking program

In this paper, we propose a cloud storage data integrity verification scheme based on Intel SGX technology for the data integrity problem as shown in Fig. 3. A third-party validation authority (TPAI) is introduced to verify the data integrity of the requests from users and cloud storage providers (CSPs). The scheme includes users, CSPs and Blockchain Network (BCN) in addition to TPAI. The user is the user who uses the cloud storage service, which can be an individual user or a small or medium-sized enterprise. The CSP provides the cloud storage service, which provides the user with highly reliable, secure, low-cost, and easy-to-scalable data storage service. The BCN provides the storage and validation service, which stores the user's data integrity proof and auxiliary validation information into the block and provides the data with legally valid integrity proof based on the tamper-resistant feature of the blockchain. The BCN provides the storage and validation service. The smart contract in BCN forwards the user-initiated integrity challenge to CSP and verifies the CSP's response. TPAI, as a trustworthy and legally valid verification organization, can verify the data integrity of the arbitration initiated by the user and CSP.

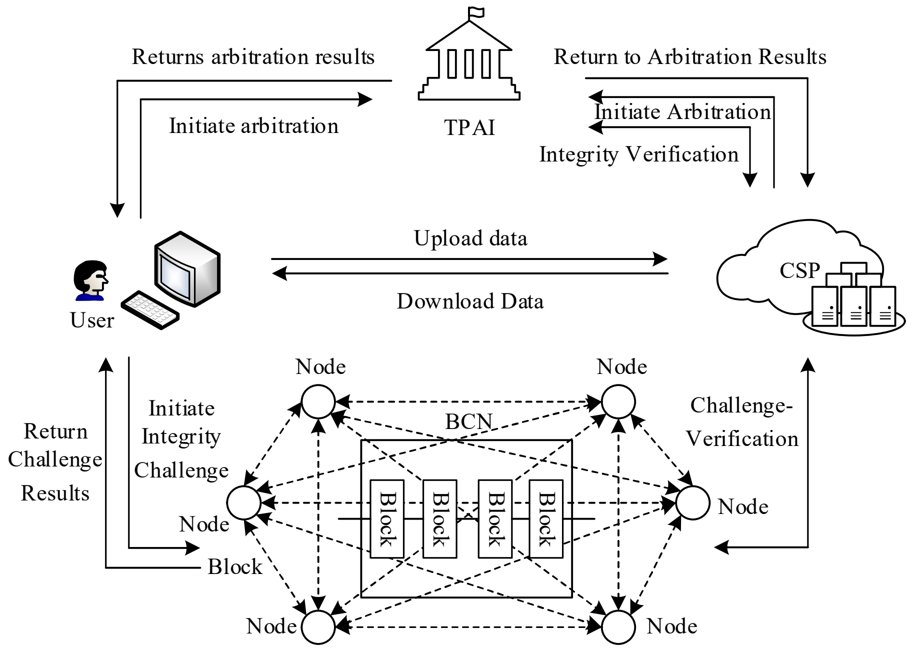


Figure 3: Sensitive data integrity verification scheme

The whole scheme can be divided into three parts:

(1) Data uploading. User encrypts the data before uploading it and generates a data integrity proof tree (DIPT) and an auxiliary verification information tree (AVIT), and then uploads the DIPT and AVIT into the blockchain network.

(2) Challenge-Answer: User initiates integrity verification challenge, smart contract in BCN forwards User's challenge to CSP, CSP receives and answers the challenge, and then sends the answer result to smart contract.

The smart contract verifies the answer based on the AVIT stored in the block and sends the verification result to User.

(3) Data Integrity Verification: After receiving the successful challenge result from the smart contract, User initiates data integrity verification to TPAI, which utilizes DIPT stored in the block to verify the integrity of the data on the CSP and returns the verification result to User; meanwhile, the CSP can initiate verification to TPAI against the illegal challenge of User.

III. B. 2) Integrity check specific constructs

For ease of description, define the notation: G and G_t are 2 q-ordered multiplicative cyclic groups, q is a large prime, g is the generating element of G, $e:(G\times G)\to G_t$ is a bilinear mapping from G to G_t; $hash(\cdot)\to Z_p^*$ is the anti collision hash function, $chal$ is the set of challenges, σ is the labeling evidence, μ is the data evidence, and u and v are the 2 public keys.

