electronics

MDPI

Article

TrustHealth: Enhancing eHealth Security with Blockchain and Trusted Execution Environments

Jun Li [1,2], Xinman Luo [3,*] and Hong Lei [1,2]

[1] School of Cyberspace Security, Hainan University, Haikou 570228, China
[2] SSC Holding Company Ltd., Chengmai 571924, China
[3] School of Information Science and Technology, Qiongtai Normal University, Haikou 571127, China
* Correspondence: luoxinman@mail.qtnu.edu.cn

Abstract: The rapid growth of electronic health (eHealth) systems has led to serious security and privacy challenges, highlighting the critical importance of protecting sensitive healthcare data. Although researchers have employed blockchain to tackle data management and sharing within eHealth systems, substantial privacy concerns persist as a primary challenge. In this paper, we introduce TrustHealth, a secure data sharing system that leverages trusted execution environment (TEE) and blockchain technology. TrustHealth leverages blockchain to design smart contracts to offer robust hashing protection for patients' healthcare data. We provide a secure execution environment for SQLCipher, isolating all sensitive operations of healthcare data from the untrusted environment to ensure the confidentiality and integrity of the data. Additionally, we design a TEE-empowered session key generation protocol that enables secure authentication and key sharing for both parties involved in data sharing. Finally, we implement TrustHealth using Hyperledger Fabric and ARM TrustZone. Through security and performance evaluation, TrustHealth is shown to securely process massive encrypted data flows at a rate of 5000 records per second, affirming the feasibility of our proposed scheme. We believe that TrustHealth offers valuable guidelines for the design and implementation of similar systems, providing a valuable contribution to ensuring the privacy and security of eHealth systems.

Keywords: eHealth; TEE; blockchain; data sharing; security

Citation: Li, J.; Luo, X.; Lei, H. TrustHealth: Enhancing eHealth Security with Blockchain and Trusted Execution Environments. *Electronics* **2024**, *13*, 2425. https://doi.org/10.3390/electronics13122425

Academic Editor: Hung-Yu Chien

Received: 7 May 2024
Revised: 15 June 2024
Accepted: 17 June 2024
Published: 20 June 2024

1. Introduction

The rapid digitization of healthcare services has led to the widespread adoption of electronic health (eHealth) systems. These systems are pivotal in streamlining operations, from patient data management to diagnostic processes, thereby significantly improving the quality of healthcare [1]. However, this digital transformation also introduces substantial security risks, particularly regarding the privacy and integrity of electronic health records (EHRs). The vast volumes of sensitive data managed by eHealth systems are attractive targets for cyber threats, ranging from data breaches to unauthorized data manipulation. EHR data are used in a variety of scenarios related to healthcare management and patient care, such as early detection of disease [2]. Outsourcing EHR to the cloud is a common solution to these challenges [3,4]. This has the benefit of making eHealth system services more convenient, efficient, and flexible, but there are certain issues with data security and privacy. Once a healthcare provider uploads a patient's EHR to a cloud server, the patient no longer retains physical control over their data [5–7]. Moreover, current cloud service providers merely offer assurances to protect data privacy to the extent that minimizes their own risk [8]. Additionally, many instances of medical malpractice regarding the leakage of patients' medical records have occurred [9,10]. From the perspectives of both patients and healthcare organizations, the privacy of medical data must be inviolable, as EHRs represent some of the most sensitive and critical data. Furthermore, the COVID-19 pandemic also

highlights the need to protect the privacy of healthcare data [11]. As a result, eHealth systems are eager to innovate with new technologies to ensure the security and privacy of healthcare data.

Recently, the emergence of innovative technologies such as blockchain [12] and trusted execution environment (TEE) [13] has offered a promising avenue for enhancing data security and protection. The immutability, auditability, and transparency of blockchain can protect data in distributed network work as compared to traditional centralized solutions [14]. Due to its decentralized, tamper-proof, and distributed nature, blockchain ensures that each operation on EHR is recorded as a transaction, making it immutable and resistant to tampering. Thus, any patient can check the access record by checking the on-chain transactions. Numerous studies have investigated the application of blockchain and smart contracts to bolster the security of eHealth systems [1,15]. Compared to traditional centralized approaches, these studies use blockchain as a decentralized platform to provide functions such as storage, sharing, and auditing of healthcare data, which has many benefits but still has some drawbacks. Firstly, compared to traditional database solutions, blockchain-based electronic health systems exhibit lower efficiency in data retrieval [16]. Secondly, the inherent limitations of blockchain technology result in high storage costs for large datasets. The decentralized and distributed nature of blockchain requires each participating node to maintain a complete copy of the entire blockchain. As the volume of data grows, the storage requirements for each node increase significantly, leading to substantial costs. While redundancy enhances fault tolerance and immutability, it also introduces significant storage overhead [17]. Moreover, storing plaintext data on the blockchain raises serious privacy concerns, especially for sensitive information like EHRs [18]. Fortunately, TEE focuses on runtime protection and secure storage of data [19]. To protect the confidentiality and integrity of data, traditional cryptographic primitives have been widely adopted by cloud service providers, such as public key encryption [20–23], digital signatures [24,25], and message authentication codes [26]. However, devices are easily exposed to various attacks in an open environment. Compared to software-only data protection solutions, those that integrate TEE can protect private data with enhanced security, leveraging robust hardware isolation mechanisms. With the combination of blockchain and TEE, the security and privacy of EHR in the eHealth system can be further improved. Thus, we can integrate blockchain and TEE into eHealth systems to achieve a high-level data privacy protection framework.

This paper introduces TrustHealth, a robust eHealth security framework that harnesses the strengths of both TEE and blockchain technology. The blockchain records fingerprints of operations, including data summaries, ensuring their immutability, while the TEE is employed to store EHR. We extend the functionalities of the open portable trusted execution environment (OP-TEE) [19], a widely used, open-source, trusted operating system designed for TrustZone. This extension not only ensures the confidentiality and integrity of the database but also establishes an isolated environment for the execution of sensitive operations during the processing of EHR. Our key contributions can be summarized as follows:

- A secure database design atop TrustZone hardware ensures the confidentiality and integrity of EHRs.
- We designed a TEE-empowered secure session key generation protocol to create a secure data-sharing channel between TEE and hospitals or healthcare institutions.
- We performed a thorough security and performance analysis, demonstrating that TrustHealth is both efficient and practical. Experimental results indicate that TrustHealth can securely handle a large volume of encrypted data flows at a rate of 5000 records per second.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 presents Preliminaries. Section 4 describes the system design of TrustHealth, emphasizing the architecture and key components. Section 5 details the implementation.

In Section 6, the evaluation of the proposed system is presented. Section 7 discusses the security analysis. Finally, Section 8 presents the conclusion and future work.

2. Related Work

Recently, blockchain technology has been widely adopted to decentralize data management, thereby reducing the inherent risks associated with centralized systems. Blockchain's immutable ledger and consensus mechanisms ensure that all transactions are transparent and tamper-proof, making unauthorized data alterations easily detectable. Zhou et al. [27] introduced a human-in-the-loop-aided approach that employs a block design technique for privacy preservation in smart healthcare settings. However, the scheme requires training multiple machine learning models, which may increase computational costs and delays. Li et al. [28] introduced a novel public audit scheme that utilizes blockchain technology to verify data integrity. This scheme aims to tackle the issue of tampering with remote data stored in the cloud. Similarly, Yang et al. [4] implemented a shared medical record solution in the cloud using vertical partition of healthcare datasets. However, in cloud storage environments, there is usually a lack of trust between data owners and cloud service providers, so a mechanism is needed to ensure that cloud service providers do not leak or tamper with data. The HealthDep scheme [29] minimized server storage costs and enhanced healthcare data security through an efficient and secure encrypted EMR deduplication strategy. However, while these schemes focus on securing remotely stored data in the cloud, patients still lack real control over their medical data due to the distrust of cloud providers and open operating systems.

