information

Article

Traversable Ledger for Responsible Data Sharing and Access Control in Health Research †

Sunanda Bose * and Dusica Marijan

Simula Research Laboratory, 0164 Oslo, Norway; dusica@simula.no

* Correspondence: sunanda@simula.no

† This article is a revised and expanded version of a paper entitled [Secure Traversable Event logging for Responsible Identification of Vertically Partitioned Health Data], which was presented at [TrustCom, Exeter, UK, 1–3 November 2023].

Abstract: Healthcare institutions and health registries often store patients' health data. In order to ensure privacy, sensitive medical information is stored separately from the identifying information of the patient. Generally, institutions anonymize medical information while sharing it for external use. However, internal users may also use it for identifying inaccuracies or missing information. Even though internal users may be legally permitted to access sensitive medical information, such access may lead to the identification of the patient, which can be vulnerable to patient privacy. Ensuring the accountability and responsibility of the internal users may lead to tractability in case of adversarial access with malicious intentions. Therefore, a secure system must be developed for the storage and retrieval of health data. To this end, in this paper, we propose a ledger-based system that cryptographically ensures that all access to health data must be logged into a ledger. Nevertheless, the ledger entries must be protected against adversarial access, too. At the same time, the ledger must be traversable by the patients as well as internal users. To address these needs, we propose techniques for the construction of a ledger to permit both internal users and patients to securely traverse and view only the entries to which they are linked.

Keywords: privacy; healthcare; health registry; health data; responsible de-identification; traversable ledger; secure private information retrieval

check for updates

Citation: Bose, S.; Marijan, D. Traversable Ledger for Responsible Data Sharing and Access Control in Health Research. *Information* **2024**, *15*, 815. https://doi.org/10.3390/info15120815

Academic Editor: Evaggelos Karvounis

Received: 11 October 2024

Revised: 3 December 2024

Accepted: 9 December 2024

Published: 18 December 2024

1. Introduction

Health Data (HD), typically seen as private information belonging to patients, is often held in the possession of a healthcare institution. These data can potentially be utilized for medical research involving researchers both within and outside the institution's jurisdiction. Generally, institutions anonymize the data before it is shared with researchers who are not within their jurisdiction. However, there can also be internal researchers who operate under the institution's jurisdiction. They may employ the data to identify any incompleteness and inconsistencies present in the medical record [1]. Incompleteness in the data could suggest that some essential documents (for example, laboratory tests or doctor's reports) that should be present in the repository are yet to be submitted. If any inconsistency is detected by the internal researchers, it could lead to the decision to repeat certain clinical tests or could even result in the identification of a misdiagnosis.

Registry-based healthcare is not only limited to healthcare institutions like hospitals. National healthcare systems of several countries also maintain documentation of the population [1–3]. The first population-based cancer registries were created in Germany in 1929 [3]. In different countries, several registries of health registries are maintained [1–3]. These registries comprise different types of health data associated with their personal identity numbers [2]. This way, the health data of approximately 27 million people are documented in several registries, including birth registries, death registries, patient registries, prescription registries, cancer registries, etc. [2]. Each of these registries has different purposes and sources of data collection. The cancer registries usually collect data from various sources, like hospitals, clinicians, dentists, laboratories, radiotherapy data, and death certificates [4]. Both national and city-wide registries exist to document and monitor small groups of cancer patients located in the same city. The Geneva Cancer Registry, created in 1970, documents the cancer cases from a small population of Geneva [3]. The first cancer registry in India was established in 1963 in Mumbai as a unit of the Indian Cancer Society [1]. In India, hospital-based registries are maintained along with population-based ones [1]. In [1,5], two methods of data collection are suggested. In the passive method, the institutions send information to the cancer registries. In the active method, the staff from the cancer registry collects the data from these institutions. A team of registry personnel working in the jurisdiction ensures the quality of the data by checking duplicate entries and validating the consistency of records. Such teams are typically led by a medically qualified supervisor (often referred to as principal investigator [1]) who has a background in epidemiology and/or public health.

It is common for healthcare registries to maintain a group of internal experts who regularly analyze the data stored in their custody [1]. The medical records of a patient may contain sensitive information that is associated with the identifying information of that patient. However, as these data contain highly sensitive information, they must be protected against adversarial access even when they are being accessed by internal researchers. To minimize risk, the identifying part of HD is stored separately from the medical information. However, as a patient can have multiple medical records, the many-to-one relationship between medical information and the identifying information must be manifested by some techniques that securely link the identifying information with the sensitive information. A foreign key-based approach may provide fast retrieval, but that does not offer a solution for securing the retrieval operation. In our case, the only legitimate users of these data are the internal experts, who may identify the patient associated with any given medical record. It may also be necessary to retrieve all medical records belonging to a patient. Hence, we summarize two legitimate data access scenarios: (1) A custodian identifying the patient associated with a record; and (2) A custodian fetching all records of a patient.

In both scenarios, the custodian performing these operations is gaining significant private information about the patient. Although the legal framework allows the custodian to access this information, it is crucial to ensure that the information obtained is not used for harmful purposes. However, the incidents of such information being exploited for malicious purposes may only occur in the future after the information has been accessed. In the case of such future events, the events of information gain can be correlated only if those events are logged. Such an approach can promote the legitimacy of information gain by ensuring the responsibility of the custodians. Although there have been research works addressing the security and privacy concerns of private information storage and retrieval, which is mentioned in Section 2, these works do not address the problem of responsible identification of de-identified sensitive data.

Therefore, in this paper, we expand our earlier work [6] and propose a secure system of storage and retrieval for sensitive medical information that can be accessed by custodians with sufficient credentials. As such access can lead to information gain about the patients, the events of such access are documented in an immutable ledger, which can be securely traversed by the custodians and the patients with appropriate credentials. However, in order to design such a solution, we must overcome some technical challenges. The ledger must be protected from adversarial access to ensure the privacy of the custodians as well as the patients. Simultaneously, the legitimate users (custodians and the patients) should be permitted to traverse through the ledger and analyze the events of information gain that relate to them. Moreover, the supervisor(s) (often known as the principal investigator [1]) may need to access the ledger to correlate the events with some malicious incident and verify its legitimacy.

Hence, in this paper, we propose an approach for enabling secure linkage between de-identified medical data and identifying information through a ledger-based system that custodians and patients can traverse securely, given proper credentials. The proposed approach ensures that sensitive data can be accessed responsibly without exposing it to risks of unauthorized access or misuse. It integrates the concept of "custodianship" where designated individuals have the responsibility and authority to manage data access, which aligns with legal and ethical standards. The proposed approach offers a framework that could be adapted beyond healthcare to other domains where sensitive data requires stringent access controls and accountability. Although we have focused on the use case of securing internal researchers' (data managers) access to health data, healthcare registries often deal with a similar problem while sharing health data with external researchers. The proposed framework may be extended to ensure privacy and responsibility for health data sharing and research.

The paper is organized as follows: In the next Section, we present a summary of related problems encountered in the scientific literature and the solutions proposed by the authors. In Section 3, the scientific problems that must be solved to address these challenges are discussed. Our proposed solution to address the problem is presented in Section 4. In Section 5, we present an evaluation of the proposed approach. First, we theoretically evaluate the security aspects of our proposed solution in Section 5.1. Then, we present performance analysis by performing experiments on the implementation of this work in Section 5.2. The limitations of our proposed framework are presented in Section 6.1. Finally, Section 7 concludes the paper.

2. Related Work

Studying the confidentiality of HD has been an active research field [7]. To ensure the confidentiality of HD, authors in [8] implemented an AT&T-based scheme for access control of medical records. The proposed scheme uses XACML for defining access policies. Various components are involved in ensuring the mechanism. When storing, HD is encrypted using symmetric encryption. In [9], the authors describe several access control mechanisms (RBAC, MAC, DAC, and PBAC) and their applicability for ensuring the privacy of the HD. Discretionary Access Control (DAC) specifies per user per object-based granular permissions, which can be materialized using Access Control List (ACL) and Capability List (CL). ACL takes an object-centric approach where an object is associated with a set of users with different permissions. CL, on the other hand, takes a user-centric approach where objects are associated with users with different permissions. Mandatory Access Control (MAC) assigns a security level to each of the objects and a clearance level to each of the users. Role-Based Access Control (RBAC), on the other hand, does not associate objects with individuals. Instead, it assigns roles to each of the users, and access to objects is governed by the role. However, restricting access to the documents is not our only objective. We want to make the user responsible for accessing the document. Moreover, encrypted documents are difficult to search for or analyze. We only need de-identified data that can be used for knowledge discovery without directly revealing the patient's identity.

The authors in [10] mention three levels of the confidentiality of medical information that allow the owner to define confidentiality of his/her own personal health records. A restricted level of information can be accessed by the emergency staff only if k out of a predefined n trusted users grant permission. An exclusive level of information can never be accessed by the emergency staff. The encrypted medical records are stored on a server. Encryption is performed using (k,n)-threshold cryptosystem. In [11], a different approach is proposed while using the (k,n)-threshold cryptosystem. The medical records are encrypted using the RSA algorithm. The private key, which is required to decrypt the records, is shared using (k,n)-threshold cryptosystem. Instead of giving these shares to human entities, they are stored on the server. Each of these shares corresponds to different context conditions, such as the doctor's identity, role, location, duty time, patient location, status, etc. In [12] (k,n) threshold cryptosystem is used to securely store patients' healthcare records. However, the authors identify several problems while applying the original secret-sharing scheme proposed by Shamir et al. [13] in the healthcare problems. The participants must reveal their share of the secret in order to reconstruct the secret key. The authors in [12] propose a novel cryptographic scheme in which the participants keep their secret shares, but instead of sharing them directly, they share a transformed value from which the original secret share cannot be retrieved. Cryptographic access schemes like IBE and CP-ABE are used in [10,14–16]. In [15], medical data are first encrypted by the sender using the symmetric encryption algorithm AES. The secret key is encrypted using IBE and shares the encrypted document along with the encrypted key. In [14], a hierarchical access scheme is proposed, where the Public Health Office serves as the public key generator (PKG) at the highest level, and the hospitals and clinics are at the lower level. The storage servers located at hospitals and clinics store the medical records of their patients only. The public storage server is responsible for storing the referral medical records. In [17], IBE is used along with a Markle Hash Tree to ensure the deletion of HD. Although some of the techniques used in our work may be considered to be similar to Samir's Secret-Sharing, our intention is to allow one custodian to access the patients' records independently without any cooperation from other custodians. Therefore, a threshold-based cryptosystem may not be directly applied to address our primary problem. Moreover, we want the health data to be de-identified but not encrypted. We want the identification of the de-identified data to be a secure operation that can only be performed by a legitimate user, and the user must be held responsible for performing that operation.

Vertical partitioning of data is a popular technique of de-identifying data, which is often used along with anonymization. In [18], HD is partitioned into three tables. One contains the original medical information without the identifying attributes. The other two tables contain anonymized quasi-identifiable (The attributes that can reveal important information about the identity of the patient when correlated with publicly available information.) attributes and encrypted ciphertexts of identifying and quasi-identifiable attributes. Different healthcare institutions may maintain records as vertically partitioned data [19], which may be mined for scientific or statistical purposes. Data anonymization techniques are often used to protect medical data from being re-identified. Such techniques include k-anonymity [18,20], l-diversity [21], t-closeness [22] etc. are often used by healthcare registries while exchanging HD with external research institutions. These techniques protect the data from being re-identified when correlated with publicly available information. However, it processes the original data and generalizes the values of attributes, which reduces the amount of information [23]. However, in our problem [6], the internal users are the only legitimate entities who are permitted to access the data. Also, the quality of the data cannot be compromised because the data must be reviewed by internal researchers for correctness and completeness.

Blockchain-based techniques are often used for HD-related transactions between [24,25]. In [25], all transactions are performed using smart contracts that provide two functions, store and get, and all data are stored in the blockchain as key-value pairs. In [24], the patients may delegate hospitals to encrypt their medical records and store them on semi-trusted cloud servers. The researchers consume the medical data from patients if the requirements are met. The data requirements are published via smart contracts. Patients who believe that their records meet the published requirements present zero-knowledge proof to the smart contract. Once qualified, the semi-honest cloud server transforms the encrypted medical data into an intermediate ciphertext that can be decrypted by the researcher. However, in our case, the patients are not actively participating in the process. Instead, the patients remain passive while the events of their records being accessed are documented for later viewing.

The central challenge our work addresses transcends mere access control or sharing of medical information. We aim to ensure that even if the medical information of a patient is accessed, the identifying information remains confidential, and vice versa, while enforcing accountability for data access. Hence, access control mechanisms are not sufficient. To address risks of re-identification of the patient from the medical information, techniques such as anonymization have been employed on vertically partitioned data. However, while these methods can mitigate direct identification risks, they often degrade the quality and completeness of the data, which may be problematic for internal researchers in critical health registries. For example, anonymization can strip data of its specificity, which is essential for detailed medical research and analysis. The (k,n)-threshold cryptosystem and other cryptographical schemes that have been proposed in the literature focus on secure sharing of health data. Since our scenario assumes health data are already shared, our goal is to enforce responsible access to these data. We aim to enforce the responsibility of accessing it. Given these gaps, our approach introduces a novel framework designed specifically for the intricate balance required in health data systems, where data must remain usable for research without compromising patient privacy. Our solution integrates enhanced accountability measures without the drawbacks of data degradation associated with standard anonymization techniques and extends beyond traditional cryptographic sharing controls.