(1) $Setup(\lambda)\to(sk,pk)$. After DO inputs the security parameter λ, a probabilistic algorithm is executed to generate DO's private and public keys. Randomly select $\alpha\in Z_p^*$ and compute $v=g^{\alpha}$, select random element $u\in G$ to get DO's private key $sk=\alpha$ and public key $pk=(g,v,u)$, and make pk public.

(2) $TagGen(sk,M)\to\phi$. DO uses the private key sk to split the encrypted file M into n data blocks of fixed size, and then calculates the corresponding data block tag for each data block $m_i(i\in[1,n])$ according to Eq. (5). DO puts all the data block tags into the tag set to be merged and uploaded with the data file for storage. uploaded to the CSP cloud space for preservation. i.e:

$$\sigma_i=\left(hash(i)*u^{m_i}\right)^{\alpha} \tag{5}$$

(3) $ChalGen(M)\to chal$. DC arbitrarily chooses the set of random numbers $R=\left\{R_1=(r_{i1},r_{i2},\cdots,r_{in})_{1\le i\le n},R_2=(r_{i(n+1)})_{1\le i\le n},c\right\}$ is sent to the CSP, where R is the number of n linearly independent vectors in n dimensions and c is the number of challenge blocks. To further reduce the computational burden on DC, the generation of R can be realized using the lookup table method, while DC can simply retrieve the set of random numbers from the table 1 at a time. The CSP receives R and uses R_2 to construct a new n-dimensional vector $y=\begin{pmatrix} r_{1(n+1)}\\ r_{2(n+1)}\\ \vdots\\ r_{n(n+1)}\end{pmatrix}$. The CSP uses R_1 and y to obtain a vector of random numbers a according to equation (6), and publishes a random number $r=a_1\|a_2\|\cdots\|a_n$, by which the set of blocks of data to be extracted is determined. If the total number of data blocks is n and c data blocks need to be extracted for validation, the data is divided into c regions of capacity (n/c) (rounded down). The subscripts of the extracted data blocks are obtained according to Eq. (7), and then the corresponding random numbers v_i are selected from the finite field to obtain the challenge set, i.e., $chal=\{i,v_i\}_{0\le i\le c}$.

$$a=\begin{pmatrix}(r_{11},r_{12},\cdots,r_{1n})\\(r_{21},r_{22},\cdots,r_{2n})\\ \vdots \\(r_{n1},r_{n2},\cdots,r_{nn})\end{pmatrix}^{-1},y=\begin{pmatrix}a_1\\a_2\\ \vdots\\a_n\end{pmatrix}\bmod p \tag{6}$$

$$i=r\%(n/c)+i*(n/c) \tag{7}$$

(4) $\Pr oofGen\left(M,\ \phi,chal\right)\to\Pr oof$. After the CSP obtains the set of challenges $chal$, it generates the evidence of data completeness according to Eqs. (8) and (9) $\Pr oof=\{\sigma,\mu\}$. I.e:

$$\sigma=\prod\nolimits_{(i,v_i)\in chal}(\sigma_i)^{v_i} \tag{8}$$

$$\mu = \sum_{(i,v_i)\in chal} v_i \cdot m_i \tag{9}$$

CSP computes $C = \prod_{(i,v_i)} (H(i)^{v_i} \cdot u^{\mu})$, noting that $A = \sigma$, with $B = g$ and $D = v$. CSP verifies the evidence by computing $z_1 = e(A,B)$ and $z_2 = e(C,D)$, comparing z_1 and z_2 are equal or not, and send the result along with z_1 and z_2 to DC.