Recent studies have leveraged blockchain technology to empower patients with enhanced control over their medical data. The transparent and immutable nature of blockchain enables patients to dynamically manage access permissions to their EHRs, allowing them to grant or revoke access to various parties involved in their healthcare. This level of granularity and flexibility is pivotal in safeguarding sensitive health information and addressing the trust concerns associated with traditional cloud providers. MeDShare utilized blockchain technology to enhance medical data sharing among large data entities by offering data provenance, auditing, and control mechanisms [30]. In another major study, MedBlock highlighted the need to propose a blockchain-based information management system that exhibits high information security while enabling efficient EMR access and inspection in a distributed ledger [31]. Huang et al. [32] provided an in-depth analysis of zero-knowledge proof and the proxy re-encryption technology, showing that the blockchain-based privacy-preserving scheme satisfies confidentiality, integrity, and availability. Collectively, these studies employ either permissioned or public blockchains, depending on the specific requirements of their respective applications. Azaria et al. [33] introduced MedRec, an innovative record management system that leverages blockchain technology and smart contracts to effectively manage the EMRs of patients. MedRec not only supports patient-defined access control rules but also provides a unique reward system. The former enables access to medical information between different institutions, and the latter incentivizes medical stakeholders to actively participate in the network as "miners". Jiang et al. [34] proposed an IoT access control model CcBAC based on blockchain and cryptocurrency and supported by trusted execution environment technology. This model offers fine-grained, robust auditability and access process control, and its practicality has been verified through theoretical analysis and experimental evaluation. Cao [3] centered their research on blockchain technology, proposing a robust cloud-assisted eHealth system designed to safeguard outsourced EHRs against unauthorized tampering. Mandarino et al. [18] propose a hybrid storage strategy, where medical records are stored on users' devices, and blockchain merely manages the index of these data, thereby enhancing security and cost-effectiveness. Patients can set authorization policies through smart contracts, ensuring that only authorized entities can access their medical records. However, this system's reliance on the storage and computational capabilities of user devices may lead to data availability and security issues in the event of device failure or loss. However, its

access control mechanism is somewhat coarse and lacks transparency. Although these studies have made considerable advancements, they still present certain limitations that require addressing. One of the main issues is the increased transmission, computation, and storage burden associated with blockchain, which needs to be considered when storing large amounts of data in practical scenarios. Additionally, the storage of significant amounts of data on the blockchain is very resource-intensive and is not considered from the perspective of hardware-assisted security.

Table 1 shows a summary of the comparison among several EHR solutions. Most blockchain-based EHR systems focus on addressing access control and ensuring the security of both on-chain and off-chain data. Access control is typically implemented using smart contracts, while data encryption is employed to protect the security of medical data. However, relying solely on encryption is insufficient to ensure the security of data during its use. It is crucial to ensure that medical data are both accessible to authorized entities and protected from unauthorized access during processing. In this study, we propose TrustHealth by integrating TEE with blockchain technology to address this critical issue. The use of TEE ensures that data remains secure even during its use, providing a secure environment for processing sensitive medical information without exposing it to potential threats. By leveraging TEE, TrustHealth ensures that medical data are not only protected at rest and in transit but also during computation, thereby enhancing the overall security framework of EHR systems. Furthermore, TrustHealth mitigates the risks posed by cloud service providers and operating systems. Traditional cloud environments and open operating systems may expose medical data to unauthorized access and tampering. However, with TEE, even if the cloud service provider or the operating system is compromised, the medical data processed within the TEE remain secure and confidential. This approach significantly reduces the risk of data breaches and unauthorized data manipulation by isolating the execution of sensitive operations within the TEE.

Table 1. Comparison of existing blockchain-based EHR solutions.

Existing Work	Blockchain Types	Platform	Smart Contract	Storages
TP-EHR [3]	Public	Ethereum	Authority management	On-chain: hashes, Off-chain: EHR data
EdgeHR [18]	Public	Ethereum	Access control, Data manage	On-chain: EHR hashes, Off-chain: EHR data
MeDShare [30]	Permissioned	N/A	Data provenance, auditing, and control	On-chain: data access log, Off-chain: medical data
MedBlock [31]	Permissioned	N/A	EMR access and inspection	On-chain: EHR summaries, Off-chain: medical data
Huang et al. [32]	Permissioned	Hyperledger Fabric	Data sharing and verification	On-chain: proofs, hashes, Off-chain: medical data
MedRec [33]	Permissioned	Ethereum	Access control, data sharing, data integrity	On-chain: hashes, metadata, Off-chain: medical records
Jiang et al. [34]	Public/Permissioned	Ethereum	Fine-grained access control, auditability	On-chain: access control policies, Off-chain: IoT data

3. Preliminaries

3.1. Blockchain

Blockchain is a decentralized, distributed ledger technology that ensures data integrity and transparency. There is no central node in the network, and all nodes are equal, using distributed ledger and consensus mechanisms. The failure or exit of a single node does not affect the entire system, providing good fault tolerance and robustness. All transaction data on the distributed ledger is public and transparent, and anyone can query and verify it. Confirmed transaction data are linked into blocks using cryptographic techniques, making subsequent data tamper-proof and ensuring the authenticity and immutability of the data [35].

Blockchain employs various consensus mechanisms to allow nodes to reach a consensus on the correctness of the data. Common consensus mechanisms include Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA), Practical Byzantine Fault Tolerance (PBFT), and Raft [12]. PoW requires nodes to solve complex mathematical problems to validate transactions and add them to the blockchain, which ensures security but is energy-intensive. PoS, on the other hand, allows nodes to validate transactions based on the number of coins they hold and are willing to "stake" as collateral, which is more energy-efficient. Raft is a consensus algorithm designed for managing replicated log consistency, ensuring high throughput and low latency, making it suitable for permissioned blockchain networks where nodes are known and trusted. These consensus mechanisms avoid the risks of single-point failures and data tampering that are common in centralized systems. Additionally, blockchain supports the deployment of smart contracts that automatically execute various application logic. Smart contracts are immutable, and their execution process is transparent and auditable [36].

Blockchain technology can be categorized into three main types based on its access permissions: public, private, and consortium chains. Each type of blockchain has its unique advantages and limitations, and the choice of which type to use depends on the specific requirements of the application, such as the need for transparency, security, and efficiency. Table 2 compares various types of blockchain platforms and their characteristics.

Table 2. Comparison of different types of blockchain.

Type	Public Blockchain	Consortium Blockchain	Private Blockchain
Permission Type	Non-permissioned	Permissioned	Permissioned
Participants	Anyone	Multiple organizations	Single organization
Consensus	PoW, PoS, DPoS	PBFT, Raft, PoA	PoA, PBFT
Efficiency	Low	High	High
Security	High, widely distributed nodes, high decentralization	Medium, security ensured by multiple entities	Low, security relies on a single entity
Examples	Bitcoin, Ethereum	Hyperledger, Fisco Bcos	Multichain, Blockstack

3.2. ARM TrustZone

ARM TrustZone is a hardware security extension integrated into ARM processors. It provides a secure environment known as the TEE that isolates sensitive operations and data from the rest of the system. TrustZone creates two separate worlds: the secure world and the normal world. At the hardware level, system resources (CPU, memory, peripherals, etc.) are divided into two logically isolated execution environments [37]. The secure world has higher execution privileges and can access all system resources without interference from the normal world. The normal world can only access non-secure resources and cannot directly access secure world resources, ensuring the isolation and protection of critical data and code. The secure world provides the foundation for creating a trusted execution environment where security-sensitive code and data processing can be executed. Hardware-assisted security mechanisms ensure that code and data within the trusted environment are protected from theft, tampering, and other attacks. Additionally, the secure world can offer system-level security services such as key management, device authentication, and secure boot. Applications in the normal world can obtain encryption, authentication, and other security capabilities through the security service interfaces provided by the secure world [38].

OP-TEE is an open-source project that operates within the ARM TrustZone, a hardware-based security technology that creates an isolated execution environment. This separation, known as the secure world, is used for handling sensitive operations, while the normal world runs the standard operating system. OP-TEE enhances security by providing secure storage, supporting a range of cryptographic operations, and enabling the development of trusted applications that can perform critical tasks securely [39]. These features make

OP-TEE a crucial component for enhancing security in embedded systems by ensuring that sensitive operations are protected from potential threats in the normal world.

4. System Design

4.1. Threat Model

In TrustHealth, doctors and medical institutions can be trusted only during the treatment period, and they can violate the rules to manipulate EHR after the diagnosis period for different purposes, such as liability for medical disputes or increasing profits. Meanwhile, the cloud server providing various services is a rational entity, as commonly assumed in the existing literature [3,29]. It is important to note that this study does not delve into behavior-related trust issues. Our primary focus is on ensuring data confidentiality, integrity, and authenticity using TEE and blockchain technology. For discussions on behavior trust, readers are referred to the following works [40,41].

Hardware. Assuming that an attacker manages to intrude on the device, we can rely on the hardware-enforced protection provided by the ARM TrustZone security extensions. Rollback attacks are out of our consideration, as they can be mitigated with hardware monotonic counters [42].

Normal world (NW). At the software level, the commodity operating system (COS) is typically considered insecure and susceptible to potential compromises. We consider that a powerful user can obtain permission to operate the file system, including reading, writing, and modifying the database file of SQLCipher. However, we make no assumptions about the NW in a rich execution environment (REE), which includes COS and user space. We note that the Linux kernel supports the OP-TEE driver and works properly.

Secure world (SW). We consider the software components of TEE, such as the security monitor, bootloader, and trusted operating system (TOS), to be secure and immune from vulnerabilities that could compromise the TEE. The trusted application (TA) uses the GlobalPlatform-defined API to interact with OP-TEE OS and cross-world communication channels. Side-channel attacks [43–45] are also beyond the scope of this paper.