In the next section, we define our problem along with the formulation of the technical requirements that need to be satisfied by a feasible solution. The rest of the paper is organized as follows. In Section 4, we explain the proposed solution. In Sections 5.1 and 5.2, we present an adversarial and experimental evaluation of our proposed work. In Section 6.1, we mention the limitations of our work. Finally, Section 7 concludes the paper.

3. Problem Definition

For privacy-related concerns, the database is often vertically partitioned, where personal information is separated from sensitive medical information [18]. In our proposed system, we assume that a medical record is de-identified and the sensitive information is stored separately from the identifying information. Only the permitted users can identify the patient associated with that de-identified record and can also find all medical records associated with the patient. We may refer to this action as record identification. However, there are two constraints applied to the identification operations.

1. Even if some adversary obtains control of the secrets of the storage server, it will not be possible to perform any of these identification actions without the participation of the permitted users.
2. In order to ensure the responsibility of the identifier, an entry must be created in the immutable ledger whenever each of these actions is performed.

It is important to note that by de-identification, we mean that the identifying information has been stripped and stored separately from the sensitive medical information. This work does not focus on protecting health data against re-identification attacks. It is technically possible to re-identify the de-identified data using sophisticated statistical methods. Many anonymizing techniques can be applied in order to protect that. However, it is not the focus of our work to propose solutions for re-identification techniques. Instead, we focus on ensuring the security and privacy of health data by enforcing accountability while acting on identification.

We refer to these actions of record identification as an *Access Event*. An access event is participated by two users. The permitted user that initiates the identification process is considered *active* because this user is actively communicating in order to identify the record for further analysis. The patient whose record is being accessed may be unaware of such an event before it happens. The decision that the records associated with that patient must be accessed may not be taken in the active participation of that patient. Hence, we consider the patient as a *passive* user. We need an immutable public ledger that documents the access events between these two users. Both of these users should be allowed to browse through the events logged into the immutable ledger securely. We refer to these documented events as blocks. The event participation information must not be disclosed to anyone other than these active and passive users and the supervisors who have sufficient credentials to access any random block and view participation information. The active user is allowed to navigate to the next and the previous blocks in which the same user was active. Similarly, the passive user is allowed to navigate to the next and the previous blocks in which the same user was passive. However, no users should be allowed to navigate to blocks associated with a different user. We formulate this problem as follows (the symbols used in the formulation and throughout the paper are shown in Table 1):

Table 1. Symbol Table.

Symbol	Usage
p, q, g	Modulus, Subgroup order and generator of the group.
$\pi_x, y_x = g^{\pi_x}$	Private and Public key of user A_x
$Y = \bigcup_{x \in X} y_x$	Set of public keys of all users X.
ξ, ζ	Symmetric Encryption, Decryption Algorithm. e.g., AES
H, H_2	Cryptographic Hash functions e.g., SHA512
$x \in_{\mathbb{R}} X$	x is a random integer from set X
$\frac{a}{b}$	ab' where b' is the multiplicative inverse of b in $\mathcal{Z}_p$, such that $bb' \equiv 1\ (mod\ p)$
a^{-1}	Multiplicative inverse of a in $\mathcal{Z}_{(p-1)}$, such that $aa^{-1} \equiv 1\ (mod\ (p-1))$
$f(x) \to y$	y is deterministically computable using function f and x.
$\tau_x^{(k)}$	k^{th} block in which user A_x was active
$\tau_{\hat{x}}^{(k)}$	k^{th} block in which user A_x was passive
$\tau^{(r)}$	r^{th} block in the ledger, where the information regarding the involvement of any user is either irrelevant or unknown.

3.1. *Storage Requirements*

For the purpose of securing record identification, we associate each side of the partitioned record with a ciphertext that we refer to as an *anchor*. In Figure 1, the identifying anchor is shown as a circle inside the red bars on the left, while the sensitive information anchor is shown as a circle inside the yellow bars. Formally, given a complete record D_i, the set of personal information dp_i and the set of medical information dm_i are annotated with different numbers dp_i^*, dm_i^*, respectively. However, $\exists\ \overrightarrow{f}(dp_i^*, x) \to dm_i^*$, $\overleftarrow{f}(dm_i^*, x) \to dp_i^*$ can relate both of these anchors, where x is the secret that can be obtained securely by the cooperation of permitted users only. For fast retrieval, the records are indexed with dp_i^*, dm_i^* .

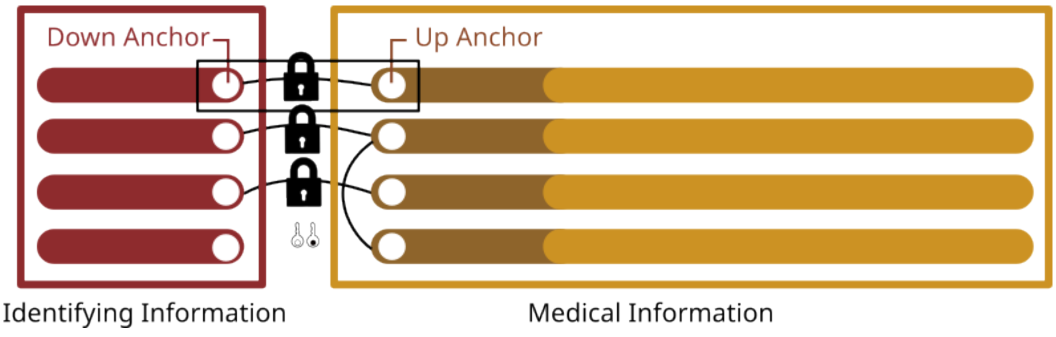
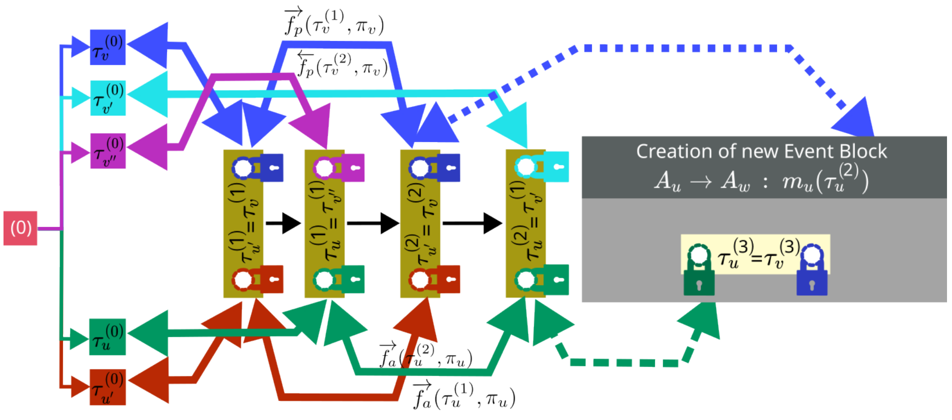


Figure 1. Secure De-Identification.

3.2. *Traversability Requirements*

Given two ordered entries B_m, B_n in which user A_i is active $\exists\ \overrightarrow{f_a}, \overleftarrow{f_a}$ such that $\overrightarrow{f_a}(B_m, \pi_i) \to B_n$ and $\overleftarrow{f_a}(B_n, \pi_i) \to B_m$ where π_i is a secret information that only A_i has access to. Similarly, if user A_i is passive and B_m, B_n are two ordered entries in which it was passive, then $\exists\ \overrightarrow{f_p}, \overleftarrow{f_p}$ such that $\overrightarrow{f_p}(B_m, \pi_i) \to B_n$ and $\overleftarrow{f_p}(B_n, \pi_i) \to B_m$. We call $\overrightarrow{f_a}, \overleftarrow{f_a}, \overrightarrow{f_p}, \overleftarrow{f_p}$ as traversal functions. In order to make the traversal secure $\nexists \overrightarrow{f_a'}(B_m, x) \to B_n$ such that $x \neq \pi_i$, and same applies for the other traversal functions. Also, there $\nexists\ F_a(B_m, B_n) \to [0,1], F_p(B_m, B_n) \to [0,1]$ that deterministically produces a binary output denoting the given two entries are related to the same active or passive user, respectively. In that case, an adversary can apply that function of all pairs or entries to partition all entries belonging to the same user. We refer to each of these entries as a *block*.

In Figure 2, we show 4 blocks (shown in yellow rectangles), each referring to an access event. In the first block from the top, user $A_{u'}$ is active while the user A_v is passive, and it is the first access event associated with both of these users. Similarly, in the fourth block, the user A_u is active while the user $A_{v'}$ is passive. The user A_u can reach this block using the function $\overrightarrow{f_a}$ and its private credentials, which is reachable from $\tau_u^{(0)}$. Although it is the fourth block in the order of time, it is the second block that A_u can jump into through active traversal. Similarly, the user $A_{v'}$ can jump into this block by traversing only once from $\tau_{v'}^{(0)}$. We label these blocks from these users' perspectives. Hence, the same block is referred to as $\tau_u^{(2)}$ and $\tau_{v'}^{(1)}$ by users A_u and $A_{v'}$. Genesis blocks are shown on the top, which are the first blocks (0^{th}) associated with each user.


$\overrightarrow{f}_p(\tau_v^{(1)}, \pi_v)$
$\overleftarrow{f}_p(\tau_v^{(2)}, \pi_v)$
$\tau_v^{(0)}$
$\tau_{v'}^{(0)}$
$\tau_{v''}^{(0)}$
(0)
$\tau_u^{(0)}$
$\tau_{u'}^{(0)}$
$\tau_{u'}^{(1)} = \tau_v^{(1)}$
$\tau_u^{(1)} = \tau_{v''}^{(1)}$
$\tau_{u'}^{(2)} = \tau_v^{(2)}$
$\tau_u^{(2)} = \tau_{v'}^{(1)}$
Creation of new Event Block
$A_u \rightarrow A_w \ : \ m_u(\tau_u^{(2)})$
$\tau_u^{(3)}=\tau_v^{(3)}$
$\overrightarrow{f}_a(\tau_u^{(2)}, \pi_u)$
$\overrightarrow{f}_a(\tau_u^{(1)}, \pi_u)$
Figure 2. Traversable Ledger.

Each user is associated with a dedicated chain of events, which has a traversable total order. However, a block in a user-specific chain may overlap with some other user's chain. As these chains are sets of blocks, the ledger proposed in our work can be described as the union of totally ordered sets. All these totally ordered sets start with a genesis block that does not have a mutual order in the context of the users' secret, which makes this union a partially ordered set. However, all blocks, including the genesis blocks, are totally ordered with respect to time. Although the order of these sets is secret in the absence of users' secret, the total order of the ledger, which is the union of all these sets, is transparent as it is ordered by time. Additionally, ensuring the total order of the blocks inside the ledger implies that the order of the sets is also maintained. A blockchain platform can be used to maintain the integrity of this temporal total order of the ledger. The colored arrows in the figure denote active or passive traversals. In the end, the creation of a new block is shown in the figure, which can be traversed from the last blocks $\tau_u^{(2)}$ and $\tau_v^{(2)}$.

3.3. Challenges

We intend to motivate users toward privacy awareness. Therefore, the users should participate in the security system while using resource-constrained devices, e.g., cell phones. Such devices operate using limited computing abilities and do not have enough storage and memory capabilities. Previous surveys have found that users do not favor apps that consume huge storage [26]. Immutable ledgers like blockchain can grow over time. The current size of the Bitcoin blockchain is 400 GB, and it is growing every day [27]. It may also consume memory to deal with such a big blockchain. Such huge resource requirements may demotivate users and force them to ignore their privacy concerns. Hence, we intend to design our solution in such a way that does not impose large storage or memory requirements on the end users. The proposed system uses a ledger with which every patient can synchronize. However, there can be independent parties synchronizing with the ledger and maintaining the offchain index of blocks. As the block contents are encrypted and immutable, the ledger can be securely accessed by all users. In the proposed scheme, the users use their private credentials to securely compute the address of the block, which they look up in the offchain index. Such a lookup works, irrespective of the location of the ledger. The proposed scheme only requires the users to memorize their secrets and the public key of one entity for the traversal. We facilitate bidirectional traversal over the ledger without requiring any local cache or network communication.

The custodians require one or more pieces of sensitive information associated with multiple patients to find consistency in their healthcare and further medical research. In order to make the solution practically usable, we need to make sure that the process of secure retrieval does not obstruct the actual medical research with time-consuming operations. Hence, the time taken for secure information retrieval and the subsequent ledger entry construction must be deterministic and polynomially proportional to the number of records retrieved.

We secure the identification of de-identified medical information. However, there can be multiple medical information associated with a single patient. We also need to protect the database from attackers who intend to horizontally partition the database into groups such that each group represents the same patient.