(5) $Verify(pk, chal, Proof, z_1, z_2) \to true/false$. DC verifies the validity of the challenge sets $chal$, z_1, z_2, and $Proof$ one by one after receiving them from CSP. For the challenge set $chal$, the DC verifies the validity of r by equation (10) using $R_1 = \{V_i\}_{1\le i\le n}$, where $V_i = (r_{i1}, r_{i2}, \cdots, r_{in})$. If the validation fails, it indicates that the CSP has provided false random numbers, and the DC refuses to continue the validation and reports this result to the DO and the CSP, otherwise the validation continues. I.e:

$$V_i \cdot a = (r_{i1}, r_{i2}, \cdots, r_{in}) \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = r_{i(n+1)} \tag{10}$$

For z_1, z_2 and Proof, DC randomly selects $x_1 \in Z_p^*$, $x_2 \in Z_p^*$, $y_1 \in Z_p^*$ and $y_2 \in Z_p^*$, and computes the following parameters (X_1, X_2, α and β can be computed in advance, and look up the table during validation) obtained): $X_1 = g^{x_1}$, $X_2 = g^{x_2}$; $\alpha = e(g,g)^{x_1 \cdot x_2}$; $T_1 = A^{1/x_2} \cdot X_1$, $T_2 = B^{1/x_1} \cdot X_2$; $Y_1 = g^{y_1}$; $Y_2 = g^{y_2}$; $\beta = e(g,g)^{y_1 \cdot y_2}$; $U_1 = C^{1/y_2} \cdot Y_1$; $U_2 = D^{1/y_1} \cdot Y_2$. DC sends $\pi = (T_1, T_2, U_1, U_2)$ to the CSP. The CSP computes Eqs. (11) and (12) and sends $\{z_i\}_{3\le i\le 4}$ to the DC. I.e:

$$z_3 = e(T_1, T_2)[e(g,B)e(A,g)]^{-1} \tag{11}$$

$$z_4 = e(U_1, U_2)[e(g,D)e(C,g)]^{-1} \tag{12}$$

DC determines whether both Eq. (13) and Eq. (14) are valid, if they are valid then the CSP has truly performed the bilinear pair operation in the evidence validation computation, otherwise the validation is terminated and the result of this validation is recorded as $false$. DC then compares whether z_1 is equal to z_2, if it is equal then the result of this validation is $true$, otherwise it is $false$. After the DC validation is complete, the validation result $(true/false)$ is reported to the DO. ie:

$$z_3 = \alpha \cdot z_1^{1/(x_1 \cdot x_2)} \tag{13}$$

$$z_4 = \beta \cdot z_2^{1/(y_1 \cdot y_2)} \tag{14}$$

III. B. 3) Optimization of integrity checking scheme

In order to avoid server-side spoofing and to reduce storage pressure on the client side, a new parameter is introduced on top of the basic scheme: the block identifier BI. When generating tag, the client generates the corresponding BI for each file block based on the held secret and adds it to the original tag. This computation is done in the preparation phase by the following function:

$$BIgenerate(sk, i, filename) \to BI_i \tag{15}$$

$$PrepareUpdate(sk, pk, F) \to T\{t\{M_1\}, t\{M_2\}, \cdots, t\{M_s\}\} \tag{16}$$

In this scheme, $t\{M_i\} = BI_i \times g^{m_i} \bmod N$.

In the validation phase, the client needs to first remove BI of the specified file block from Y before performing the verification of content consistency and integrity. Therefore, before performing the verification process, the client first executes the following function: $removeBI(sk, S, P) \to P'$.

Because the client removes only the BI of the specified data block, all credentials computed using non-specified data blocks fail to satisfy consistency and integrity. The server is also unable to spoof the user with the computed results of non-specified file blocks.

In some cases, a user needs to update a file on a file server to a new version. However, each version of the file is complete and consistent, and it is possible for the server to maintain the old version of the file and still be able to validate the file at all times. Therefore, customers need to be able to verify the correct version of a file.

To fulfill this need, this solution introduces a new parameter: version serial number s, which helps the user to verify the version serial number. When the client prepares a file for upload, unlike the basic scheme, the key at the time of generation of tag is updated from g to g^s. When the content of the file block remains unchanged but the version is updated, the client only needs to use s'/s as the re-encryption key and perform the corresponding re-encryption computation to complete the update of the version number. For a data block with updated content, the user is required to re-update tag. The related functions are as follows:

$$Seria\ln umber\left(sk,i\right)\rightarrow s \tag{17}$$

$$BIgenerate(sk,i,filename)\rightarrow BI_i \tag{18}$$

$$\Pr epareUpdate(sk,pk,F)\rightarrow T\{t\{M_1\},t\{M_2\},\cdots,t\{M_s\}\} \tag{19}$$

In this scheme, $t\{M_i\}=BI_i\times g^{sm_i} \bmod N$, $Tagupdate(t,s'/s)\rightarrow t'$ is the corresponding re-encryption algorithm.