4.2. Design Overview

This section provides an overview of our system's approach and key features of TrustHealth before diving into the details. There are five different entities: the patient, hospital, doctor, blockchain platform, and TEE-assisted server.

Patient. The patient, in search of medical services, utilizes a variety of computing devices, including personal desktops and wireless-enabled portable devices, to communicate with the hospital and the TEE-assisted server.

Hospital. The hospital includes multiple departments, such as Surgery, Internal Medicine, Pediatrics, Obstetrics and Gynecology, and more. While hospitals can be considered authorized users who may be curious about the data, they will not operate on patients' private data in an unauthorized or harmful manner.

Doctor. A licensed healthcare professional, the doctor is authorized to offer an extensive array of medical services to patients.

Blockchain platform (BP). The blockchain platform is composed of various nodes that collectively maintain an immutable evidence platform through the operation of a specific consensus algorithm. This platform supports the automatic execution of Turing-complete smart contracts and stores the hash of EHR.

TEE-assisted server (TS). The TS operates as the dedicated data storage server for hospitals, guaranteeing the secure storage and management of EHRs. In response to authorized patient requests for EHR access, the TS is responsible for retrieving and delivering the patient data.

The system architecture is shown in Figure 1. Now, we describe the steps involved in conducting a medical consultation within TrustHealth. The patient initiates identity registration, supplying supplementary information and symptom-related data. These data serve as the basis for generating diagnostic information. The diagnostic information

includes essential details regarding the doctor, the appointment's timing, and location. It is protected through a shared diagnostic key, fostering data security between the hospital and the patient. Subsequently, the patient engages in a scheduled diagnostic and treatment session with the doctor. At the end of the diagnosis, the doctor creates the patient's EHR and corresponding hash value. This EHR hash is then securely stored on the blockchain to uphold data integrity and immutability. A trusted channel is established between the hospital and the TEE-assisted server. This channel enables the doctor to securely outsource the EHR data to the TEE-assisted server for secure storage. Ultimately, patients can readily authenticate the integrity of their EHR data by accessing both the blockchain and TEE-assisted server to retrieve and compare the stored EHR hash value with the corresponding EHR data.

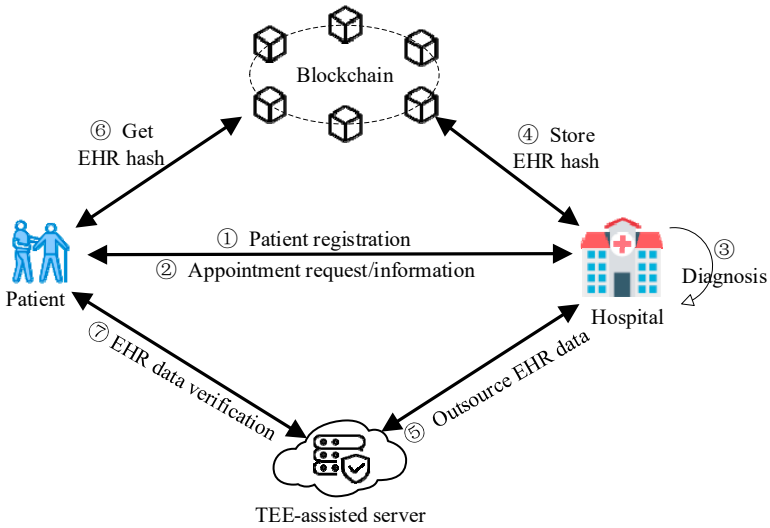

Figure 1. System architecture.

To better illustrate the functionalities and use cases of TrustHealth, consider a patient, Bob, who uses TrustHealth for the first time and grants his doctor, Alice, the necessary permissions to access and update his EHR. Bob then visits Alice for a medical consultation due to some symptoms he is experiencing. During the consultation, Alice accesses Bob's EHR stored on the blockchain. She retrieves the relevant medical history and diagnostic information, ensuring that all access requests are logged and verifiable. In TrustHealth, both patients and hospitals act as clients inter-acting with the blockchain. Patients use their devices to manage and access their medical records, while hospitals update and maintain patient records. Both types of clients leverage Hyperledger Fabric SDK for Go [46] to perform operations such as querying patient records, updating medical information, and managing access permissions. As part of the treatment, Alice updates Bob's EHR to include new diagnostic results and treatment plans. She securely stores these updates by interacting with the TEE-assisted server. The TEE-assisted server ensures that all sensitive operations, such as encryption and decryption of EHR data, are performed in a secure and isolated environment. This process guarantees the confidentiality and integrity of Bob's EHR, protecting it from unauthorized access and tampering.

In TrustHealth, the blockchain stores encrypted EHRs, hash values of EHRs for integrity verification, access control policies, and transaction logs. The TEE-assisted server, utilizing SQLCipher, stores encrypted EHRs, patient data, and cryptographic keys used for data encryption and decryption. The SQLCipher database supports common database operations such as CRUD (INSERT, SELECT, UPDATE, and DELETE), ensuring efficient data management and retrieval within the secure environment provided by the TEE.

4.3. Construction of TrustHealth

In TrustHealth, we assume that each patient completes registration at the hospital to ensure better access to care. In addition, patients are provided with a treatment key before diagnosis, facilitating the establishment of secure communication channels among the patient, hospital, and diagnosing doctor. All diagnostic information is protected by the treatment key.

Initially, the patient schedules an appointment with the hospital and acquires relevant diagnostic information. Before treatment, the patient generates a warrant to delegate the doctor to perform the treatment. The doctor generates an EHR for the patient after the diagnosis is completed. Patients may receive treatment from one or multiple doctors, resulting in multiple EHRs. In either case, each doctor is responsible for the EHR they generated. Subsequently, the hash of the EHR is recorded on the blockchain as a transaction, while the encrypted EHR, along with auxiliary information specific to the doctor, is stored on a TEE-assisted server.

A set of patients $\{P_1, P_2, \ldots, P_n\}$, a TEE-assisted server, a set of doctors $\{D_1, D_2, \ldots, D_x\}$, and a hospital H are involved in TrustHealth. TrustHealth consists of the following stages, as explained below.

Registration and Setup. Each doctor registers a new blockchain account and shares it with others, allowing patients to access information about the doctor, including basic details and diagnostic privileges. The TEE-assisted server also registers a new account on the blockchain and sends it to everyone, enabling medical institutions and individuals to access its public information. Additionally, the TEE-assisted server and the hospital store their identity certificates and public keys on the blockchain platform to facilitate secure authentication. During the initialization process, the TEE-assisted server securely stores all the information of the doctors in the hospital. For a patient, P_i, the hospital randomly assigns a treatment key, P_{tk}.

Appointment and Delegation. The patient and hospital make an appointment, and the hospital assigns a doctor to the patient. The interaction between the patient and the hospital is protected by the treatment key. The hospital sends the appointment details to the patient and the treatment key to the doctor through a secure channel. The patient decrypts and extracts the appointment information to obtain the doctor's ID_D, the valid period $TimePeriod_P$, and some auxiliary information Aux_P, such as specialty and introduction. During delegation, there are two scenarios: (1) a single doctor is responsible for diagnosing and generating, uploading, and storing the EHR, and (2) a group of doctors is responsible for diagnosing the disease, with each doctor generating and uploading their EHR and the last doctor storing the entire EHR.

Case 1: P is diagnosed by a single doctor D_1 in H.

P generates a warrant W_P to delegate the diagnosis and EHR generation to D_1, where

$$W_P = ID_P \,||\, ID_{D1} \,||\, TimePeriod_P \,||\, Aux_P.$$

P creates a structured data object (e.g., JSON) containing the above information. Subsequently, P sends $Enc(P_{tk}, W_P)$ to D_1, where $Enc()$ is a secure symmetric encryption algorithm, such as AES.

Case 2: P is diagnosed by a group of doctors $\{D_i\}(i \in I)$.

There are multiple doctors taking turns diagnosing; P generates a warrant $W_{P,i}$ for a set of doctors $\{D_i\}(i \in I)$ to diagnose and generate EHR, where

$$W_P = ID_P \,||\, ID_{Di} \,||\, TimePeriod_{P,i} \,||\, Aux_{P,i}.$$

P sends $Enc(P_{tk}, W_P)$ to D_i, i = 1, 2, . . ., I.

Upload and Store. Case 1: When D_1 completes P's diagnosis, D_1 generates the EHR ehr_P for P. D_1 hashes ehr_P and uploads it to the blockchain. Then, D_1 encrypts the ehr_p with P_{tk} of P:

$$C_p = Enc(P_{tk}, ehr_P \,||\, W_P).$$

D_1 creates a transaction $Tx(D_1)$, which contains the following data: $H(ID_P)||H(ID_{D1})||H(ehr_P)$, where $H()$ is a hash algorithm, e.g., SHA-256.

D_1 sends C_p to TS. TS will verify D_1's identity by querying the database, and if the verification passes, TS accepts (D_1, C_P, W_P).