4. Secure Ledger

The human entities in the problem are the data managers, supervisors, and patients, as shown in Figure 3. In this paper, we often refer to the data managers and supervisors as custodians because they are in the jurisdiction of the Institution. A custodian is permitted to access and identify the medical record(s) without depending on any other custodian. The responsibility is enforced by the creation of an entry in the immutable ledger. The database is vertically partitioned but not encrypted. Hence, any unauthorized user, if granted access to the database, may see the medical records but will not have the ability to identify the patient associated with the medical record. We choose to keep the data unencrypted to permit fast access and query processing, which is often required for medical research. The focus of this work is to make the custodians responsible for record identification. Therefore, we do not consider data thefts while the stolen data remains non-identifiable. With this setup, an adversary may still use statistical techniques to re-identify the vertically partitioned data. But we ensure that given all records, an adversary cannot partition them into groups, each associated with an individual patient without using some external information. Problems of re-identification can be solved using various techniques like encrypted quasi-identifiable attributes, k-anonymity, l-diversity, differential privacy, etc. However, our paper does not deal with re-identification threats. The focus of our paper is concentrated on enforcing the responsibility of identification.

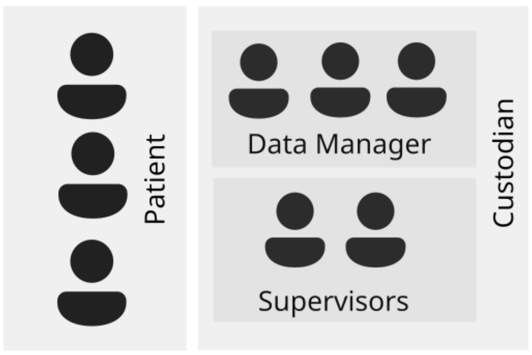


Figure 3. Actors.

As we have already discussed in the previous section, we require pre-processing on the existing medical records in order to perform vertical partitioning and associating cryptographically computable anchors with the partitioned data. We have explained the vertical partitioning and computation of the cryptographic anchor in Section 4.1. We assume that a trustworthy agent has already performed this operation. The proposed scheme uses a trusted server (TS) to resolve either of dp_i^* given dm_i^* and vice versa. The TS is trustworthy enough to follow the protocol. However, it can be compromised by an attacker who may intend to identify some medical records using the secrets found in the storage. Therefore, our scheme requires the cooperation of two entities in order to identify a record, one of which is the TS, and the other is a custodian.

After each identification operation, an entry is written into the immutable ledger. The ledger works like a conventional blockchain, where each block contains the checksum of the previous block. However, in our case, all blocks are not relevant for all users. Hence, the users can selectively browse through the blocks that are associated with them with appropriate credentials. The trusted server (TS) is the only entity that writes into the ledger. The other entities, e.g., the custodians and the patients, can read from it and interact with TS to write to the ledger. Different entities can synchronize with the ledger without requiring any special credentials. However, we require offchain indexing based on some contents of each block for fast search and retrieval. Therefore, those who maintain a copy of the ledger are also expected to maintain an offchain index. The offchain index can be a key-value store that maps two integer addresses (found in each block) with the block ID.

Every user A_t and the trusted server in the system has a public $y_t = g^{\pi_t}, y_w = g^w$ and private keys π_t, w. When generating the private keys, we make sure that $\exists \pi_t^{-1}, w \in \mathbb{Z}_{p-1}$ such that $g^{\pi_t \pi_t^{-1}} \equiv g (mod\ p)$ and $g^{ww^{-1}} \equiv g (mod\ p)$.

4.1. Securely Identifiable Vertical Partitioning

Every record is associated with a patient A_v identified by public key g^{π_v}. Each of these records is vertically partitioned such that the medical information parts of these records $\{M_{v,1}, M_{v,2}, \dots\}$ do not include any identifying information. The identifying information part contain the public key g^{π_v} of the patient and a random number $p_v \in_R [1, 2^{512}]$. We annotate each record $M_{v,j}$ with an anchor $a_{v,j}$ expressed as a tuple. The system is initialized with $\theta \in_{\mathbb{R}} \mathbb{Z}_{p-1}$ such that $\nexists\ \theta^{-1} \in \mathbb{Z}_{p-1}$, but for its hash $h = H(g^\theta)$, $\exists h^{-1}$ such that $g^{hh^{-1}} \equiv g\ (mod\ p)$. The construction of anchor $a_{v,j}$ is shown in Equation (1). The identifying parts are indexed by g^{π_v} and $M_{v,j}$ are indexed by $m_{v,j}$ for fast retrieval.

$$\begin{aligned} a_{v,j} &= (m_{v,j}, \eta_{v,j}, t_{v,j} \in_R [1, 2^{512}]) \\ m_{v,j} &= \xi\Big(g^{\pi_v}, H_2(g^{\theta t_{v,j-1}})\Big) \\ \eta_{v,j} &= t_{v,j-1} H(g^{\theta t_{v,j}}) \end{aligned} \tag{1}$$

With knowledge of $t_{v,j}$ and g^θ, one can compute $H_2(g^{\theta t_{v,j}})$, which is used as the key in the symmetric encryption algorithm ξ. Hence, with that knowledge, it is also possible to decrypt $m_{v,j+1}$ and extract g^{π_v} and identify the patient. $t_{v,j}$ is obtained as a plaintext from $a_{v,j}$. However, one can also extract $t_{v,j}$ from $\eta_{v,j+1}$ by computing the hash of $(g^\theta)^{t_{v,j+1}}$. Therefore, we can find out g^{π_v} from any $m_{v,j}$ $\forall\, j$ in two different ways as shown in Equation (2). The first case is useful while iterating through all records associated with the same patient, while the second is useful for identifying a random record.

$$\zeta\left(m_{v,j}, H_2\Big((g^\theta)^{\frac{\eta_{v,j}}{H((g^\theta)^{t_{v,j}})}}\Big)\right) = g^{\pi_v} = \zeta\Big(m_{v,j}, H_2((g^\theta)^{t_{v,j-1}})\Big) \tag{2}$$

However, in order to iterate through all medical records of a patient, we need a first record with which to start. However, the first record needs a $t_{v,-1}$ which does not exist. Hence, we use p_v associated with the identifying part of the record as shown in Equation (3).

$$\begin{aligned} a_{v,0} &= (m_{v,0}, \eta_{v,0}, t_{v,0} \in_R [1, 2^{512}]) \\ m_{v,0} &= \xi\left(g^{\pi_v}, H_2(g^{\theta p_v})\right) \\ \eta_{v,0} &= p_v H(g^{\theta t_{v,0}}) \end{aligned} \tag{3}$$

Now, in order to iterate through all records belonging to patient A_v, we need p_v, which is included with the identifying part as plaintext and g^θ. However, θ or g^θ is not remembered or stored in any persistent storage. Instead, a key is computed using that and the user's private key, which is shared with all users, as explained in the next section. However, before forgetting the θ, it computes the access key using its private key w for all users A_t and distributes it to them as shown in Equation (4).

$$(g^{\pi_t})^{\theta w} = g^{\pi_t \theta w} \tag{4}$$

A user A_u can use the multiplicative inverse of its private key π_u and send that to the trusted server, which can use the multiplicative inverse of w to recover the g^θ which was previously lost in the beginning as shown in Equation (5).

$$(g^{\pi_u \theta w})^{\pi_u^{-1}} = g^{\theta w} \Rightarrow (g^{\theta w})^{w^{-1}} = g^\theta \tag{5}$$

However, we need to make the exchange in a way that enforces the creation of an entry in the immutable ledger. Therefore, the exchange of shared secrets is incorporated into a protocol. We call that process the Request for Sensitive Information (RSI). RSI can be of two types. One is for retrieving all medical records for a patient. The other is when the custodians want to identify the patients associated with a set of medical records. In the next Section, we formulate the entries in the immutable ledger. In Section 4.3, we explain the construction of the entries, and then we describe the protocol that integrates ledger construction and exchange of access key.

4.2. Ledger Formulation

An access event is associated with two users, one of which is active and the other is passive. The active user invokes the access operation with sufficient credentials, while the passive user may not be aware of the event at that point in time. In our case, the custodians work as active users as they identify the patients' records. The patients remain passive in this event. We use the symbols A_u and A_v to denote the active and the passive users, respectively. Now we translate the problem mentioned in Section 3.2 in Equations (6) and (7) in terms of A_u and A_v. In Equations (6) and (7) $\tau_u^{(k)}, \tau_u^{(k+1)}$ are two consecutive blocks ordered by time in which user A_u was active. Similarly, $\tau_{\hat{v}}^{(k)}, \tau_{\hat{v}}^{(k+1)}$ are two consecutive blocks ordered by time in which user A_v was passive and the arrows denote unidirectional deterministic computability.

$$\overrightarrow{f_a}(\tau_u^{(k)}, \pi_u) \to \tau_u^{(k+1)} \tag{6a}$$

$$\overleftarrow{f_a}(\tau_u^{(k+1)}, \pi_u) \to \tau_u^{(k)} \tag{6b}$$

$$\overrightarrow{f_p}(\tau_{\hat{v}}^{(k)}, \pi_v) \to \tau_{\hat{v}}^{(k+1)} \tag{7a}$$

$$\overleftarrow{f_p}(\tau_{\hat{v}}^{(k+1)}, \pi_v) \to \tau_{\hat{v}}^{(k)} \tag{7b}$$

We refer to Equations (6) and (7) as traversability requirements while Equations (6a), (6b), (7a) and (7b) are referred to as active forward, active backward, passive forward and passive backward traversability requirements, respectively. The blocks must be constructed in such a way that it satisfies these requirements. However, these blocks are constructed as an effect of the access event, which happens in collaboration with the trusted server, which is supposed to enforce the creation of the block. Hence the trusted server should construct the block and post it on the ledger. Every block has three parts, namely address, active, and passive.

The address component is used for indexing each block for fast retrieval, Active and passive parts contain information for the active and passive traversals. Now, we discuss how the different parts of the block are constructed by the trusted server.

4.2.1. Block Address

Each block has a pair of active and passive addresses c_u and $c_{\widehat{v}}$. These values are indexed locally by all maintainers of the ledger so that it does not require a linear search across all blocks to find a block associated with an active or passive address. The address of the k^{th} block depends on the random numbers introduced in the previously active and passive blocks $r_u^{(k-1)}$ and $r_v^{(k-1)}$, respectively. The random numbers $r_u^{(k-1)}$ and $r_v^{(k-1)}$ are contained in the active and the passive parts of the corresponding previous blocks. The addresses are constructed in such a way that given sufficient credentials, one active user can compute $c_u^{(k)}$ using the information in $\tau_u^{(k-1)}$. In addition, a passive user can compute $c_{\widehat{v}}^{(k)}$ using the information in $\tau_{\widehat{v}}^{(k-1)}$ and its credentials. As the addresses $c_u, c_{\widehat{v}}$ are indexed against the block ID $\tau_u, \tau_{\widehat{v}}$, it is possible to find the block associated with that address quickly. The formulation of the addresses is provided in Equation (8).

$$c_u^{(k)} = \tau_u^{(k-1)} H\left(g^{\pi_u r_u^{(k-1)}}\right) \tag{8a}$$

$$c_{\widehat{v}}^{(k)} = \tau_{\widehat{v}}^{(k-1)} H\left(g^{\pi_v r_v^{(k-1)}}\right) \tag{8b}$$

4.2.2. Active Component

Previously, when constructing the address $c_u^{(k)}$ of the current block $\tau_u^{(k)}$, we have mentioned the usage of random number $r_u^{(k-1)}$ which was introduced in the active component of block $\tau_u^{(k-1)}$. Now, we describe the composition of the active component. The active component of the block needs to consist of sufficient information that can be used by the active user to reach the next and the previous blocks in which the same user is active. It consists of two parts, $\overrightarrow{l_u^{(k)}}$, which is used for forward traversal, and $\overleftarrow{l_u^{(k)}}$, which is used for backward traversal by the active user.