By utilizing the version number parameter, this scheme can easily update the version information of a file block with unchanged content, and can prevent the server from deceiving the user by utilizing the old version of the file.

IV. Program validation for trusted execution environments

In order to solve the security problems existing in the traditional architecture, trusted computing came into being. The goal of trusted computing is to introduce a root of trust in the computing system, and then establish a chain of trust that extends the trust relationship from the underlying hardware to the upper layer applications, in order to enhance the security of the computing system. Trusted computing plays an important role in protecting terminal security and has now become a standardized technology.

IV. A. Calibration efficiency and program evaluation

IV. A. 1) Audit efficiency tests

In this paper, a trusted execution environment framework based on Intel SGX technology is implemented using the PBC library, developed in C. The system parameters are Ubuntu Linux, 32GB of RAM, and 160GB of hard disk. In each sensitive data verification cycle, files with different hotness are generated and randomly corrupted in accordance with the law of two-eighths, and the size of the test file is 0.5MB.Select B-PDP and SA-PDP as a comparison program, set the test time to 48 hours, the number of files is set to 600~1900, and the total execution time of the files in the process of sensitive data verification is counted. Figure 4 shows the comparison results of the efficiency of sensitive data integrity verification.

As can be seen from the figure, when the sensitive data verification files reach 1900, the total execution time of the sensitive data integrity verification scheme given in this paper is reduced by 51.29% and 32.76% compared to the total execution time of the files of B-PDP and SA-PDP, respectively, and the audit time of this paper's sensitive data integrity verification scheme has a flat growth trend. This is because the B-PDP scheme audits documents according to a fixed frequency causing documents to be audited frequently, and the SA-PDP scheme distributes the document audit tasks over multiple time cycles according to the document audit requirements, which increases unnecessary audits to some extent. The more time the sensitive data integrity verification scheme saves, the more files can be verified. The verification execution time in the B-PDP scheme is slightly higher than in the SA-PDP method because, with the B-PDP scheme, the files require the system to respond to their verification requests, resulting in an increase in the number of file verifications, and the SA-PDP scheme sacrifices some of the verification efficiency to increase the intensity of file verification.

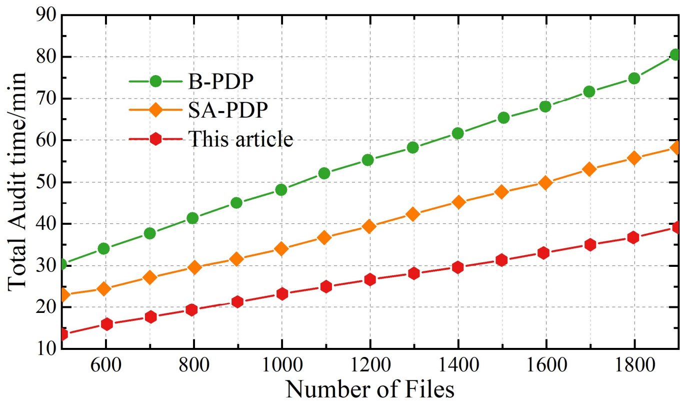


Figure 4: Comparison of the efficiency of integrity verification for Sensitive data

Set the test time as 48h, every hour to the verification system to add 2000 files, the test system can verify the number of files, the comparison results are shown in Figure 5. As can be seen from the figure, in the early stage of sensitive data integrity verification, the B-PDP program checks the number of files more than the SA-PDP program, while the sensitive data integrity verification program designed in this paper checks more files than the other two programs. In this paper, the scheme distributes the sensitive data verification tasks in multiple cycles, and at the early stage of verification, some files are not verified because they are not in the verification cycle. At the later stage of verification, the B-PDP scheme wastes a large amount of computational resources by frequent verification, which makes the system reach the verification limit prematurely, while the SA-PDP scheme sacrifices part of the computational resources with its positive verification characteristics, so that the number of verified files is lower than that of the sensitive data integrity verification scheme designed in this paper.