Case 2: The first doctor D_1 generates an EHR $ehr_{p,1}$ for P. D_1 hashes $ehr_{p,1}$ and encrypts it with P_{tk}.

$$C_{p,1} = Enc(P_{tk}, ehr_{p,1}||W_{P,1}).$$

D_1 creates a transaction $Tx(D_1)$ that includes the following data:

$$H(ID_P)||H(ID_{D1})||H(ehr_{P,1}).$$

D_1 sends $C_{P,1}$ to doctor D_i, i = 2, 3, . . ., I − 1, then D_i decrypts $C_{P,i-1}$ to obtain $\{ehr_{P,1}, \ldots, ehr_{P,i-1}\}$.

D_i generates an EHR $ehr_{p,i}$ and encrypts it with P_{tk} of P:

$$C_{P,i} = Enc(P_{tk}, ehr_{P,1}||\ldots||W_{P,i}).$$

D_i creates a transaction $Tx(D_i)$, which contains the following data: $H(ID_P)||H(ID_{Di})||H(ehr_{P,i})$.

D_i sends $C_{P,i}$ to D_{i+1}, then D_{i+1} performs the same operation.

For doctor D_I, D_I decrypts $C_{P,I-1}$ to obtain $\{ehr_{P,1}, \ldots, ehr_{P,I-1}\}$.

D_I generates an EHR $ehr_{P,I}$ and encrypts it with P_{tk} of P:

$$C_{P,I} = Enc(P_{tk}, ehr_{P,1}||\ldots||ehr_{P,I}||W_{P,I}).$$

D_I creates a transaction $Tx(D_I)$, which contains the following data:

$$H(ID_P)||H(ID_{DI})||H(ehr_{P,I}).$$

D_I sends $C_{P,I}$ to TS. TS will verify $D_{P,I}$'s identity by querying the database, and if the verification passes, TS accepts $(D_I, C_{P,I}, W_{P,I})$.

Data Sharing. To protect the security of the data sharing phase, we designed a TEE-empowered session key generation protocol combined with blockchain. The TEE-assisted server and hospital establish a key negotiation with authentication, ensuring the establishment of a shared secret key and verifying the authenticity of the communicating parties. The TOS kernel utilizes the Diffie–Hellman (DH) key exchange algorithm provided by the OP-TEE encryption library (either Libtomcrypt [47] or Libmbedtls [48]) to securely generate shared secret keys. The data-sharing mechanism is depicted in Figure 2 and can be briefly summarized as follows.

Step 1: The hospital generates a DH key $A = g^a$ and sends the public key A to the TEE-assisted server.

Step 2: The TEE-assisted server transfers A to the TOS. TOS kernel also generates its own DH key $B = g^b$ and computes the shared secret key SK based on $<A, B>$. To ensure the integrity and authenticity of the exchanged information, the TEE-assisted server signs both A and B, resulting in $Sign_{AB}$. TOS sends B, $Sign_{AB}$, and SK to TA.

Step 3: TA calculates the session key SK_{sess} = (SK, $sessionkey$, $klen$). Then, TA transmits B, $Sign_{AB}$ to the hospital.

Step 4: The hospital receives the data from the TEE-assisted server and begins the verification process. It verifies the validity of $Sign_{AB}$ by accessing the blockchain to obtain the public key of the TEE-assisted server. The hospital then computes the shared secret key SK based on $<A, B>$. Similar to TA, the hospital derives the session key SK_{sess} using the HKDF function.

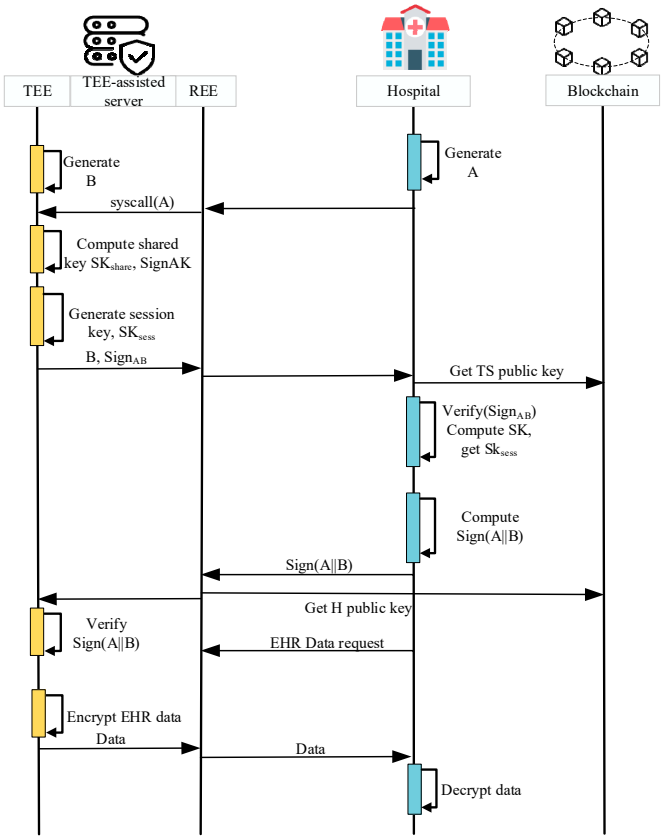

Figure 2. TEE-empowered data sharing key generation flow.

Step 5: The hospital signs A and B using its signing key P_{rivH} and sends $Sign(A||B)$ to the TEE-assisted server.

Step 6: The TEE-assisted server uses the hospital's public key obtained from the blockchain to verify $Sign(A||B)$. Once the verification is successful, a trusted channel is established between the TEE-assisted server and the hospital using the derived session key SK_{sess}. Subsequently, data are securely transmitted from the TEE-assisted server to the hospital through this trusted channel.

5. Implementation

5.1. Blockchain Platform

Hyperledger. TrustHealth is built on the permissioned platform Hyperledger Fabric. We selected Hyperledger Fabric due to its suitability for permissioned networks [49], where participants are known and trusted entities, making it ideal for medical institutions. Hyperledger Fabric offers a highly modular and configurable architecture, allowing customization of various components such as the consensus mechanism, membership services, and smart contracts [50]. Furthermore, Hyperledger Fabric is designed to handle high transaction throughput, which is essential for a system like TrustHealth that needs to process numerous transactions efficiently. The technology and modularity of Hyperledger Fabric simplify the development process by separating the core system from the application domain, allowing applications to choose transaction rules, govern access permissions, and select consensus algorithms. Additionally, being part of the Hyperledger framework, it is developed and maintained by the Linux Foundation, ensuring robust support and continuous improvement.

Consensus. Considering the low possibility of malicious nodes within medical institutions and the fact that the whole system uses mechanisms such as digital proofs and cryptographic algorithms to enhance security, we chose the Raft consensus mechanism [12]. Raft is a leader-based consensus algorithm that is simpler and more efficient compared

to more complex algorithms like PBFT. This simplicity helps in reducing the system's complexity and operational costs. Furthermore, Raft enhances system throughput by minimizing the communication overhead required for consensus, making it well-suited for environments where high performance and low latency are critical.

In TrustHealth, we utilize the Hyperledger chaincode as the smart contract. The smart contract is responsible for defining and executing the logic related to handling data hashes on the blockchain. In the context of a proposed transaction, other applications can invoke the smart contract to read and modify data on the blockchain. The primary functions of the smart contract involve querying and updating the hash of the EHR. The logical flow of the smart contract is presented in Algorithm 1. In the following, we describe the smart contract implemented to manage patient records within the TrustHealth system. The code is written in Go language, utilizing the Hyperledger Fabric SDK. This section details the primary functions of the smart contract, which include creating, querying, and updating patient records.

Algorithm 1: Logical flow for smart contracts

```
Definitions:
ehrHashes: Map of patient identifiers to their HER hashes
Operations:
Initialize(): ehrHashes ← {}
1:  function create
2:        patient ← {patientID: patientID, create_Time: create_Time, status: status,
EHR_Type: EHR_Type, EHR_Hash: EHR_Hash, Hospitals: Hospitals}
3:        patientAsBytes ← JSON.Marshal(patient)
4:        ctx.GetStub().PutState(patientID,patientAsBytes)
5:        ehrHashes[patientID] ← EHR_Hash
6:        Emit("HashUpdated("+patientID+","+EHR_Hash+")")
7:  end function
8:  function query
9:        patientAsBytes ← ctx.GetStub().GetState(patientID)
10:       if patientAsBytes = null then
11:             Exit("No patient found for " + patientID)
12:       patient ← JSON.Unmarshal(patientAsBytes)
13:       return patient
14:  end function
15:  function update
16:       patient ← QueryPatient(ctx, patientID)
17:       patient.Hash ← newEHR_Hash
18:       patient.update_Time ← update_Time
19:       patient.EHR_Type ← EHR_Type
20:       patient.Hospitals ← Hospitals
21:       patientAsBytes ← JSON.Marshal(patient)
22:        ctx.GetStub().PutState(patientID, patientAsBytes)
23:       ehrHashes[patientID] ← newEHR_Hash
24:       Emit("HashUpdated("+patientID+","+newEHR_Hash+")")
25:  end function
```

Create: This function is called to create a new patient record.