Another information that the active component consists of is $l_u^{(k)*}$, which works as a checksum in the process of construction, which is explained later. The composition of the active component is described in Equation (9).

$$\overrightarrow{l_u^{(k)}} = g^{r_u^{(k)}} \tag{9a}$$

$$\overleftarrow{l_u^{(k)}} = H(g^{\pi_u r_v^{(k)}}) g^{r_u^{(k-1)}} \tag{9b}$$

$$l_u^{(k)*} = H\left(g^{w \pi_u r_u^{(k)}} g^{\pi_u}\right) \tag{9c}$$

The traversal function $\overrightarrow{f_a}(\tau_u^{(k-1)}, \pi_u)$ is modeled as shown in Equation (10). The objective of $\overrightarrow{f_a}$ is to make it possible to compute c_u using sufficient credentials, which can then be used to look up the ID of the block.

$$\begin{aligned}\overrightarrow{f_a}(\tau_u^{(k-1)}, \pi_u) &= \tau_u^{(k-1)} H\left(\left(\overrightarrow{l_u^{(k-1)}}\right)^{\pi_u}\right) \\ &= \tau_u^{(k-1)} H\left(g^{\pi_u r_u^{(k-1)}}\right) \\ &= c_u^{(k)}\end{aligned} \tag{10}$$

However, for the active backward traversal, the function $\overleftarrow{f_a}(\tau_u^{(k)}, \pi_u)$ computes the ID of the previous active block without requiring looking up in the index of the addresses. The traversal function $\overleftarrow{f_a}(\tau_u^{(k)}, \pi_u)$ is modeled as shown in Equation (11).

$$\begin{aligned}\overleftarrow{f_a}(\tau_u^{(k)}, \pi_u) &= \frac{c_u}{H\left(\left(\frac{\overleftarrow{l_u^{(k)}}}{H\left(\left(\overrightarrow{l_v^{(k)}}\right)^{\pi_u}\right)}\right)^{\pi_u}\right)} \\ &= \frac{c_u}{H\left(g^{r_u^{(k-1)}\pi_u}\right)} \\ &= \frac{\tau_u^{(k-1)} H\left(g^{\pi_u r_u^{(k-1)}}\right)}{H\left(g^{\pi_u r_u^{(k-1)}}\right)} = \tau_u^{(k-1)}\end{aligned} \tag{11}$$

4.2.3. Passive Component

Like the active component, the passive component of block $\tau_{\widehat{v}}^{(k-1)}$ uses random number $r_v^{(k-1)}$ which is also used to construct the passive address $c_{\widehat{v}}^{(k)}$. The passive component should have sufficient information for the forward and backward traversals $\overrightarrow{f_p}, \overleftarrow{f_p}$. In the active component, we have used the hash of $g^{\pi_u r_v^{(k)}}$ backward traversal. Similarly, here we use the hash of $g^{\pi_v r_u^{(k)}}$ in the passive component. The formulation of the passive component is shown in Equation (12).

$$\overrightarrow{l_{\widehat{v}}^{(k)}} = g^{r_v^{(k)}} \tag{12a}$$

$$\overleftarrow{l_{\widehat{v}}^{(k)}} = H(g^{\pi_v r_u^{(k)}}) g^{r_v^{(k-1)}} \tag{12b}$$

$$l_{vw}^{(k)} = H(g^{w r_v^{(k)}}) g^{h \pi_v r_v^{(k)}} \tag{12c}$$

The forward traversal function $\overrightarrow{f_p}(\tau_{\widehat{v}}^{(k-1)}, \pi_v)$ is modeled as shown in Equation (13). Like the active traversal, the passive user A_v can use its private key π_v to perform the passive traversal.

$$\begin{aligned}\overrightarrow{f_p}(\tau_{\widehat{v}}^{(k-1)}, \pi_v) &= \tau_{\widehat{v}}^{(k-1)} H\left(\left(\overrightarrow{l_{\widehat{v}}^{(k-1)}}\right)^{\pi_v}\right) \\ &= \tau_{\widehat{v}}^{(k-1)} H\left(g^{\pi_v r_v^{(k-1)}}\right) = c_{\widehat{v}}^{(k)}\end{aligned} \tag{13}$$

The passive traversal function computes the ID of the previous block $\tau_{\widehat{v}}^{(k-1)}$ from the information in the block $\tau_{\widehat{v}}^{(k)}$. But $\tau_{\widehat{v}}^{(k)}$ is constructed using $g^{r_v^{(k-1)}}$, which is found in the previous block. However, $g^{r_v^{(k-1)}}$ can also be extracted from $\overleftarrow{l_{\widehat{v}}^{(k)}}$ using $\overleftarrow{l_{\widehat{v}}^{(k)}}$ and π_v. As $\overleftarrow{l_{\widehat{v}}^{(k)}}$

is accessible to the passive user too, it can compute the hash $H(g^{\pi_v r_u^{(k)}})$ and extract $g^{r_v^{(k-1)}}$. The passive traversal function $\overleftarrow{f_p}(\tau_{\hat{v}}^{(k)}, \pi_v)$ is shown in Equation (14).

$$\overleftarrow{f_p}(\tau_{\hat{v}}^{(k)}, \pi_v) = \frac{c_{\hat{v}}}{H\left(\left(\frac{\overleftarrow{l_{\hat{v}}^{(k)}}}{H\left(\left(\overrightarrow{l_u^{(k)}}\right)^{\pi_v}\right)}\right)^{\pi_v}\right)} = \tau_{\hat{v}}^{(k-1)} \tag{14}$$

4.2.4. Genesis Blocks

A genesis block does not have addresses. However, it has a fixed ID, which is deterministically computable by anyone so that a user with proper credentials can traverse forward till the last block without memorizing anything other than its private key. The ID is computed as a hash of the public key of the user as shown in Equation (15). At the same time, anyone with the public key of the user can verify that the user exists in the system. As the genesis block does not describe any actual access event, it does not follow the usual semantics of active and passive users. Instead, the same user is simultaneously considered active and passive.

$$\tau_t^{(0)} = \tau_{\hat{t}}^{(0)} = H(y_t) = H(g^{\pi_t}) \tag{15}$$

As the genesis block is the first block belonging to the user, it is not meant to traverse backward from that block. Hence, only the forward traversal information is sufficient, and we do not include the $\overleftarrow{l_u^{(0)}}$ and $\overleftarrow{l_{\hat{v}}^{(0)}}$ as we have done in other blocks.

$$\overrightarrow{l_u^{(0)}} = g^{r_t^{(0)}} \tag{16a}$$

$$l_u^{(0)*} = H\left(g^{w\pi_t r_t^{(0)}} g^{\pi_t}\right) \tag{16b}$$

$$\overrightarrow{l_{\hat{u}}^{(0)}} = g^{r_t^{(0)}} \tag{17a}$$

$$l_{uw}^{(0)} = H(g^{wr_t^{(0)}}) g^{h\pi_t r_t^{(0)}} \tag{17b}$$

Other than that, the construction of the active and passive components is similar to the regular blocks.

4.3. Sensitive Information Request

Blocks are constructed by the trusted server through active participation with the active user A_u. The request for sensitive information (RSI) contains information regarding which record it wants to retrieve. The trusted server constructs the block registering the event and returns the record(s) requested. While retrieving that record, the trusted server learns about the patient, who is the passive user A_v in this context. With the knowledge of A_u, A_v, and some private information computed by the A_u, the trusted server constructs the block.

We propose a two-stage protocol through which the trusted server obtains sufficient information securely without damaging the privacy of A_u. The protocol is illustrated in Figure 4. In the first stage of the protocol, A_u sends the RSI with the tuple shown in Equation (18). $\tau_u^{(n)}$ is the ID of the last block in which A_u was active. As the blocks are traversable by the active user, it is expected that A_u can have this information even without caching or memorizing anything. g^{π_u} is the public key of A_u. A_u computes the token $g^{\pi_u r_u^{(n)}}$ using $\overrightarrow{l_u^{(n)}}$ and its private key π_u as shown in Equation (19).

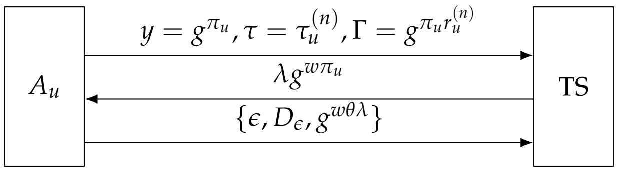

A_u
TS
$y = g^{\pi_u}, \tau = \tau_u^{(n)}, \Gamma = g^{\pi_u r_u^{(n)}}$
$\lambda g^{w\pi_u}$
$\{\epsilon, D_\epsilon, g^{w\theta\lambda}\}$

Figure 4. RSI Protocol.

$$\left(\tau = \tau_u^{(n)}, \Gamma = g^{\pi_u r_u^{(n)}}, y = g^{\pi_u}\right) \tag{18}$$

However, the two parts of the request, $\tau_u^{(n)}$ and g^{π_u}, are claims by A_u. Because the public key is known by anyone, it is not guaranteed that A_u is not using someone else's public key with the intention of impersonating that user. Also, $\tau_u^{(n)}$ may not be the last block in which the user was active. Even if A_u was active, it may not be the last block. It may even be a block that does not exist in the ledger. In all those cases, the ledger will be malformed if these claims are not verified.

$$g^{\pi_u r_u^{(n)}} = \left(\overrightarrow{l_u^{(n)}}\right)^{\pi_u} = \left(g^{r_u^{(n)}}\right)^{\pi_u} \tag{19}$$

Hence, the trusted server needs to verify these claims. To verify the existence by checking equality of $H(y\Gamma^w)$ and $l_u^{(n)*}$ which is available in block $\tau_u^{(n)}$. To satisfy this equality, $\tau = \tau_u^{(n)}$ needs to exist, and $y = g^{\pi_u}$ must be the public key using which $\tau_u^{(n)}$ is constructed. Hence, if $\tau_u^{(n)}$ is constructed using the correct public key, then y is also correct and the same public key. Additionally, the checksum $l_u^{(n)*}$ is matched because the token Γ is correctly computed. But it requires the knowledge of private key π_u to compute that. Hence, the user A_u has the private key π_u.

$$l_u^{(n)*} = H(y\Gamma^w) = H\left(g^{\pi_u}\left(g^{\pi_u r_u^{(n)}}\right)^w\right) = H\left(g^{\pi_u r_u^{(n)} w} g^{\pi_u}\right) \tag{20}$$

However, we do not yet know if $\tau_u^{(n)}$ is the last block in which A_u was active. In order to check that, the trusted server computes $\tau_u^{(n)}\Gamma$ as shown in Equation (21) and searches for its existence in the index of addresses. If it exists, then $\tau_u^{(n)}$ is not the last block in which the user was active because $\tau_u^{(n)}\Gamma = c_u^{(n+1)}$. In that case, the user could be malicious and try to fork the ledger, so that request is rejected. Otherwise, the request is accepted, and it goes to the next stage.

$$\tau_u^{(n)}\Gamma = \tau_u^{(n)} g^{\pi_u r_u^{(n)}} = c_u^{(n+1)} \tag{21}$$

However, in Equation (21), we have already computed $c_u^{(n+1)}$, which is a component of the next block that is due to be constructed.

In the second stage of the protocol, the active user communicates the intended operation and related data along with the access key, with which the trusted server recomputes g^θ, which was lost initially. The trusted server generates a random $\lambda \in Z_q^*$ and sends $\lambda\left(g^{\pi_u}\right)^w = \lambda g^{w\pi_u}$ to the A_u. The server also computes its inverse λ^{-1} and stores it temporarily. A_u can extract the λ by computing $\left(g^w\right)^{\pi_u}$ and then send $g^{\theta\lambda w}$, as shown n Equation (22).

$$\left(g^{\theta\pi_u w}\right)^{\lambda\pi_t^{-1}} = g^{\theta\lambda w} \tag{22}$$

The tuple shown in Equation (23) summarizes the message that the active user sends to the trusted server in Stage 2. The ω and D_ω denote the action that the active user intends to perform and the corresponding data, respectively. The set of possible actions and their corresponding data are represented in Table 2.

$$(\omega, D_w, g^{\theta \pi_u w}) \tag{23}$$

Table 2. Actions in an access event.

ω	D_ω	**Intention**
identify	$a_{v,j}$	Retrieve public key of the patient associated with medical information a_{v_j}
fetch	g^{π_v}	Fetch all records of patient identified by public key g^{π_v}
insert	g_v^{π}, A	Insert records $A = \{a_{v,1} \ldots a_{v,m}\}$ and associate them with patient identified by g^{π_v}
delete	$a_{v,j}$	Delete Record $a_{v,j}$

After receiving the response, the trusted server uses the λ^{-1} to reconstruct g^θ as shown in Equation (24). This g^θ is the secret with which the database anchors are encrypted. After reconstruction of g^θ, the trusted server does not need to remember the value of λ or λ^{-1}.

$$(g^{\theta \lambda w})^{w^{-1}\lambda^{-1}} = g^\theta \tag{24}$$

To construct $\overrightarrow{l_u^{(n+1)}}$, the trusted server generates a $r_u^{(n+1)} \in_{\mathbb{R}} Z_p$ and computes $g^{r_u^{(n+1)}}$. The public key g^{π_u} has been verified in the request stage. Therefore, the checksum $l_u^{(n+1)*}$ can be computed reliably. Hence, the active component is constructed by the trusted server as summarized in Equation (25).

$$\overrightarrow{l_u^{(n+1)}} := g^{r_u^{(n+1)}} \tag{25a}$$

$$\overleftarrow{l_u^{(n+1)}} := H\Big((g^{\pi_u})^{r_v^{(n+1)}}\Big) g^{r_u^{(n)}} = H\Big(g^{\pi_u r_v^{(n+1)}}\Big) g^{r_u^{(n)}} \tag{25b}$$

$$l_u^{(n+1)*} := H\Big((g^{\pi_u})^{w r_u^{(n+1)}} g^{\pi_u}\Big) = H\Big(g^{\pi_u w r_u^{(n+1)}} g^{\pi_u}\Big) \tag{25c}$$

However, while generating the random number $r_u^{(n+1)}$, we have to be careful that the inequality shown in Equation (26) has to be satisfied. More on this inequality is explained later at the end of Section 4.4. Additionally, $r_u^{(n+1)}$ may not be invertible in $Z_{(p-1)}$.

$$H_2\Big(g^{\pi_v r_u^{(n+1)}}\Big) \not\equiv H_2\Big(g^{\pi_u r_u^{(n)}}\Big) \ (mod\ 2) \tag{26}$$

The trusted server has already computed g^θ as shown in Equation (24). Hence, if the RSI is for identifying medical records, then the value in id^* can be decrypted using Equation (2).