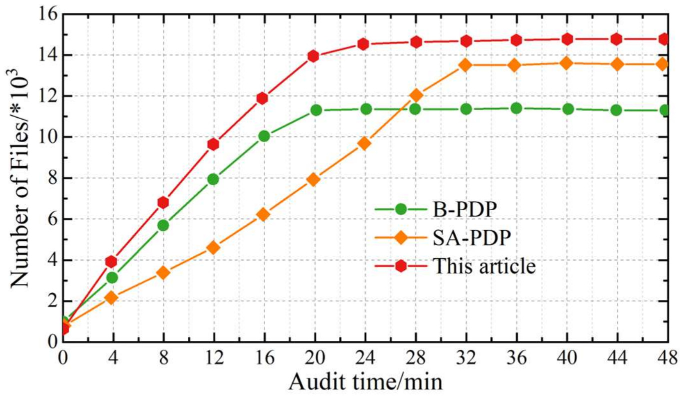


Figure 5: The comparison of the number of verifiable files

IV. A. 2) Overall program assessment

The above experimental data are used randomly generated data, and each data is set to be 0.5MB. In order to enhance the diversity of the data, this paper uses the MNIST dataset for the experiments, which is divided into the test dataset and the training dataset, in which the test dataset contains 12,000 samples of data with a data size of 7.5MB, and we divide it into 30KB/strip and 60KB/strip of data for experiments. The training dataset contains 50,000 sample data with a data size of 45MB, which is divided into 100KB/bar and 200KB/bar for experimentation. Table 1 shows the time overhead of this paper's scheme on MNIST dataset under different stages.

As can be seen from the table, the insertion and query operations of the sensitive data integrity checking scheme designed in this paper are basically unaffected by the size of the data volume inserted and queried, and the execution time of the trusted execution environment is affected by the size of the data volume. Therefore, the same total data volume is divided into multiple data for cloud storage, the larger the single data volume, the longer the trusted execution environment execution time will be, but the number of times it needs to be stored will be relatively reduced. The smaller the amount of individual data, the shorter the execution time of the trusted execution environment, but the number of times it needs to be stored increases. In addition, for the overall verification of the program in different stages of storage and verification time overhead, based on the support of the trusted execution environment technology, it can be seen that the storage and verification phase of the time overhead of only 0.728ms and 0.291ms, to a certain extent, can be obtained with a high degree of time efficiency.

Table 1: Integrity verifies the time overhead of each phase

Time overhead (ns)		Test dataset		Training dataset	
		30KB	60KB	100KB	200KB
Storage stage	Execute	1281057	1759428	3827916	2402961
	Store	1306	1115	1158	1072
Verification phase	Execute	850342	1314126	3128217	4894237
	Store	938	998	990	1030

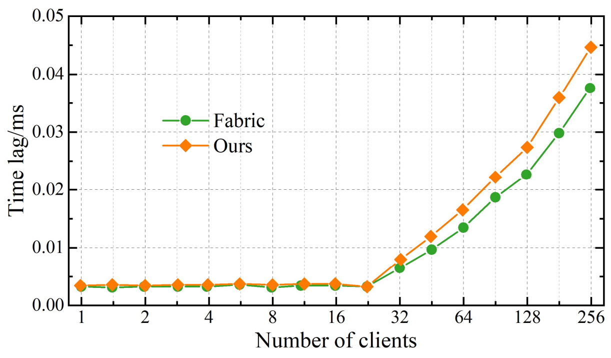
IV. B. Verification of Trusted Execution Environment