- patientID: The unique identifier for the patient.
- create_Time: The timestamp when the patient record is created.
- status: The current status of the patient.
- EHR_Type: The type of electronic health record.
- EHR_Hash: A hash value associated with the patient's EHR, used for data integrity verification.
- Hospitals: The list of hospitals associated with the patient's care.

Query: This function is called to retrieve a patient record using their unique identifier.

- patientID: The unique identifier for the patient.

Update: This function is called to update the patient's status, EHR hash, update time, EHR type, and associated hospitals.

- patientID: The unique identifier for the patient.
- update_Time: The timestamp when the patient's record is updated.
- newStatus: The new status of the patient.
- EHR_Type: The type of electronic health record.
- newEHR_Hash: The new hash value associated with the patient's EHR.
- Hospitals: The list of hospitals associated with the patient's care.

5.2. Secure Database Services

To ensure the secure operation of EHRs, we integrate SQLCipher within TEE [51]. SQLCipher is a widely used, lightweight, portable, low-memory relational database written in ANSI-C with a simple and easy-to-use API. Specifically, SQLCipher is based on SQLite [52] with added encryption and decryption features. This approach offers several advantages. Firstly, it supports AES encryption and decryption of database files, increasing the security of data stored on the disk. Secondly, it allows TEE to support the secure relational database, which enriches the application scenarios. However, it is not enough to protect data in memory, as data can be lost after a power failure, which is unacceptable for doctor and patient privacy data. Therefore, we implemented data persistence for SQLCipher running in TEE. Moreover, we extended OP-TEE to provide the same file system calls as SQLCipher. By introducing a customized set of file system calls based on secure storage, we enable SQLCipher to run as it does in REE.

Encryption/decryption interface. In SQLCipher, we implemented an encryption and decryption layer based on the Libmbedtls library. The modifications primarily encompassed the integration of encryption and decryption interfaces and the addition of specific system calls. These changes involved the adaptation and substitution of the OpenSSL library (version 1.1.1f) with the Libmbedtls library, thereby enhancing the security and encryption standards within the TrustHealth framework. Moreover, key functions such as the random, hmac, kdf, and cipher were implemented to seamlessly incorporate SQLCipher into TrustHealth. Each database is stored as an individual encrypted file (*.db), and AES-CBC encryption and decryption are performed using the Libmbedtls library.

File system calls. OP-TEE originally only provided secure storage interfaces and lacked comprehensive file operation interfaces. To ensure the consistency of file operations and the security of disk operations using SQLCipher, we have implemented a more extensive set of encrypted file interfaces based on the native secure storage design of OP-TEE. These interfaces include read, write, and open operations, which encrypt file blocks and interact with the disk through RPC requests, ensuring the security of file operations within a trusted environment. Specifically, we have reimplemented the file system API in OP-TEE, which comprises 17 system calls. These system calls encompass various aspects of file and directory operations, such as hardware-protected directory creation, directory deletion, access checks, synchronization, and the configuration of file permissions and masks. The functionality interfaces and features are detailed in Table 3. Additionally, we added the new instruction CFG_NEW_FS to the compile instruction set in our prototype, simplifying the process of enabling, compiling, and disabling new file system features. To support SQLCipher in TEE, we made some adjustments to the SQLCipher, but these modifications only need to be performed once and are not dependent on a particular OS version, ensuring their reusability. We believe that the advantages of supporting a secure database in the TEE compensate for this limitation.

Table 3. File system calls in TrustHealth.

Functions	Enforce Capability	LoC
File	Hardware protection, including read, write, and related file operations.	498
Directory	Hardware protection, including directory creation, directory deletion, and related operations.	164
Permission	Hardware protection, checking the accessibility of files, setting file permissions and masks.	103
Synchronization	Hardware protection, flushing file data from memory to disk.	20

6. Performance Evaluation

6.1. Experiment Setup

We developed TrustHealth using TEE and Hyperledger Fabric (v2.3). We integrated the secure database SQLCipher into a TrustZone-enabled development board equipped with the Kylin V10, a 3 GHz ARM 64-bit CPU, and 16 GB of RAM. The development board served as an ideal environment for running both SW and NW applications, and it provided the essential infrastructure for conducting all experiments related to TrustHealth. It is worth noting that TA was written using C language. To implement the cryptographic algorithms within TrustHealth, we leveraged the cryptographic libraries provided by the OP-TEE framework, including Libtomcrypt [47] and Libmbedtls [48]. These libraries support various cryptographic operations, such as hash functions, digital signatures, and key exchange, ensuring robust security for our system.

To comprehensively evaluate the effectiveness of TrustHealth, we define the following key performance and security metrics. These metrics ensure the system meets the necessary standards for secure and efficient eHealth data management.

1. Data Confidentiality: Ensures that all sensitive healthcare data are protected during transmission and storage, preventing unauthorized access. This metric evaluates the system's ability to keep data confidential.
2. Data Integrity: Ensures that healthcare data are not tampered with. This metric evaluates the system's ability to detect and prevent unauthorized modifications to the data.
3. Resistance to Forgery and Tampering: Measures the system's ability to prevent unauthorized entities from generating or modifying EHRs. This metric evaluates the system's defenses against forgery and tampering attempts.
4. Response Time: Measures the time taken for various operations (create, query, and update) to ensure the system remains responsive. This metric evaluates the system's performance under different levels of load and concurrency.
5. Communication Overhead: Calculates the time from sending a request to receiving a response, including connection setup and data transmission times. This metric evaluates the efficiency of data transmission in the network.
6. Secure Database Performance: Analyzes the time overhead for database operations, including CRUD operations, to ensure operational efficiency. This metric evaluates the impact of data management on overall system performance.

6.2. Performance of Blockchain

We rigorously tested the performance of three smart contracts, namely create, query, and update. To achieve this, we utilized http_load [53], a widely recognized tool for conducting load tests. Our experiment included setting the number of concurrent users to 500, and the duration of the test was set to one minute. The results indicated that http_load achieved a 100% success rate. Figure 3 shows the performance of the three smart contracts under 500 concurrent users. Our findings indicate that query functions exhibited relatively faster since they did not modify existing data. In contrast, the create and update functions modify the data, thereby requiring modifications to the underlying database of

the blockchain. Comparing the two sets of data, we can observe that the second set of data generally has higher response times across all three operations compared to the first set of data. This indicates that response time is directly proportional to the number of concurrent user visits.

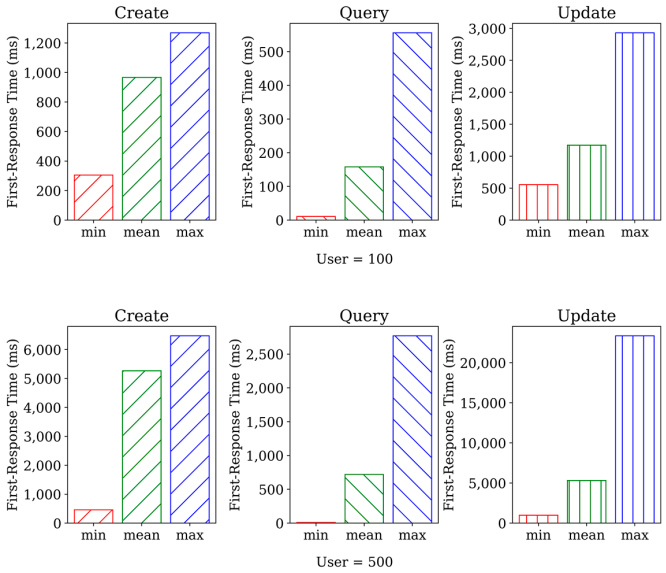

Figure 3. The results of http_load.

6.3. Communication Overhead

The communication overhead for different operations (create, query, and update) was measured to evaluate the performance of TrustHealth. Figure 4 shows that the create operation has a communication overhead of approximately 2.05 s. This overhead is primarily due to the initial setup required for creating new entries, which involves multiple steps such as generating keys, storing data securely, reaching consensus, and updating the blockchain. The query operation is significantly faster, with an average communication overhead of approximately 7 ms, indicating that querying existing data from the blockchain is an efficient process, as it primarily involves retrieving and verifying the stored information. The update operation also has a high communication overhead of around 2.04 s, similar to the create operation, because modifying existing entries involves similar steps, including updating the data, ensuring its integrity and security, and reaching a consensus to validate the changes. The higher overhead for write operations suggests potential areas for optimization. Future work will focus on improving the efficiency of these operations to reduce latency and enhance overall system performance.