If the RSI is for retrieving all records of a patient, then the g^{π_v} is provided in the request. For the passive component, the random number $r_v^{(n+1)} \in_{\mathbb{R}} Z_p$ is generated such that the first one has no multiplicative inverse in $Z_{(p-1)}$ but the second one does not. This construction also uses the random number $r_u^{(n+1)}$ generated for the active component. The construction is summarized in Equation (27). The random number $r_v^{(n+1)}$ is also used to compute the $\overleftarrow{l_u^{(n+1)}}$. As, the trusted server has recomputed the forgotten g^θ using which it can now also compute the hash $h = H(g^\theta)$ and construct $l_{vw}^{(n+1)}$.

$$\overrightarrow{l_{\widehat{v}}^{(n+1)}} := g^{r_v^{(n+1)}} \tag{27a}$$

$$\overleftarrow{l_{\widehat{v}}^{(n+1)}} := H\Big((g^{\pi_v})^{r_u^{(n+1)}}\Big)g^{r_v^{(n)}} = H\Big(g^{\pi_v r_u^{(n+1)}}\Big)g^{r_v^{(n)}} \tag{27b}$$

$$l_{vw}^{(n+1)} := H\Big((g^{w})^{r_v^{(n+1)}}\Big)\Big(g^{\pi_v}\Big)^{hr_v^{(n+1)}} = H\Big(g^{wr_v^{(n+1)}}\Big)g^{\pi_v hr_v^{(n+1)}} \tag{27c}$$

The active part of the address component can be computed as shown in Equation (21). The construction of the address component is summarized in Equation (28).

$$c_u^{(n+1)} := \tau_u^{(n)}\Gamma \tag{28a}$$

$$\begin{aligned} c_{\widehat{v}}^{(n+1)} &:= \tau_{\widehat{v}}^{(n)}\left(\frac{l_{vw}^{(n)}}{H\Big(\Big(\overrightarrow{l_{\widehat{v}}^{(n)}}\Big)^{w}\Big)}\right)^{h^{-1}} \\ &= \tau_{\widehat{v}}^{(n)}\left(\frac{H(g^{wr_v^{(n)}})g^{h\pi_v r_v^{(n)}}}{H(g^{wr_v^{(n)}})}\right)^{h^{-1}} \\ &= \tau_{\widehat{v}}^{(n)}\Big(g^{h\pi_v r_v^{(k)}}\Big)^{h^{-1}} \\ &= \tau_{\widehat{v}}^{(n)}\Big(g^{\pi_v r_v^{(k)}}\Big) \end{aligned} \tag{28b}$$

4.4. *Encrypting Block Contents*

Each block should have content explaining the reason behind the access event. Additionally, it should include information about the active and passive users associated with it. The encryption technique we apply to the contents of the block should satisfy the following properties.

1. Given a random block, only the active user A_u involved in the creation of that block will be able to decrypt its content.
2. Given a random block, only the passive user A_v whose data have been accessed will be able to decrypt its content.
3. A supervisor can decrypt the contents of any block.
4. It should be possible to decrypt the contents of a block offline without connecting to any other server.

Such requirements are very similar to the well-known secret-sharing problem such as Samir's Secret-Sharing. While using the secret-sharing scheme, the ciphertext can be decrypted using a secret, which is shared among multiple entities. It can be recovered by the participation of a threshold number of actors who hold shares of that secret. However, in our case, it does not require the participation of more than one actor in order to decrypt, which is similar to the Diffie–Hellman key exchange. In the Diffie–Hellman key exchange, the two participants use symmetric encryption for communication. However, the key they use for symmetric encryption is derived from both of their secrets, which are not revealed but exchanged securely before the data communication begins. In our case, the active user A_u is not communicating with the passive user A_v or the supervisors. However, both A_v and supervisors need the ability to decrypt the ciphertext. The encryption scheme that we follow for encryption of the block content is discussed below.

The TS formulates a straight line primarily with two coordinates (shown in Equation (29)) that only the users A_u and A_v can compute. Although the TS can compute those coordinates during construction, it loses the information it needs to reconstruct that again. Once the straight line is constructed, it finds two random coordinates that satisfy that linear equation. One of those coordinates is published with that block as plain text along with the x value of the other. The cryptographic hash of the y value of the other random coordinate is used as a password to symmetrically encrypt the message.

$$d^{(u)} = \begin{bmatrix} H_2\left(g^{\pi_u r_u^{(n)}}\right) \\ c_v \end{bmatrix}, \qquad d^{(v)} = \begin{bmatrix} H_2\left(g^{\pi_v r_u^{(n+1)}}\right) \\ c_u \end{bmatrix} \tag{29}$$

The line can be formulated as $ax + by = c$ where the coefficients can be computed as shown in Equation (30). However, we can divide both sides with the $e = Gcd(a, b)$, and the equation remains the same.

$$\begin{aligned} a &= d_y^{(v)} - d_y^{(u)} \\ b &= d_x^{(u)} - d_x^{(v)} \\ c &= d_y^{(v)} d_x^{(u)} - d_x^{(v)} d_y^{(u)} \end{aligned} \tag{30}$$

As we need to find integer coordinates only, we can consider the straight line as a linear diophantine equation. In that case, we can generate a random coordinate r on that line using Equation (31). The s on Equation (31) can be any coordinate on that line. As we already have two coordinates, we can choose either $d^{(u)}$ and $d^{(v)}$.

$$r = s + \Delta m \ \forall \ m \in \mathcal{Z} \ where \ \Delta = \begin{bmatrix} \frac{b}{e} \\ -\frac{a}{e} \end{bmatrix} \tag{31}$$

We generate two random coordinates γ, δ using Equation (31). γ and δ_x are published with the block. The contents are encrypted using $H_2(\delta_y)$. This straight line is shown in Figure 5. Both active and passive users can compute either $d^{(u)}$ or $d^{(v)}$ and then interpolate a straight line using γ, which is available as plaintext. Then, the users put $x = \delta_x$ on that equation and calculate $y = \delta_y$. Once δ_y is retrieved, its hash $H_2(\delta_y)$ can be computed, with which the encrypted message can be decrypted and the plaintext can be obtained.

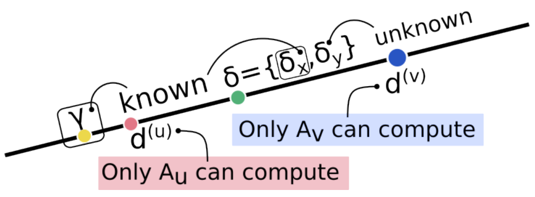


Figure 5. Straight line used for encrypting block content.

However, the supervisors cannot compute either $d^{(u)}$ or $d^{(v)}$, hence cannot use γ to interpolate the straight line. As the supervisors can also perform access events, they too have their access key like the data managers. Additionally, they have $g^{\phi w \pi_s}$, which is specifically used for viewing, but unlike g^{θ}, it is not lost by the TS. Although g^{θ} is lost, it is reconstructed in the second stage of the protocol by the TS. Hence, the TS computes a suffix $\left(g^{\theta w} g^{\phi w}\right)^{\gamma_x}$ and multiplies the $H_2(\delta_y)$, as shown in Equation (32), and stores that ciphertext in the block.

$$\delta_y^{(s)} = H_2(\delta_y)\left(g^{\theta w} g^{\phi w}\right)^{\gamma_x} = H_2(\delta_y) g^{(\theta+\phi) w \gamma_x} \tag{32}$$

The supervisors A_s can compute the suffix and obtain $H_2(\delta_y)$ by division in $\mathcal{Z}_p$, as shown in Equation (33).

$$\frac{\delta_y^{(s)}}{\left((g^{\phi w \pi_s})^{\pi_s^{-1}} (g^{\theta w \pi_s})^{\pi_s^{-1}}\right)^{\gamma_x}} = \frac{H_2(\delta_y) g^{(\theta+\phi) w \gamma_x}}{\left(g^{\phi w} g^{\theta w}\right)^{\gamma_x}} = \frac{H_2(\delta_y) g^{(\theta+\phi) w \gamma_x}}{g^{(\phi+\theta) w \gamma_x}} = H_2(\delta_y) \tag{33}$$

However, the A_u and A_v also knows $H_2(\delta_y)$ and can extract the suffix $g^{(\theta+\phi)w\gamma_x}$ through division. As γ_x is known, anyone can compute γ_x^{-1} and extract $g^{(\theta+\phi)w} = (g^{(\theta+\phi)w\gamma_x})^{\gamma_x^{-1}}$ using that. Once extracted, it can be reused with some other γ_x associated with some other block. But if we can ensure that $\nexists\gamma_x^{-1}$, such that $\gamma_x\gamma_x^{-1} \equiv 1 \in \mathcal{Z}_{(p-1)}$, then the malicious user cannot reuse that suffix. γ_x is non-invertible only if it is not coprime with $p-1$, i.e., $Gcd(\gamma_x, p-1) > 1$. Given that p is a prime and any prime greater than 2 is odd, p is odd, and $p-1$ is even. Hence, if γ_x is also even, then their Gcd is at least 2. However, γ_x is computed in Equation (31) using a random integer m that contributes to γ_x being odd or even. γ_x can be expressed by Equation (34).

$$\gamma_x = s_x + \Delta_x m \tag{34}$$

If s_x is even, then γ_x is also even only if $\Delta_x m$ is even. We can trivially ensure that $\Delta_x m$ is even by making m an even random number even if Δ_x is odd. If s_x is odd, we need $\Delta_x m$ to be odd in order to make γ_x even. If Δ_x is odd, then choosing an odd m will result in an even γ_x. However, if Δ_x itself is even, then we cannot make it odd by finding an odd or even m. In that case, γ_x is odd. In order to make an odd number non-coprime with $(p-1)$ (which is even), γ_x must be divisible by some other factors of $(p-1)$. If $(p-1)$ is the product of a few large primes, one of which is 2, then it will be difficult to find a random number m that results in an even $\gamma_x$because now there are <50% random numbers from which we can choose. The number of favorable random numbers decreases as the smallest factors of $p-1$ (other than 2) increase. If we can ensure that Δ_x is always even, then we can avoid such an exhaustive search. For that, we need to ensure that $d_x^{(u)} - d_x^{(v)}$ is odd, as shown in Equation (30). Now, both of these values $d_x^{(u)}$ and $d_x^{(v)}$ are results of hashing, and there are exactly 50% odd and 50% even numbers in the output space. $d_x^{(u)}$ is determined already because it depends on $r_u^{(n)}$, which was decided while constructing the previous block. However, $r_u^{(n+1)}$ is decided while constructing the current block. Hence, if $d_x^{(u)}$ is odd, then we need to find a $r_u^{(n+1)}$ such that $H_2\left(g^{\pi_v r_u^{(n+1)}}\right)$ is even. And if $d_x^{(u)}$ is even, then we need to find a $r_u^{(n+1)}$ such that $H_2\left(g^{\pi_v r_u^{(n+1)}}\right)$ is odd. Hence, although block construction is a non-deterministic process, there is a 50% probability of termination in each iteration.

Finally, a checksum of the block is calculated, and the blocks are signed by the trusted server. All the terms that we have computed for a block are shown in Figure 6. At first, the block addresses are shown in yellow. Then, the active and the passive components of the block are shown in light red and blue, respectively. In the end, we show the block contents part in green.

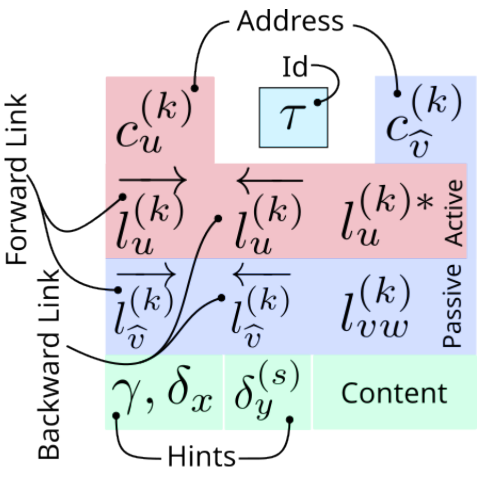


Figure 6. Block design.

In the above sections, we have explained different parts of our proposed system. In Section 4.3, we have discussed the protocol that active users (data managers) use to interact with TS. We have shown that the request for sensitive information (RSI) originates from an active user, leading to the creation of an entry in the ledger documenting the access event. Someone who is not a custodian but who is trying to interact with the trusted server for the de-identification of patients' health data is considered to be an adversary. Additionally, a data manager who is trying to gain information about an entry for an access event in which some other data manager was involved is also considered to be adversarial access. The patients are supposed to be able to retrieve information only about the events in which that patient was involved. Otherwise, this is considered to be adversarial access. In the next section, we present a list of possible adversarial accesses. Then, we show that the performance of each of these adversarial accesses is practically impossible for a polynomial adversary.