IV. B. 1) Implementation environment performance

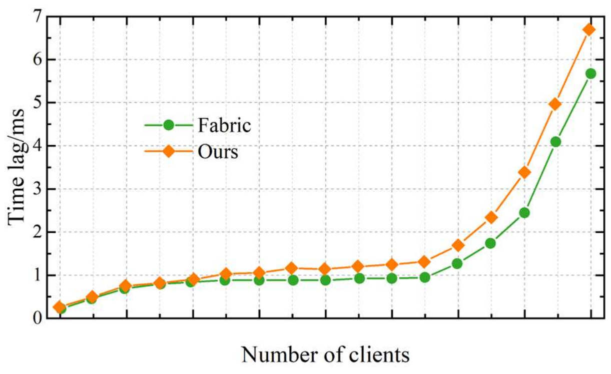
The trusted execution environment designed in this paper is developed based on Intel SGX and Hyperledger Fabric, built using EGov, SGX DCAP driver and efabric/base:0.1 image. The host device uses an Intel Core (TM) i5-10300 CPU with 4 cores and SGX support, and the entire network was built and tested on Ubuntu. Two organizations are used to simulate three enterprises or institutions in the blockchain network respectively, with a total of six Peer nodes and three Orderer nodes, and all clients and nodes are deployed on the same host.

In this paper, Hyperledger Caliper is used to test the Trusted Execution Environment network. First, this paper selects the high throughput chain code in the Hyperledger Fabric sample to test the variation of latency and throughput during the increase of the number of users from 1 to 256, and then compares this variation with the original Fabric network. In order to exclude the effect of network fluctuations, 10 sets of data were tested in each round and the average value was taken as the final data. Reading the ledger information (Fabric Query operation) does not create new blocks or endorsements, and it returns the reading result directly, which is characterized by low latency and high efficiency. Writing book information (Fabric Invoke operation) requires endorsement, distribution and verification, and its overhead is higher. These two operations are very different in performance, so this paper tests for these two cases separately. The relationship between network latency and the number of clients is shown in Fig. 6, where Figs. 6 (a)~(b) show the time delays of Query and Invoke operations, respectively. The relationship between the throughput rate and the number of clients is shown in Fig. 7, where the throughput rate refers to the number of transactions per second, and Fig. 7 (a)~(b) shows the throughput rate of Query and Invoke operations, respectively.

From Fig. 6 and Fig. 7, it can be seen that the latency of the trusted execution environment network designed in this paper is elevated by about 6.42%, the throughput rate is decreased by about 5.03%, and the overall performance reaches about 95.48% of the original Fabric performance, which is within the acceptable range. In the case of both read ledger and write ledger operations, the latency of the trusted execution environment network designed in this paper is slightly higher than that of the original Fabric network, and its throughput rate is slightly lower than that of the original Fabric network. Considering that Enclave creation and operation need extra overhead, and the trusted execution environment network designed in this paper adds the remote authentication process between Peer node and chain code, and the information interaction between the two needs to be encrypted/decrypted, which all need extra overhead. Therefore, it is normal that the performance of the trusted execution environment network in this paper decreases appropriately.