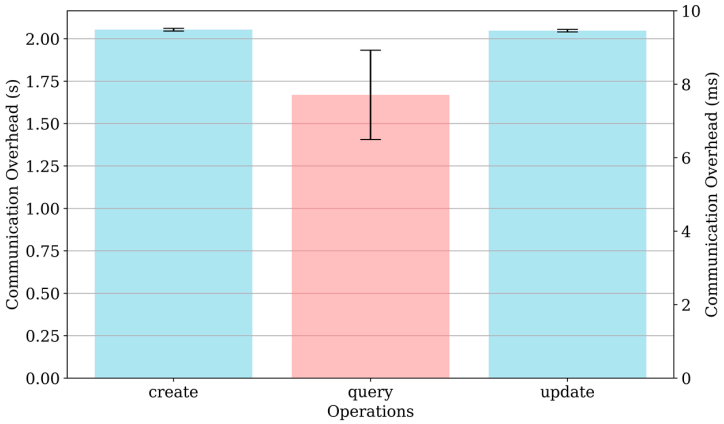

Figure 4. The communication overhead for different operations.

6.4. TrustZone Latency

In developing TA, it is crucial to consider the potential overhead resulting from frequent switching between the NW and the SW. This is especially important because the trusted application needs to be able to actively serve client applications by relying on the NW invoke mechanism. To shed light on this issue, we conducted a series of experiments and analyzed the switching time between the NW and the SW, as shown in Figure 5. To ensure accurate measurements, we configured the heap and stack of the trusted application to 2 MB and 32 KB, respectively. We recorded the elapsed time before and after the TA invocation, and our results demonstrate that the average time required to call a function in the SW is approximately 57 µs, while the time to return is around 13 µs, consistent with findings reported in prior research [54]. These results provide valuable insights into the performance of trusted applications and highlight the importance of optimizing the switching process between the NW and SW to minimize overhead and ensure optimal system performance.

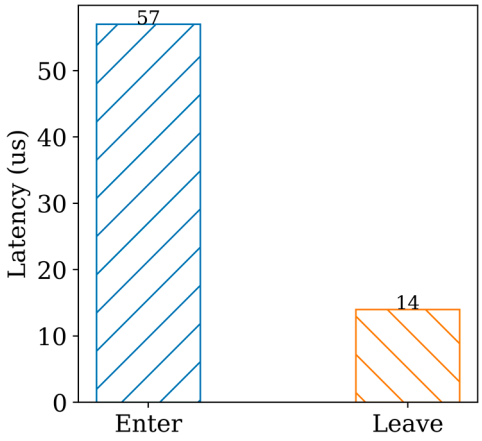

Figure 5. TrustZone latency.

6.5. SQLCipher Benchmark

The performance evaluation of TrustHealth-ported SQLCipher v4.5 was conducted using the built-in benchmark suite Speedtest1 [55]. Each benchmark in Speedtest1 evaluates an aspect of the database engine. We allocated the heap size of 800 MB for a single TA, which was sufficient to test the performance of Speedtest1. The experiments were conducted by instantiating an in-memory database and a disk database to run tests with the full dataset in the benchmark, respectively. The results of the experiments, which ran 10 times and were normalized, are presented in Figure 6. While read queries had a small impact on performance, the results of write-intensive experiments (e.g., experiments 100–120, 180, 190, 210, 290, 300, 400, and 500) showed a significant difference compared to SELECT, which had a greater impact on database performance. Overall, the experiments analyzed the correlation between the in-memory database and the disk database, with experiments 130–145, 160–170, 260, 310, 320, and 520 used to analyze the performance of database reads. It is important to note that memory limitations are not caused by TrustZone, and as long as OP-TEE addresses the limitation of a maximum of 1024 MB of memory for a single TA, memory will not be a significant factor for TrustHealth performance.

We allocated a significant amount of shared memory to enhance the communication performance between ROS and SQLCipher. Specifically, we set the shared memory configuration, CFG_SHMEM_SIZE, to 768 MB, which enabled efficient data transfer between ROS and SQLCipher. We conducted experiments with varying data sizes and found that data transfer between ROS and SQLCipher was efficient, as demonstrated in Table 4, thus guaranteeing high communication efficiency.

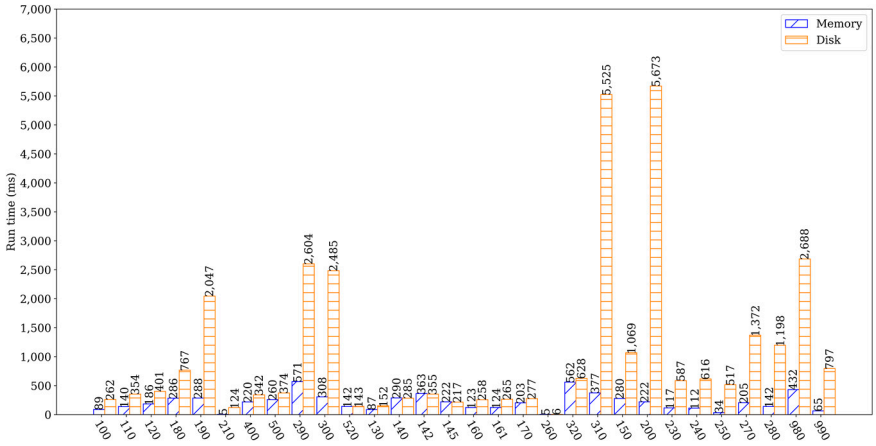

Figure 6. SQLCipher benchmark results.

Table 4. Data communication performance.

Data Size (MB)	Time (ms)
4	9
16	36
64	144
128	287
256	573
512	1143

6.6. CRUD Performance

To enhance the security of sensitive healthcare data stored in the cloud, traditional encryption methods typically employed by cloud services are insufficient to protect data during its usage. TrustHealth offers high data security but comes with tradeoffs in system usability and performance. To accurately evaluate CRUD performance, specifically INSERT, SELECT, UPDATE, and DELETE, corresponding evaluations need to be conducted. We compare the time overhead to execute CRUD commands in two scenarios: (1) CRUD operations on plaintext without any encryption, and (2) CRUD operations on ciphertext using TrustHealth. We created two tables, test1 and test2, with four columns each: id (integer type), t1 (integer type), t2 (float type), and t3 (string type). The only difference between them is that all data in test1 are encrypted at all times. This experiment is performed on both plain and encrypted disk databases. We compare the overhead for plaintext and ciphertext for four common operations. The INSERT involved inserting 10,000 records into two tables. The SELECT involved retrieving all records if t1 matched a specific value. The UPDATE involved updating t1 if its data matched a random value. Finally, the DELETE operation deleted all data in both tables. Each test was repeated 100 times, and the average time results are presented in Figure 7. The findings revealed that the INSERT and UPDATE operations took 2 s and 5.8 s, respectively, while the SELECT and DELETE operations could be completed within 100 ms.

We conducted a detailed analysis of the execution process for an SQL statement, including the INSERT operation, which involves several steps such as initialization, random number generation, cipher, and hmac check. To assess the performance impact of these operations, we used AES-256-CBC to encrypt the data during the INSERT operation. Table 5 illustrates the experiment configuration, where all the data in test1 are encrypted. The primary operations in the encryption process are random number generation, cipher, and hmac check. Normally, cipher takes 69 µs, whereas random number generation and hmac check take 167 µs. Hence, encryption overhead is not a major factor affecting database performance. Figure 8 depicts the results of each phase of the experiment. Furthermore, the reason for the large time overhead of UPDATE is that it calls random number

generation, cipher, and hmac check multiple times. Overall, from the discussion of the experiment results, it can be concluded that TrustHealth can provide CRUD services at an acceptable cost.

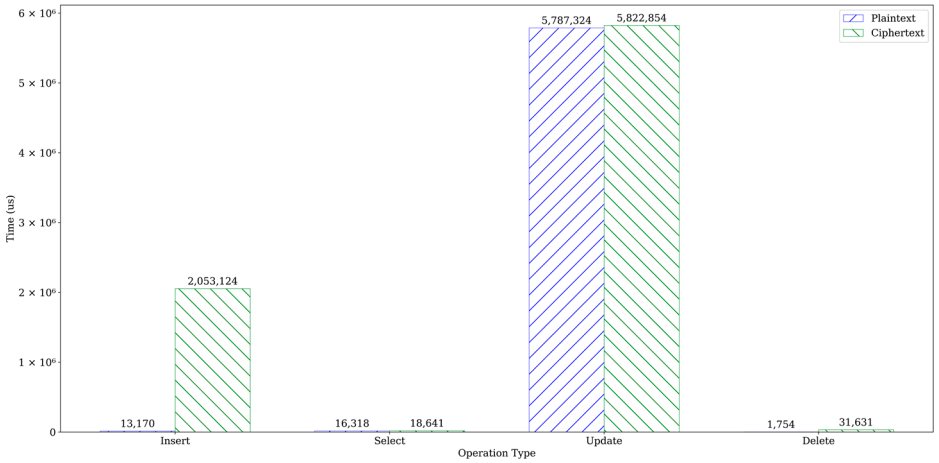

Figure 7. CRUD performance comparison.