5. Evaluation

We summarize the computational overhead for active and passive traversals in terms of the number of arithmetic operations in Table 3. The columns in the table represent multiplication, exponentiation, inversion, and hashing, respectively. We can observe that the backward traversals require more computation than the forward traversal.

Table 3. Computations required for traversals.

Action		**Number of Operations**			
		ab	a^b	a^{-1}	$H(a)$
Traversal	Forward	1	1	0	1
	Backward	2	2	2	2

Now, we evaluate our proposed framework theoretically as well as experimentally in the following sections.

5.1. Adversarial Evaluation

First, we evaluate our proposed framework in terms of its usability and security concerns. The objective of this paper is to propose a secure scheme to ensure the responsibility and privacy of the involved entities, and the proposed work is deterministic. In Section 1, we have described the primary use case of the work. Here, we theoretically analyze the related security concerns and discuss how these concerns are addressed.

5.1.1. Adversarial Model

Now, we analyze the threats that the proposed system may encounter. The entities involved in the proposed system are a vertically partitioned database, a trusted server, patients, and custodians (data managers and supervisors). We consider that the trusted server is honest but curious and follows the protocol, but it may use the secret it stores to gain information that it is not allowed to obtain. We claim that our security solution is resilient against scenarios when the trusted server is compromised, but the custodians do not interact with the compromised server. Malicious users, including the custodians, may try to gain information from the blockchain. We ensure that the information contained in the blockchain should not disclose information regarding the association of the active and the passive users with any event block. A list of adversarial attack scenarios motivated by malicious intentions is presented below.

1. Find records associated with a patient while compromising the trusted server and the database. (Section 5.1.2)
2. All custodians collude to perform (1) while compromising the database but not the trusted server (Section 5.1.2)

3. Perform active or passive traversal without compromising any entity in the system. (Theorems 2–5 in Section 5.1.3)
4. Partition blocks into groups, each associated with a user without compromising any entity in the system. (Theorems 6 and 7 in Section 5.1.3)
5. Read contents of a random block in the blockchain without compromising any entity in the system. (Theorem 8 in Section 5.1.3)
6. Adversarial custodian performing access event while impersonating another custodian without compromising the victim's private key. (Theorem 9 in Section 5.1.3)

We analyze the security of our proposed work based on computational and decisional Diffie–Hellman assumptions [28]. The computational and decisional problems are defined in Definitions 1 and 2. There have been several generalizations of Diffie–Hellman assumptions because the application is group-based protocols [29,30]. Based on these assumptions, we define the generalized computational Diffie–Hellman (GenCDH) problem in Definition 3. In Definition 4, we define the security assumptions about the cryptographic hash function H.

Definition 1. *Given a cyclic group G of order q, with generator g, and $\{g^a, g^b\}$, the computational Diffie–Hellman (CDH) problem is to compute g^{ab}, where $a, b \in_R Z_q^*$.*

Definition 2. *Given a cyclic group G of order q, with generator g, and $\{g^a, g^b\}$, the decisional Diffie–Hellman (CDH) problem is to distinguish g^{ab} from g^c, where $a, b, c \in_R Z_q^*$.*

Definition 3. *Given a cyclic group G of order q, with generator g, and $\{g^{r_1}, \ldots, g^{r_n}\}$ the generalized computational Diffie–Hellman (GenCDH) problem is to compute $g^{r_1 \cdots r_n}$ where $r_i \in_R Z_q^* \forall\, i \in [1, n]$.*

Definition 4. *A cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^n$ is a deterministic function that satisfies the following properties:*

Pre-image resistance: *Given a hash value h, it is computationally infeasible to find any input x such that $H(x) = h$.*

Second pre-image resistance: *Given an input x, it is computationally infeasible to find another input $x' \neq x$ such that $H(x) = H(x')$.*

Collision resistance: *It is computationally infeasible to find any two distinct inputs x and x' such that $H(x) = H(x')$.*

Additionally, we assume that H satisfies the non-correlation property as mentioned in [31]. The non-correlation property implies that the inputs and the outputs of the hash function should not be statistically correlated.

Based on these definitions, we present Lemmas 1 and 2, which we will extensively use to demonstrate the difficulties of adversarial access in the rest of the section.

Lemma 1. *Computing $H(g^{ab})g^c$ from g^c is intractable under the CDH assumption when a, b are unknown exponents but g^a, g^b are known, and H is an irreversible cryptographic hash function.*

Proof. Let us assume that the adversary $\mathcal{A}$ can compute $H(g^{ab})g^c$ from g^c using an efficient oracle $\mathcal{B}$.

Assumption:

Given a cyclic group G of order q, with generator g $\mathcal{B}(g^a, g^b, g^c) = H(g^{ab})g^c$

Goal:

Construct an algorithm $\mathcal{C}$ using oracle $\mathcal{B}$ to compute g^{ab} from g^a and g^b without knowing a, b.

Inputs: g, g^a, g^b, g^c
Unknowns: a, b

$$\begin{aligned} &\mathcal{B}(g^a, g^b, g^c) &&= H(g^{ab})g^c \\ \Rightarrow &\mathcal{B}(g^a, g^b, g) &&= H(g^{ab})g \text{ substitute } c = 1 \\ \Rightarrow &\mathcal{B}(g^a, g^b, g)/g = H(g^{ab}) \end{aligned}$$

1. Hence, $\mathcal{A}$ can construct algorithm $\mathcal{C} = \mathcal{B}(g^a, g^b, g)/g$ to compute $H(g^{ab})$.
2. However, H being an irreversible cryptographic hash function, the only way to compute $H(g^{ab})$ is to be able to compute g^{ab}.
3. However, computing g^{ab} using g^a, g^b without knowledge of a, b is intractable under CDH assumption.

$\therefore$ if oracle $\mathcal{B}$ can be used to compute g^{ab}, then the adversary $\mathcal{A}$ can use $\mathcal{B}$ to solve the CDH problem. □

Lemma 2. *Computing $H(g^{war}g^a)$ from $w, g^a, g^r, H(g^{ar})$ is intractable under the CDH assumption when a, r are unknown exponents but g^a, g^r are known and H is an irreversible cryptographic hash function.*

Proof. Let us assume that the adversary $\mathcal{A}$ can compute $H(g^{war}g^a)$ from $w, g^a, g^r, H(g^{ar})$. However, H is an irreversible cryptographic hash function. Hence, knowledge of $H(g^{ar})$ does not provide any advantage, and in order to construct $H(g^{war}g^a)$. Also, because H is irreversible cryptographic hash function, $g^{war}g^a$ needs to be computed first in order to compute $H(g^{war}g^a)$. Now, let us assume that an adversary $\mathcal{A}$ has an oracle $\mathcal{B}$ that can compute $g^{war}g^a$ given w, g^a, g^r without knowing a, r.

Assumption:

Given a cyclic group G of order q, with generator g $\mathcal{B}(g^a, g^r, w) = g^{war}g^a$

Goal:

Construct an algorithm $\mathcal{C}$ using oracle $\mathcal{B}$ to compute $g^{war}g^a$ from w, g^a, g^r without knowing a, b.
Inputs: g, g^a, g^r, w
Unknowns: a, r

$$\begin{aligned} &\mathcal{B}(w, g^a, g^r) &&= g^{war}g^a \\ \Rightarrow &\mathcal{B}(1, g^a, g^r) &&= g^{ar}g^a \quad \text{substitute } w = 1 \\ \Rightarrow &\mathcal{B}(1, g^a, g^r)/g^a = g^{ar} \end{aligned}$$

1. Hence $\mathcal{A}$ can construct algorithm $\mathcal{C} = \mathcal{B}(1, g^a, g^r)/g^a$ to compute g^{ar}.
2. But computing g^{ab} using g^a, g^b without knowledge of a, b is intractable under CDH assumption.

$\therefore$ if the oracle $\mathcal{B}$ can be used to compute g^{ar} from g^a, g^r while a, r are unknown, then the adversary $\mathcal{A}$ can use it to solve the CDH problem. □

We now analyze the security of our proposed work against the above-mentioned threats. Throughout this section, we show that the adversary incapable of solving CDH, DDH, and GenCDH problems cannot solve the challenges required to perform the attacks.

Attacks (1–3) are the computational challenge that the adversary must perform, while (4) requires the adversary to solve a decisional challenge. In Section 5.1.2, we discuss the first two scenarios related to attacks on the database concerning the disclosure of the de-identified data.

The attack scenarios targeting the block are discussed in Section 5.1.3. In Theorems 2 and 3, we prove that Attack (3) requires the malicious user to solve a problem that is as difficult as CDH. In Theorems 4 and 5, we prove that the decision version of this problem is as difficult as the computational version. To prove the difficulty of Attack (4), we first prove the difficulty of decidability of block participation without knowledge of the user's private key in Theorem 6. Then we show that to solve the partitioning problem in (4), the malicious user needs to have an oracle that can solve problems described in Theorems 4–6, which makes it as difficult as CDH too. In Theorems 8 and 9, we prove that Attacks (5,6) require the malicious user to solve a problem that is as difficult as CDH.

5.1.2. Adversarial Attacks on Storage

In this section, we discuss Attacks (1) and (2), which are about the ability of an attacker to find records associated with a patient while assuming that the trusted server and the database have been compromised (i.e., the attacker has gained access to it). We do not consider the case of re-identification as it is out of the focus of this work. We only consider the case of direct identification of a patient from the corresponding medical information. As mentioned above in Equation (1), each entry in the medical information is annotated with an anchor $a_{v,j}$, which is a tuple $(m_{v,j}, \eta_{v,j}, t_{v,j})$. An attacker can identify a patient from corresponding medical information only if it is possible to derive the identifying information of the patient (public key g^{π_v}) from $a_{v,j}$. We formalize this identification problem in Theorem 1.

Theorem 1. *Given an anchor $a_{v,j}$, the difficulty of associating that with a public key of any patient or finding out the next record associated with the same patient depends on the difficulty of decrypting ciphertext computed using symmetric algorithm ξ and difficulty of solving the CDH problem, when θ or g^θ are not known and only the trusted server is compromised by an external adversary.*

Proof. The only term associated with a record $a_{v,j}$ that is derived using the public key g^{π_v} is $m_{v,j}$, as shown in Equation (1). Hence, if the attacker derives g^{π_v}, it must be derived from $m_{v,j}$ only.

However, $m_{v,j}$ is a ciphertext computed using symmetric encryption algorithm ξ using $H_2(g^{\theta t_{v,j-1}})$ as symmetric key. We assume that the attacker cannot decrypt the ciphertext without knowing the symmetric key. Hence, to derive g^{π_v}, the symmetric key must be constructed by the attacker.

But, computing the symmetric key $g^{\theta t_{v,j-1}}$ from $g^{t_{v,j-1}}$ without knowledge of θ is as difficult as CDH problem.

Neither θ nor g^θ is known by the TS. Hence, compromising the TS does not provide any advantage for the attacker.

Traversing to the next block requires the encryption of g^{π_v}. However, we have already shown that computing g^{π_v} is as difficult as CDH. □

However, it is important to note that a subset of users (custodians) have access keys from which they can extract $g^{\theta w}$. However, computing $g^{\theta r}$ from r and $g^{\theta w}$ is assumed to be intractable when w is not known. Moreover, given n custodians exchanging their access keys $\{g^{\theta w \pi_{t_1}}, \ldots, g^{\theta w \pi_{t_n}}\}$, computing g^θ is as difficult as GenCDH.

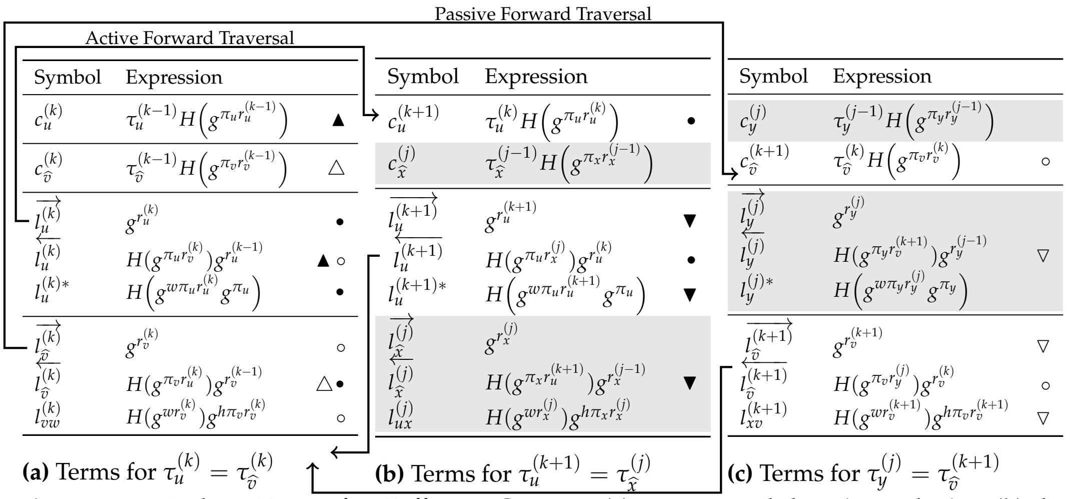
However, if the adversary compromises the trusted server as well as a custodian, then it has access to both w and $g^{\theta w}$, with which it can compute g^θ and thus $g^{\theta r}$. However, in this work, we do not provide solutions for such scenarios. We only assume that no other users are compromised when the trusted server is compromised.