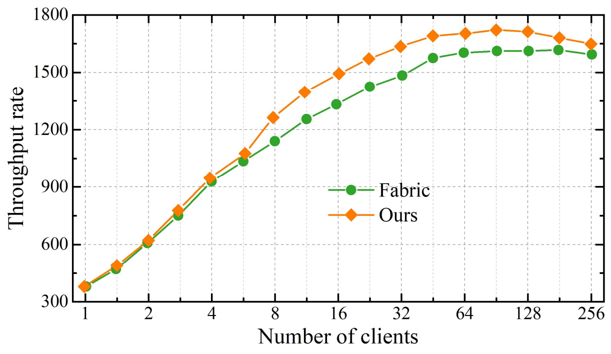


(a) Query operation time delay

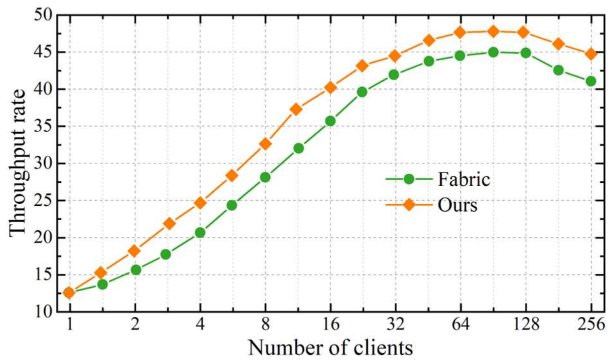


(b) Invoke operation time delay

Figure 6: The relationship between network latency and the number of clients



(a) Query operation throughput rate



(b) Invoke operation throughput rate

Figure 7: The relationship between throughput rate and the number of clients

IV. B. 2) Different network performance

In order to further illustrate the performance of the trusted execution environment network designed in this paper, the FPC network is chosen as a comparison, and its specific comparison results are shown in Table 2. Compared with FPC, the trusted execution environment framework in this paper is more lightweight and has a smaller TCB.FPC uses Go language and C language to reconstruct the whole Fabric architecture, which requires the creation of classified ledger Enclave and chain code Enclave, and the system consists of about 3,000 lines of trusted C/C++ code, of which the ledger Enclave has about 2,400 lines of code and the chain code Enclave has about 600 lines of code. In addition to the system code, the FPC also needs to load the chain code written by the user into the Enclave, which has 1,000~4,000 lines of code, so the total number of lines of trusted code is 4,000~8,000 lines. Based on the test results in the existing literature, it can be seen that the overall performance of FPC reaches 90.24% of the original Fabric. Compared with FPC, the trusted execution environment designed in this paper uses 2 chained code Enclave and is extended at runtime without adding trusted code to the system. The Enclave only loads the Go language chain code written by the user, and the number of lines of code is 300~4000 lines, so it has a smaller TCB. Therefore, the overall performance of the Trusted Execution Environment constructed based on the SGX technology in this paper is better, and it can provide a reliable support for realizing the effective discovery and verification of sensitive data.

Table 2: Different network performances

Name	Number of enclaves	The number of trusted codes	Performance
FPC	4	4000~8000	90.24% of the original Fabric
Ours	2	300~4000	95.48% of the original Fabric

V. Conclusion

In this paper, the key technical problems of sensitive data protection in cloud storage environment are effectively solved by constructing a trusted execution environment framework based on Intel SGX and a multi-branch path tree integrity verification mechanism. The experimental results show that the scheme reduces the total execution time by 51.29% compared to the traditional B-PDP scheme and 32.76% compared to the SA-PDP scheme when processing 1900 files, which significantly improves the data verification efficiency. Performance tests based on the MNIST dataset show that the time overhead of the system is only 0.728ms in the storage phase and 0.291ms in the verification phase, demonstrating good time efficiency characteristics. The overall performance of the Trusted Execution Environment network reaches 95.48% of the original Hyperledger Fabric, maintaining high system performance while introducing security protection mechanisms.

The solution realizes confidentiality and integrity protection of sensitive data during cloud processing through hardware-level security mechanisms, effectively resisting attack threats from privileged software. The application of the multi-branch path tree data structure reduces the depth of the authentication path and lowers the verification overhead, while the integration of the bilinear pair algorithm further enhances the security of the checksum. The integration of blockchain technology builds a decentralized trust mechanism, avoids the single-point-of-failure problem, and improves the reliability and transparency of the system. The optimization of the version control mechanism supports the integrity verification requirements in dynamic data environments and enhances the practicality of the scheme.

This research provides innovative solution ideas for the security protection of sensitive data in cloud storage environment, which has important theoretical value and practical application prospect.

Funding

This work was supported by Science and Technology Project of State Grid Qinghai Electric Power Company (No: 52281424000D).

References

[1] Issa, I., Wagner, A. B., & Kamath, S. (2019). An operational approach to information leakage. IEEE Transactions on Information Theory, 66(3), 1625-1657.

[2] Qamar, A., Karim, A., & Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. Future Generation Computer Systems, 97, 887-909.

[3] Shiau, W. L., Wang, X., & Zheng, F. (2023). What are the trend and core knowledge of information security? A citation and co-citation analysis. Information & Management, 60(3), 103774.

[4] Wang, M., & Jiang, Z. (2017). The defining approaches and practical paradox of sensitive data: An investigation of data protection laws in 92 countries and regions and 200 data breaches in the world. International Journal of Communication, 11, 20.