Table 5. Experimental configuration for INSERT.

Algorithm	Query
AES-CBC	INSERT INTO test1 (id, t1, t2, t3) VALUES (?1, ?2, ?3, ?4)

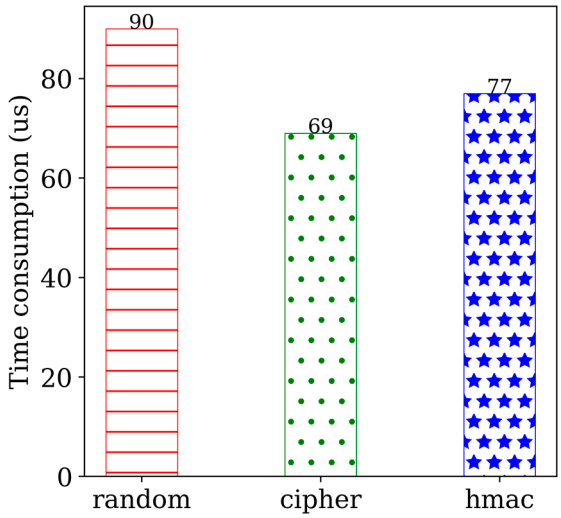

Figure 8. Performance on INSERT test.

7. Security Evaluation

7.1. Discussion

Our experiments confirmed that TrustHealth provides robust data confidentiality by encrypting all sensitive healthcare data using AES-256-CBC during transmission and storage. This ensures that unauthorized access is prevented, meeting the data confidentiality criteria. Additionally, the system effectively utilizes blockchain to store data hashes for integrity verification, and hmac checks ensure that any tampering with the data is detectable, thus maintaining data integrity. By integrating TEE and blockchain, TrustHealth offers strong resistance to forgery and tampering. The hash verification process ensures that any unauthorized attempts to generate or modify EHRs are identified and prevented. Performance metrics indicate that TrustHealth maintains acceptable response times, with

create and update operations for smart contracts averaging approximately 2 s, and query operations averaging 6.6 ms. The communication overhead is reasonable, reflecting an efficient data transmission process.

Moreover, the INSERT operation processed 10,000 encrypted data records in just 2 s, showcasing SQLCipher's ability to sustain efficient write performance alongside robust data security. The database performance analysis revealed that TrustHealth efficiently handles CRUD operations, despite the overhead introduced by encryption.

7.2. Security Analysis

Data Confidentiality. Consider a scenario in which an attacker gains access to the REE OS and attempts to tamper with the database files. However, all patient EHRs are securely encrypted and stored on the TEE-assisted server, and an attacker cannot access the TEE to obtain the encryption key. Since the encryption key is inaccessible to attackers, the encrypted data remain unintelligible, preventing access to plaintext form. Although Singh et al. [56] found that there is no difference between using TEE and traditional encryption algorithms in the process of protecting private data, TrustHealth still offers data-in-use protections by using TEE. Most importantly, TrustHealth never exposes EHR in plaintext, and decrypted data can only be displayed and processed within the TEE.

Data Integrity. In TrustHealth, data are divided into two types: on-chain data and off-chain data. The integrity of on-chain data is guaranteed by the immutability of the blockchain. Previous studies [3] have highlighted the risk of collusion between cloud servers and doctors to modify or delete EHRs. Suppose the attacker modifies the database file so that the integrity of the database is damaged. TrustHealth effectively utilizes blockchain to store data hashes for integrity verification. When a patient queries their EHR, the system compares the hash of the queried data with the pre-stored hash. This guarantees the correctness and integrity of the outsourced EHR, making any modifications detectable.

Resistance to Forgery and Tampering. TrustHealth's integration of TEE with blockchain technology offers robust protection against forgery and tampering attacks. To forge an EHR within TrustHealth, an adversary would need to alter the hash value of the EHR on the blockchain and manipulate the corresponding EHR stored within the TEE. This is an extremely challenging task as it involves tampering with both the immutable block-chain and the secure TEE, which is impractical due to the substantial cost and complexity involved. Furthermore, the encryption and decryption keys are securely stored within the TEE, safeguarding them from unauthorized access. This ensures that even if an attacker gains access to the external system, they cannot retrieve the keys necessary to decrypt or modify the EHR data. This robust security mechanism guarantees that the integrity and authenticity of EHRs are maintained, preventing unauthorized modifications.

7.3. Limitations

TrustHealth demonstrates robust security and performance. However, there are areas for improvement, particularly in its resistance to physical attacks. While TEEs provide strong protection for data in use, static data remain vulnerable to physical attacks. In some cases, the data themselves can be falsified and rendered unreadable due to encryption; an attacker could significantly restrict availability in this way. Future work will focus on enhancing the encryption and access control mechanisms for static data, mitigating the risks associated with physical attacks.

8. Conclusions and Future Work

While healthcare data sharing enhances diagnostic convenience, the paramount concern of data security cannot be ignored. In this paper, we propose TrustHealth, a solution designed to enhance the data security and privacy of EHRs. TrustHealth leverages TEE to design a secure database, and it incorporates a secure session key generation protocol to establish secure communication channels between healthcare providers and the TEE. A

consortium blockchain is employed to securely store EHR hashes on-chain, providing immutability and transparency. Comprehensive experiments evaluate the throughput of the blockchain system, the performance of the secure database, and the overhead of encryption and decryption. Our performance evaluation shows that TrustHealth can securely process massive encrypted data flows at a rate of 5000 records per second, demonstrating that TrustHealth can ensure security and privacy without incurring significant overhead. In the future, we will focus on improving data availability to ensure that encrypted data remain accessible under adverse conditions. Additionally, we will conduct extensive real-world testing to validate TrustHealth's practicality and effectiveness.

Author Contributions: Conceptualization, J.L. and X.L.; methodology, J.L.; software, J.L.; writing—original draft, J.L.; validation, X.L.; writing—review and editing, J.L.; supervision, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part by the National Key R&D Program of China (2021YFB2700600); in part by the Finance Science and Technology Project of Hainan Province (ZDKJ2020009); in part by the National Natural Science Foundation of China (62163011); in part by the Research Startup Fund of Hainan University under Grant KYQD(ZR)-21071; and in part by the Education Department of Hainan Province (Hnky2024-58).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Authors Jun Li and Hong Lei are employed by the company SSC Holding Company Ltd. The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. De Aguiar, E.J.; Faiçal, B.S.; Krishnamachari, B.; Ueyama, J. A survey of blockchain-based strategies for healthcare. *ACM Comput. Surv.* **2020**, *53*, 1–27. [CrossRef]
2. Bian, J.; Yang, S.; Xiong, H.; Wang, L.; Fu, Y.; Sun, Z.; Guo, Z. CRLEDD: Regularized causalities learning for early detection of diseases using electronic health record (EHR) data. *IEEE Trans. Emerg. Top. Comput. Intell.* **2021**, *5*, 541–553. [CrossRef]
3. Cao, S.; Zhang, G.; Liu, P.; Zhang, X.; Neri, F. Cloud-assisted secure eHealth systems for tamper-proofing EHR via blockchain. *Inf. Sci.* **2019**, *485*, 427–440. [CrossRef]
4. Yang, J.J.; Li, J.Q.; Niu, Y. A hybrid solution for privacy preserving medical data sharing in the cloud environment. *Future Gener. Comput. Syst.* **2015**, *43–44*, 74–86. [CrossRef]
5. Zhang, X.; Wang, H.; Xu, C. Identity-based key-exposure resilient cloud storage public auditing scheme from lattices. *Inf. Sci.* **2019**, *472*, 223–234. [CrossRef]
6. Campanile, L.; Iacono, M.; Marulli, F.; Mastroianni, M. Designing a GDPR compliant blockchain-based IoV distributed information tracking system. *Inf. Process. Manag.* **2021**, *58*, 102511. [CrossRef]
7. Cao, S.; Zhang, X.; Xu, R. Toward secure storage in cloud-based eHealth systems: A blockchain-assisted approach. *IEEE Netw.* **2020**, *34*, 64–70. [CrossRef]
8. Armknecht, F.; Bohli, J.M.; Karame, G.O.; Liu, Z.; Reuter, C.A. Outsourced proofs of retrievability. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–4 November 2014; pp. 831–843.
9. Liu, G.; Yan, Z.; Feng, W.; Jing, X.; Chen, Y.; Atiquzzaman, M. SeDID: An SGX-enabled decentralized intrusion detection framework for network trust evaluation. *Inf. Fusion* **2021**, *70*, 100–114. [CrossRef]
10. Zhao, Q.; Chen, S.; Liu, Z.; Baker, T.; Zhang, Y. Blockchain-based privacy-preserving remote data integrity checking scheme for IoT information systems. *Inf. Process. Manag.* **2020**, *57*, 102355. [CrossRef]
11. Li, M.; Fang, Y.; Tang, Z.; Onuorah, C.; Xia, J.; Del Ser, J.; Walsh, S.; Yang, G. Explainable COVID-19 infections identification and delineation using calibrated pseudo labels. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *7*, 26–35. [CrossRef]
12. Xu, J.; Wang, C.; Jia, X. A survey of blockchain consensus protocols. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
13. Trusted Execution Environment. Available online: https://en.wikipedia.org/wiki/Trusted_execution_environment (accessed on 15 May 2024).
14. Sengupta, J.; Ruj, S.; Bit, S.D. A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT. *J. Netw. Comput. Appl.* **2020**, *149*, 102481. [CrossRef]
15. Berdik, D.; Otoum, S.; Schmidt, N.; Porter, D.; Jararweh, Y. A survey on blockchain for information systems management and security. *Inf. Process. Manag.* **2021**, *58*, 102397. [CrossRef]
16. Zou, R.; Lv, X.; Zhao, J. SPChain: Blockchain-based medical data sharing and privacy-preserving eHealth system. *Inf. Process. Manag.* **2021**, *58*, 102604. [CrossRef]