5.1.3. Adversarial Attacks on Ledger

Attacks (3–6) are about adversarial attacks on the ledger. Attacks (3) and (4) are about adversarial active and passive traversal of the ledger, which is possible only if the adversary can decide whether the two given blocks are the next or previous block of each other in terms of active or passive traversal. In Theorems 4 and 5, we show that this decidability depends on the computability of components of one block using components of the other block. We formalize this computability problem in Theorems 2 and 3.

Attack (5), which is about the confidentiality of the contents of the block, is formalized in Theorem 8. Then, we show that the difficulty of that attack depends on the difficulty of Attacks (3) and (4). Finally, in Theorem 9, we discuss the difficulty of Attack (6), which is about the impersonation of one custodian by another.

Now, we revisit Equations (8), (9) and (12) and present all the traversal-related terms regarding three blocks as shown in Figure 7. The first block shown in Figure 7a denotes the block $\tau_u^{(k)} = \tau_{\widehat{v}}^{(k)}$, which is participated in by active user A_u and passive user A_v. From this block, the active user A_u can perform active forward traversal and reach the second block $\tau_u^{(k+1)}$ shown in Figure 7b which may be participated by a different passive user A_x. We assume that it is a $j'th$ block that the passive user A_x can reach through passive forward traversal. Although this user A_x may be irrelevant to our discussion in this section, we have included it for completeness in the table. Similarly, we represent the next block in Figure 7c that the passive user A_v can reach through passive forward traversal as $\tau_{\widehat{v}}^{(k+1)}$, which may be participated in by a different active user A_y. Again, even though this active user A_y may be irrelevant to our discussion, we have included it in the table for completeness and better understandability. The irrelevant parts, i.e., the passive part of Figure 7b and the active part of Figure 7c, are shaded.


Active Forward Traversal
Passive Forward Traversal

Symbol	Expression	
$c_u^{(k)}$	$\tau_u^{(k-1)} H\left(g^{\pi_u r_u^{(k-1)}}\right)$	▲
$c_{\widehat{v}}^{(k)}$	$\tau_{\widehat{v}}^{(k-1)} H\left(g^{\pi_v r_v^{(k-1)}}\right)$	△
$\overrightarrow{l_u^{(k)}}$	$g^{r_u^{(k)}}$	•
$\overleftarrow{l_u^{(k)}}$	$H(g^{\pi_u r_v^{(k)}}) g^{r_u^{(k-1)}}$	▲ ○
$l_u^{(k)*}$	$H\left(g^{w\pi_u r_u^{(k)}} g^{\pi_u}\right)$	•
$\overrightarrow{l_{\widehat{v}}^{(k)}}$	$g^{r_v^{(k)}}$	○
$\overleftarrow{l_{\widehat{v}}^{(k)}}$	$H(g^{\pi_v r_u^{(k)}}) g^{r_v^{(k-1)}}$	△•
$l_{vw}^{(k)}$	$H(g^{w r_v^{(k)}}) g^{h\pi_v r_v^{(k)}}$	○

(a) Terms for $\tau_u^{(k)} = \tau_{\widehat{v}}^{(k)}$

Symbol	Expression	
$c_u^{(k+1)}$	$\tau_u^{(k)} H\left(g^{\pi_u r_u^{(k)}}\right)$	•
$c_{\widehat{x}}^{(j)}$	$\tau_{\widehat{x}}^{(j-1)} H\left(g^{\pi_x r_x^{(j-1)}}\right)$	
$\overrightarrow{l_u^{(k+1)}}$	$g^{r_u^{(k+1)}}$	▼
$\overleftarrow{l_u^{(k+1)}}$	$H(g^{\pi_u r_x^{(j)}}) g^{r_u^{(k)}}$	•
$l_u^{(k+1)*}$	$H\left(g^{w\pi_u r_u^{(k+1)}} g^{\pi_u}\right)$	▼
$\overrightarrow{l_{\widehat{x}}^{(j)}}$	$g^{r_x^{(j)}}$	
$\overleftarrow{l_{\widehat{x}}^{(j)}}$	$H(g^{\pi_x r_u^{(k+1)}}) g^{r_x^{(j-1)}}$	▼
$l_{ux}^{(j)}$	$H(g^{w r_x^{(j)}}) g^{h\pi_x r_x^{(j)}}$	

(b) Terms for $\tau_u^{(k+1)} = \tau_{\widehat{x}}^{(j)}$

Symbol	Expression	
$c_y^{(j)}$	$\tau_y^{(j-1)} H\left(g^{\pi_y r_y^{(j-1)}}\right)$	
$c_{\widehat{v}}^{(k+1)}$	$\tau_{\widehat{v}}^{(k)} H\left(g^{\pi_v r_v^{(k)}}\right)$	○
$\overrightarrow{l_y^{(j)}}$	$g^{r_y^{(j)}}$	
$\overleftarrow{l_y^{(j)}}$	$H(g^{\pi_y r_v^{(k+1)}}) g^{r_y^{(j-1)}}$	▽
$l_y^{(j)*}$	$H\left(g^{w\pi_y r_y^{(j)}} g^{\pi_y}\right)$	
$\overrightarrow{l_{\widehat{v}}^{(k+1)}}$	$g^{r_v^{(k+1)}}$	▽
$\overleftarrow{l_{\widehat{v}}^{(k+1)}}$	$H(g^{\pi_v r_y^{(j)}}) g^{r_v^{(k)}}$	○
$l_{xv}^{(k+1)}$	$H(g^{w r_v^{(k+1)}}) g^{h\pi_v r_v^{(k+1)}}$	▽

(c) Terms for $\tau_y^{(j)} = \tau_{\widehat{v}}^{(k+1)}$

Figure 7. Ledger Terms for Different Cases. **(a)** participated by A_u and A_v, **(b)** by A_u and irrelevant passive user A_x, **(c)** by an irrelevant active user A_y and A_v. **Legends**: Terms involving random numbers (•$r_u^{(k)}$), (○$r_v^{(k)}$), (▲ $r_u^{(k-1)}$), (△ $r_v^{(k-1)}$), (▼ $r_u^{(k+1)}$), (▽$r_v^{(k+1)}$).

It is important to note that for the computation of different terms shown in these three tables, different random numbers have been used. We have used different legends to mark the terms that use the same random number. For example, a filled circle (•) has been used to mark the terms that use random number $r_u^{(k)}$. We also note that, $\forall\ r_1, r_2 \in Z_q^*$, given $g^{r_1}, g^{\pi_t r_1}$, computing $g^{\pi_t r_2}$ is as difficult as solving the GenCDH problem. So, the expressions that are derived from different random numbers are incompatible and to compute one from the other is intractable.

We now analyze what information a polynomial-time adversary can gain by reading the access blocks in the ledger based on two blocks $\tau_u^{(k)}, \tau^{(n)}$. We consider the case when $\tau^{(n)} = \tau_u^{(k+1)}$ as shown in Figure 7b and demonstrate the computational difficulty of a polynomial-time adversary.

To demonstrate the intractability of adversarial active traversals, we first group the cryptographically relevant terms of $\tau_u^{(k)}$ (shown in Figure 7a) and $\tau_u^{(k+1)}$ (shown in Figure 7b) in two different sets. Then, we show that under CDH assumptions, it is intractable for an adversary to compute any term of one set using at least one term from the other and vice versa in Theorem 2. Similarly, in Theorem 3, we create two sets of cryptographically relevant terms from $\tau_{\widehat{v}}^{(k)}$ (shown in Figure 7a) and $\tau_{\widehat{v}}^{(k+1)}$ (shown in Figure 7c) and show the intractability of adversarial passive traversal.

Theorem 2. *Given g^{π_u}, π_v, w, and two blocks $\tau_u^{(k)}, \tau^{(n)} = \tau_u^{(k+1)}$, computing any term of one block in Figure 7a,b from a subset of terms in another block is as difficult as the CDH problem.*

Proof. The terms in both of these blocks that are computed using the same random number are marked with the same symbol in Figure 7a,b. Here, we group the terms of the first and the second block (marked with filled circle •) that are computed using $r_u^{(k)}$ as $\beta(u,v)$ and $\beta(u,x)$. The other random number $r_v^{(k-1)}$ is used in $\beta(u,v)$ but has not been used to construct any term in $\tau_u^{(k+1)}$. Similarly, the random $r_x^{(j)}$ is not used for construction of any term in $\tau_u^{(k)}$.

$$\beta(u,v) = \{\overrightarrow{l_u^{(k)}}, l_u^{(k)*}, \overleftarrow{l_{\widehat{v}}^{(k)}}\} \qquad = \{g^{r_u^{(k)}}, H\left(g^{w\pi_u r_u^{(k)}} g^{\pi_u}\right), H(g^{\pi_v r_u^{(k)}}) g^{r_v^{(k-1)}}\}$$

$$\beta(u,x) = \{c_u^{(k+1)}, \overleftarrow{l_u^{(k+1)}}\} \qquad = \{\tau_u^{(k)} H\left(g^{\pi_u r_u^{(k)}}\right), H(g^{\pi_u r_x^{(j)}}) g^{r_u^{(k)}}\}$$

To prove the theorem, we demonstrate that the computability of each term in $\beta(u,x)$, using at least one term of $\beta(u,v)$, is as difficult as CDH in Arguments 1 and 2. Then, we show in Arguments 3–5 that computability of any term of $\beta(u,v)$, using at least one term of $\beta(u,x)$, is as difficult as CDH.

Argument 1.

1. To compute $c_u^{(k+1)}$, an adversary needs to compute $H\left(g^{\pi_u r_u^{(k)}}\right)$ first.
2. Given that $\pi_u, r_u^{(k)}$ are unknowns, and H is a cryptographic hash function, computing $g^{\pi_u r_u^{(k)}}$ from $g^{\pi_u}, g^{r_u^{(k)}}$ is equivalent to solving CDH.

Argument 2.

1. Considering $\overleftarrow{l_u^{(k+1)}} = H(g^{\pi_u r_x^{(j)}}) g^{r_u^{(k)}} \in \beta(u,x)$ in context of Lemma 1, we substitute the unknown exponents $a = \pi_u, b = r_x^{(j)}$ and known $g^c = g^{r_u^{(k)}}$.
2. Computing $H(g^{\pi_u r_x^{(j)}}) g^{r_u^{(k)}}$ is intractable under the CDH assumption.

Hence, computing any of the terms in $\beta(u,x)$ from $\beta(u,v)$ while $\pi_u, r_u^{(k)}, r_x^{(j)}$ are unknown exponents is intractable under the CDH assumption. Additionally, none of these unknown exponents are present in $\beta(u,v)$.

$\therefore$, computing any term of $\beta(u,x)$ using at least one term of $\beta(u,v)$ is intractable under the CDH assumption.

We now present our arguments to show that the computability of the terms in $\beta(u,v)$ from a subset of terms in $\beta(u,v)$ is also intractable under the CDH assumption.

Argument 3.

1. Let us consider the computability of $\overrightarrow{l_u^{(k)}} = g^{r_u^{(k)}} \in \beta(u,v)$ while using $\beta(u,x)$ as input.
2. H, being a irreversible cryptographical hash function, $c_u^{(k+1)} = \tau_u^{(k)} H\left(g^{\pi_u r_u^{(k)}}\right) \in \beta(u,x)$ does not provide any advantage for computing $g^{r_u^{(k)}}$.
3. It is possible to compute $g^{r_u^{(k)}}$ from $H(g^{\pi_u r_x^{(j)}}) g^{r_u^{(k)}} \in \beta(u,x)$ only if $H(g^{\pi_u r_x^{(j)}})$ is known.
4. But in Argument 2 we have shown that computing $H(g^{\pi_u r_x^{(j)}})$ is intractable under the CDH assumption as π_u and $r_x^{(j)}$ are unknown exponents.

Argument 4.

1. Let us consider $l_u^{(k)*} = H(g^{w\pi_u r_u^{(k)}} g^{\pi_u}) \in \beta(u,v)$ in the context of Lemma 2.
2. We substitute unknown exponents $a = \pi_u,\ r = r_u^{(k)}$ and known $g^a = g^{\pi_u}, g^r = g^{r_u^{(k)}}$.
3. Computing $H(g^{w\pi_u r_u^{(k)}} g^{\pi_u})$ is intractable under the CDH assumption.

Argument 5.