[5] Sargiotis, D. (2024). Data security and privacy: Protecting sensitive information. In Data governance: a guide (pp. 217-245). Cham: Springer Nature Switzerland.

[6] Levina, A., Mostovoi, R., Sleptsova, D., & Tcvetkov, L. (2019). Physical model of sensitive data leakage from PC-based cryptographic systems. Journal of Cryptographic Engineering, 9, 393-400.

[7] Guri, M., Zadov, B., & Elovici, Y. (2019). Odini: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields. IEEE Transactions on Information Forensics and Security, 15, 1190-1203.

[8] Sha, L., Xiao, F., Chen, W., & Sun, J. (2018). IIoT-SIDefender: Detecting and defense against the sensitive information leakage in industry IoT. World Wide Web, 21, 59-88.

[9] Cam, N. T., Pham, V. H., & Nguyen, T. (2019). Detecting sensitive data leakage via inter-applications on Android using a hybrid analysis technique. Cluster Computing, 22(Suppl 1), 1055-1064.

[10] Verma, V. (2023). Cutting-Edge AI Techniques for Data Anonymization and Privacy Protection in Sensitive Data Contexts. Eastern European Journal for Multidisciplinary Research, 2(1), 50-57.

[11] Yang, Z., & Liang, Z. (2018). Automated identification of sensitive data from implicit user specification. Cybersecurity, 1, 1-15.

[12] Yi, Y., Zhu, N., He, J., Jurcut, A. D., Ma, X., & Luo, Y. (2024). A privacy‐sensitive data identification model in online social networks. Transactions on Emerging Telecommunications Technologies, 35(1), e4876.

[13] Kužina, V., Petric, A. M., Barišić, M., & Jović, A. (2023). CASSED: context-based approach for structured sensitive data detection. Expert systems with applications, 223, 119924.

[14] Yang, R., Gao, X., & Gao, P. (2021, January). Research on intelligent recognition and tracking technology of sensitive data for electric power big data. In 2021 13th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA) (pp. 229-234). IEEE.

[15] Cui, Y., Huang, Y., Bai, Y., Wang, Y., & Wang, C. (2024). Sensitive data identification for multi‐category and multi‐scenario data. Transactions on Emerging Telecommunications Technologies, 35(5), e4983.

[16] Khan, N., Nitzsche, S., López, A. G., & Becker, J. (2021). Utilizing and extending trusted execution environment in heterogeneous SoCs for a pay-per-device IP licensing scheme. IEEE Transactions on Information Forensics and Security, 16, 2548-2563.

[17] Siddiqui, H., Idrees, M., Gudymenko, I., Le Quoc, D., & Fetzer, C. (2021, October). Credentials as a service providing self sovereign identity as a cloud service using trusted execution environments. In 2021 IEEE International Conference on Cloud Engineering (IC2E) (pp. 210-216). IEEE.

[18] Xie, H., Zheng, J., He, T., Wei, S., & Hu, C. (2023). TEBDS: A trusted execution environment-and-blockchain-supported IoT data sharing system. Future Generation Computer Systems, 140, 321-330.

[19] Cheng, J., Li, J., Xiong, N., Chen, M., Guo, H., & Yao, X. (2020). Lightweight Mobile Clients Privacy Protection Using Trusted Execution Environments for Blockchain. Computers, Materials & Continua, 65(3).

[20] Zhenzhou Tian,Yudong Teng,Xianqun Ke,Yanping Chen & Lingwei Chen. (2025). SolBERT: Advancing solidity smart contract similarity analysis via self-supervised pre-training and contrastive fine-tuning. Information and Software Technology,184,107766-107766.

[21] Zahra Yaqoubi,Behrouz Shahgholi Ghahfarokhi,Faegheh Seifhashemi & Mojtaba Mahdavi. (2025). Mobile crowd sensing based spectrum monitoring with privacy protection and malicious behavior detection using hyperledger fabric and identity mixer. The Journal of Supercomputing,81(7),850-850.

[22] Changhee Han,Taehun Kim,Woomin Lee & Youngjoo Shin. (2024). S-ZAC: Hardening Access Control of Service Mesh Using Intel SGX for Zero Trust in Cloud. Electronics,13(16),3213-3213.