17. Bernabé-Rodríguez, J.; Garreta, A.; Lage, O. A decentralized private data marketplace using blockchain and secure multi-party computation. *ACM Trans. Priv. Secur.* **2024**, *27*, 1–29. [CrossRef]
18. Mandarino, V.; Pappalardo, G.; Tramontana, E. A blockchain-based electronic health record (EHR) system for edge computing enhancing security and cost efficiency. *Computers* **2024**, *13*, 132. [CrossRef]
19. OP-TEE. Available online: https://op-tee.org (accessed on 16 May 2024).
20. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. *SIAM J. Comput.* **2003**, *32*, 586–615. [CrossRef]
21. Du, X.; Guizani, M.; Xiao, Y.; Chen, H. Transactions papers a routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 1223–1229. [CrossRef]
22. Jiang, L.; Li, T.; Li, X.; Atiquzzaman, M.; Ahmad, M.; Wang, X. Anonymous communication via anonymous identity-based encryption and its application in IoT. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 6809796. [CrossRef]
23. Li, J.; Chen, X.; Li, M.; Li, J.; Lee, P.P.; Lou, W. Secure deduplication with efficient and reliable convergent key management. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 1615–1625. [CrossRef]
24. Boneh, D.; Lynn, B.; Shacham, H. Short signatures from the Weil pairing. *J. Cryptol.* **2004**, *17*, 297–319. [CrossRef]
25. Li, T.; Chen, W.; Tang, Y.; Yan, H. A homomorphic network coding signature scheme for multiple sources and its application in IoT. *Secur. Commun. Netw.* **2018**, *2018*, 9641273. [CrossRef]
26. Bellare, M.; Kilian, J.; Rogaway, P. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.* **2000**, *61*, 362–399. [CrossRef]
27. Zhou, T.; Shen, J.; He, D.; Vijayakumar, P.; Kumar, N. Human-in-the-Loop-Aided privacy-preserving scheme for smart healthcare. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, *6*, 6–15. [CrossRef]
28. Li, J.; Wu, J.; Jiang, G.; Srikanthan, T. Blockchain-based public auditing for big data in cloud storage. *Inf. Process. Manag.* **2020**, *57*, 102382. [CrossRef]
29. Zhang, Y.; Xu, C.; Li, H.; Yang, K.; Zhou, J.; Lin, X. HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4101–4112. [CrossRef]
30. Xia, Q.; Sifah, E.B.; Asamoah, K.O.; Gao, J.; Du, X.; Guizani, M. MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access* **2017**, *5*, 14757–14767. [CrossRef]
31. Fan, K.; Wang, S.; Ren, Y.; Li, H.; Yang, Y. Medblock: Efficient and secure medical data sharing via blockchain. *J. Med. Syst.* **2018**, *42*, 136. [CrossRef]
32. Huang, H.; Zhu, P.; Xiao, F.; Sun, X.; Huang, Q. A blockchain-based scheme for privacy-preserving and secure sharing of medical data. *Comput. Secur.* **2020**, *99*, 102010. [CrossRef]
33. Azaria, A.; Ekblaw, A.; Vieira, T.; Lippman, A. MedRec: Using blockchain for medical data access and permission management. In Proceedings of the 2016 2nd International Conference on Open and Big Data (OBD), Vienna, Austria, 22–24 August 2016; pp. 25–30.
34. Jiang, W.; Li, E.; Zhou, W.; Yang, Y.; Luo, T. IoT access control model based on blockchain and trusted execution environment. *Processes* **2023**, *11*, 723. [CrossRef]
35. Mathur, S.; Kalla, A.; Gür, G.; Bohra, M.K.; Liyanage, M. A survey on role of blockchain for IoT: Applications and technical aspects. *Comput. Netw.* **2023**, *227*, 109726. [CrossRef]
36. Lin, S.Y.; Zhang, L.; Li, J.; Ji, L.L.; Sun, Y. A survey of application research based on blockchain smart contract. *Wirel. Netw.* **2022**, *28*, 635–690. [CrossRef]
37. Muñoz, A.; Rios, R.; Román, R.; López, J. A survey on the (in) security of trusted execution environments. *Comput. Secur.* **2023**, *129*, 103180. [CrossRef]
38. OP-TEE Documentation. Available online: https://optee.readthedocs.io/en/latest/ (accessed on 16 May 2024).
39. Secure Storage. Available online: https://optee.readthedocs.io/en/latest/architecture/secure_storage.html (accessed on 16 May 2024).
40. Azzedin, F.; Ghaleb, M. Internet-of-things and information fusion: Trust perspective survey. *Sensors* **2019**, *19*, 1929. [CrossRef] [PubMed]
41. Fortino, G.; Fotia, L.; Messina, F.; Rosaci, D.; Sarné, G.M.L. Trust and reputation in the internet of things: State-of-the-art and research challenges. *IEEE Access* **2020**, *8*, 60117–60125. [CrossRef]
42. Martin, A.; Lian, C.; Gregor, F.; Krahn, R.; Schiavoni, V.; Felber, P.; Fetzer, C. ADAM-CS: Advanced asynchronous monotonic counter service. In Proceedings of the 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Taipei, Taiwan, 21–24 June 2021; pp. 426–437.
43. Ryan, K. Hardware-backed heist: Extracting ECDSA keys from Qualcomm's TrustZone. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 181–194.
44. Qiu, P.; Wang, D.; Lyu, Y.; Qu, G. Voltjockey: Breaching TrustZone by software-controlled voltage manipulation over multi-core frequencies. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 195–209.
45. Zhang, N.; Sun, K.; Shands, D.; Lou, W.; Hou, Y.T. TruSense: Information leakage from TrustZone. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 1097–1105.
46. Hyperledger Fabric SDK-Go. Available online: https://github.com/hyperledger/fabric-sdk-go (accessed on 16 May 2024).

47. Libtomcrypt. Available online: https://optee.readthedocs.io/en/latest/architecture/crypto.html#libtomcrypt (accessed on 16 May 2024).
48. Libmbedtls. Available online: https://optee.readthedocs.io/en/latest/architecture/libraries.html#libmbedtls (accessed on 16 May 2024).
49. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Yellick, J. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the 13 EuroSys Conference, New York, NY, USA, 23–26 April 2018; pp. 1–15.
50. Battisti, J.H.F.; Batista, V.E.; Koslovski, G.P.; Pillon, M.A.; Miers, C.C.; Marques, M.A.; Simplício, M.; Kreutz, D. Performance analysis of the Raft consensus algorithm on Hyperledger Fabric and Ethereum on cloud. In Proceedings of the 2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Naples, Italy, 4–6 December 2023; pp. 155–160.
51. SQLCipher. Available online: https://github.com/sqlcipher/sqlcipher/tree/master (accessed on 15 May 2024).
52. SQLite. Available online: https://en.wikipedia.org/wiki/SQLite (accessed on 16 May 2024).
53. Http_load. Available online: https://acme.com/software/http_load/ (accessed on 15 May 2024).
54. Ménétrey, J.; Pasin, M.; Felber, P.; Schiavoni, V. Watz: A trusted WebAssembly runtime environment with remote attestation for TrustZone. In Proceedings of the 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), Bologna, Italy, 10–13 July 2022; pp. 1177–1189.
55. Speedtest1. Available online: https://github.com/sqlcipher/sqlcipher/blob/master/test/speedtest1.c (accessed on 16 May 2024).
56. Singh, J.; Cobbe, J.; Quoc, D.L.; Tarkhani, Z. Enclaves in the clouds: Legal considerations and broader implications. *Commun. ACM* **2021**, *64*, 78–114. [CrossRef]