1. The adversary can compute $\overleftarrow{l_{\widehat{v}}^{(k)}} = H(g^{\pi_v r_u^{(k)}}) g^{r_v^{(k-1)}} \in \beta(u,v)$ using the knowledge of $\pi_v, g^{r_u^{(k)}}, g^{r_v^{(k-1)}}$.
2. But $\pi_v \nexists \beta(u,x),\ g^{r_u^{(k)}} \nexists \beta(u,x),\ g^{r_v^{(k-1)}} \nexists \beta(u,x)$.
3. Hence, even though the adversary can compute $\overleftarrow{l_{\widehat{v}}^{(k)}}$, knowledge about $\beta(u,x)$ does not provide any advantage.

Hence, computing any of the terms in $\beta(u,v)$ from $\beta(u,x)$ is intractable under the CDH assumption, while $\pi_u, r_u^{(k)}, r_x^{(j)}$ are unknown exponents.

We have previously noted that terms other than $\beta(u,v)$ and $\beta(u,x)$ are not computed using $r_u^{(k)}$, which is an unknown exponent. Even though there are terms in $\tau_u^{(k+1)}$ that are computed using $r_x^{(j)}$, due to DLP assumption, it is intractable to compute $r_x^{(j)}$ from $g^{r_x^{(j)}}$ or $g^{ar_x^{(j)}}$.

$\therefore$, by combining these five arguments, we conclude that due to $\pi_u, r_u^{(k)}, r_x^{(j)}$ being unknown exponents, it is intractable to compute any term of $\tau_u^{(k)}$ using a subset of terms in $\tau_u^{(k+1)}$ and vice versa. □

Theorem 3. *Given g^{π_v}, π_u and two blocks $\tau_{\widehat{v}}^{(k)}, \tau^{(n)} = \tau_{\widehat{v}}^{(k+1)}$, computing any term of one block from a subset of terms in another block is as difficult as the CDH problem.*

Proof. We prove this theorem in a similar way. First, we group the terms of the first and the second block that are computed using $r_v^{(k)}$ as $\beta(u,v)$ and $\beta(y,v)$.

$$\begin{aligned} \beta(u,v) &= \{\overrightarrow{l_{\widehat{v}}^{(k)}}, l_{vw}^{(k)}, \overleftarrow{l_u^{(k)}}\} && = \{g^{r_v^{(k)}}, H(g^{wr_v^{(k)}})g^{h\pi_v r_v^{(k)}}, H(g^{\pi_u r_v^{(k)}})g^{r_u^{(k-1)}}\} \\ \beta(y,v) &= \{c_{\widehat{v}}^{(n)}, \overleftarrow{l_{\widehat{v}}^{(n)}}\} && = \{\tau_{\widehat{v}}^{(k)} H\left(g^{\pi_v r_v^{(k)}}\right), H(g^{\pi_v r_u^{(k+1)}})g^{r_v^{(k)}}\} \end{aligned}$$

By swapping u and v, we show that all terms in $\beta(y,v)$ are equivalent to $\beta(u,x)$. Hence, given the unknowns $\pi_v, r_v^{(k)}, r_u^{(k+1)}$, computing any of the terms in $\beta(y,v)$ from $\beta(u,v)$ is intractable under the CDH assumption.

$\therefore$, computing any term of $\beta(y,v)$ using at least one term of $\beta(u,v)$ is intractable under the CDH assumption.

We now present our arguments to show that the computability of the terms in $\beta(u,v)$, from a subset of terms in $\beta(y,v)$, is also intractable under the CDH assumption.

Let us consider that an adversary $\mathcal{A}$ can compute $l_{vw}^{(k)} = H(g^{wr_v^{(k)}})g^{h\pi_v r_v^{(k)}}$ from $g^{\pi_v}, g^{r_v^{(k)}}$, g^h using oracle $\mathcal{B}$ while $h, w, \pi_u, r_v^{(k)}$ are unknown exponents.

$$\mathcal{B}(g, g^{\pi_v}, g^{r_v^{(k)}}, g^w, g^h) = H(g^{wr_v^{(k)}})g^{h\pi_v r_v^{(k)}}$$

$\mathcal{A}$ can construct algorithm C using $\mathcal{B}$ as shown below.

$$\begin{aligned} C &\Rightarrow \mathcal{B}(g, g^{\pi_v}, g, g, g^h) && = H(g)g^{h\pi_v} \\ C(g, g^{\pi_v}, g^h) &\Rightarrow \mathcal{B}(g, g^{\pi_v}, g, g, g^h)/H(g) && = g^{h\pi_v} \end{aligned}$$

However, π_v, h are unknowns and not present in $\beta(y,v)$. Additionally, g^h is also unknown to the adversary. Hence, computing $l_{vw}^{(k)}$ from $\beta(y,v)$ is intractable under the CDH assumption.

The rest of the terms in $\beta(u,v)$ are equivalent to the terms discussed in the previous proof. Hence, it is also infeasible to compute them from $\beta(y,v)$ under the CDH assumption. Therefore, computing any of the terms in $\beta(u,v)$ from $\beta(y,v)$ is intractable under the CDH assumption, while $\pi_v, r_v^{(k)}, r_u^{(k+1)}$ are unknown exponents.

$\therefore$, we conclude that due to $\pi_u, r_v^{(k)}, r_u^{(k+1)}$ being unknown exponents, it is intractable to compute any term of $\tau_{\widehat{v}}^{(k)}$ using a subset of terms in $\tau_{\widehat{v}}^{(k+1)}$ and vice versa. □

Theorem 4. *Given g^{π_u} and two blocks $\tau_u^{(k)}\tau^{(n)}$, deciding whether $\tau^{(n)} = \tau_u^{(k+1)}$ is infeasible.*

Proof. Previously, while proving Theorem 2, we have grouped the terms of $\tau_u^{(k)}$ and $\tau_u^{(k+1)}$ that are computed using $r_u^{(k)}$ as $\beta(u,v)$ and $\beta(u,x)$. To prove this theorem, we re-introduce these terms.

$$\beta(u,v) = \{\overrightarrow{l_u^{(k)}}, l_u^{(k)*}, \overleftarrow{l_{\widehat{v}}^{(k)}}\} \qquad = \{g^{r_u^{(k)}}, H\left(g^{w\pi_u r_u^{(k)}} g^{\pi_u}\right), H(g^{\pi_v r_u^{(k)}})g^{r_v^{(k-1)}}\}$$
$$\beta(u,x) = \{c_u^{(k+1)}, \overleftarrow{l_u^{(k+1)}}\} \qquad = \{\tau_u^{(k)} H\left(g^{\pi_u r_u^{(k)}}\right), H(g^{\pi_u r_x^{(j)}})g^{r_u^{(k)}}\}$$

In Theorem 2, we prove that computing any term of $\beta(u,v)$ using one or more terms of $\beta(u,x)$, and vice versa, is intractable under the CDH assumption. Here, we show that given $\beta(u,v)$, any term in $\beta(u,x)$ is indistinguishable from the same terms of another block and vice versa.

Under the assumptions of the Random Oracle model, $H(x_1)$ is indistinguishable from $H(x_2)$. Additionally, the inputs and outputs of the hash function H cannot be correlated by an adversary. Hence, we can rewrite these above-mentioned terms by substituting the outputs of the hash functions as H_1, H_2, H_3, H_4, as mentioned below.

$$\beta(u,v) = \{\overrightarrow{l_u^{(k)}}, l_u^{(k)*}, \overleftarrow{l_{\widehat{v}}^{(k)}}\} \qquad = \{g^{r_u^{(k)}}, H_1, H_2 g^{r_v^{(k-1)}}\}$$
$$\beta(u,x) = \{c_u^{(k+1)}, \overleftarrow{l_u^{(k+1)}}\} \qquad = \{\tau_u^{(k)} H_3, H_4 g^{r_u^{(k)}}\}$$

Now, H_1, being indistinguishable, cannot provide any advantage to the adversary for distinguishing one term of $\beta(u,x)$ from the same term of another block. Additionally, we note that, now, the $\beta(u,x)$ does not involve any term related to the passive user A_x. Therefore, H_1, x becomes irrelevant.

We consider $\beta(u)'$ to be a set of the same terms from the $t'th$ block in which the same user, A_u, was active. Hence, the terms in $\beta(u)'$ are composed of $r_u^{(t)} \in_R Z_p$ (following our notation), which is randomly chosen and cannot be correlated with $r_u^{(k)}$ or π_u. The challenge for the adversary is to gain a non-negligible advantage in distinguishing between two sets $\beta(u), \beta(u)'$, given $\beta(u,v), g^{\pi_u}$. These sets are shown below.

$$\beta(u,v) = \{\overrightarrow{l_u^{(k)}}, l_u^{(k)*}, \overleftarrow{l_{\widehat{v}}^{(k)}}\} \qquad = \{g^{r_u^{(k)}}, H_2 g^{r_v^{(k-1)}}\}$$
$$\beta(u) = \{c_u^{(k+1)}, \overleftarrow{l_u^{(k+1)}}\} \qquad = \{H_3 \tau_u^{(k)}, H_4 g^{r_u^{(k)}}\}$$
$$\beta(u)' = \{c_u^{(t)}, \overleftarrow{l_u^{(t)}}\} \qquad = \{H_3' \tau_u^{(t)}, H_4' g^{r_u^{(t)}}\}$$

$\forall\, g^a, g^b\ \exists c \in Z_p$ such that $g^a g^c = g^{a+c} = g^b$. Hence, $\forall\, t, \exists m$ such that $H_4' g^{r_u^{(t)}} = (H_4' g^m) g^{r_u^{(k)}}$. However, given g, g^x it is infeasible to distinguish between $\alpha g^x, \beta g^x$ if α, β are unknowns.

Therefore, given $g, g^{r_u^{(k)}}$ it is infeasible to distinguish between $H_4 g^{r_u^{(k)}}$ and $H_4' g^{r_u^{(t)}}$ when $r_u^{(k)}, r_u^{(t)}$ are unknowns, H_4, H_4' are also unknowns, and outputs of H are a hash function following the Random Oracle model.

Similarly, given $\beta(u)$, it is infeasible to distinguish between $\beta(u,v)$ and another set $\beta(u,v)' = \{g^{r_u^{(t)}}, H_2' g^{r_v^{(m-1)}}\}$ from another block.

$\therefore$ Due to the properties of the Random Oracle Hash function and unfeasibility of computation of the inputs of H, given two blocks $\tau_u^{(k)} \tau^{(n)}$, it is infeasible to decide whether $\tau^{(n)} = \tau_u^{(k+1)}$. □

Theorem 5. *Given g^{π_v} and two blocks $\tau_{\widehat{v}}^{(k)}, \tau^{(n)}$, deciding whether $\tau^{(n)} = \tau_{\widehat{v}}^{(k+1)}$ is infeasible.*

Proof. Previously, while proving Theorem 3, we have grouped the terms of $\tau_v^{(k)}$ and $\tau_v^{(k+1)}$ that are computed using $r_v^{(k)}$ as $\beta(u,v)$ and $\beta(y,v)$. To prove this theorem, we re-introduce these terms.

$$\beta(u,v) = \{\overrightarrow{l_{\widehat{v}}^{(k)}}, l_{vw}^{(k)}, \overleftarrow{l_u^{(k)}}\} \qquad = \{g^{r_v^{(k)}}, H(g^{w r_v^{(k)}}) g^{h\pi_v r_v^{(k)}}, H(g^{\pi_u r_v^{(k)}}) g^{r_u^{(k-1)}}\}$$
$$\beta(y,v) = \{c_{\widehat{v}}^{(n)}, \overleftarrow{l_{\widehat{v}}^{(n)}}\} \qquad = \{\tau_{\widehat{v}}^{(k)} H\left(g^{\pi_v r_v^{(k)}}\right), H(g^{\pi_v r_u^{(k+1)}}) g^{r_v^{(k)}}\}$$

By following the same approach as Theorem 4, we substitute the outputs of H.

$$\beta(u,v) = \{\overrightarrow{l_{\widehat{v}}^{(k)}}, l_{vw}^{(k)}, \overleftarrow{l_u^{(k)}}\} \qquad = \{g^{r_v^{(k)}}, H_1 g^{h\pi_v r_v^{(k)}}, H_2 g^{r_u^{(k-1)}}\}$$
$$\beta(y,v) = \{c_{\widehat{v}}^{(n)}, \overleftarrow{l_{\widehat{v}}^{(n)}}\} \qquad = \{\tau_{\widehat{v}}^{(k)} H_3, H_4 g^{r_v^{(k)}}\}$$

If we swap u and v, then all terms except $H_1 g^{h\pi_v r_v^{(k)}}$ in $\beta(u,v)$ and $\beta(y,v)$ become the same as the terms in $\beta(u,v), \beta(u,x)$ we have previously discussed while proving Theorem 4. In $l_{vw}^{(k)} = H_1 g^{h\pi_v r_v^{(k)}}$, the hash value H_1 is an output of H, which satisfies the Random Oracle model. Hence, given $H_4 g^{r_v^{(k)}}$, it is infeasible to distinguish between $H_1 g^{h\pi_v r_v^{(k)}}$ from any $H_m g^{h\pi_v r_x^{(t)}}$ when H_m is output of the Random Oracle and $r_x^{(t)}$ is a random number.

$\therefore$, due to the properties of the Random Oracle Hash function and unfeasibility of computation of the inputs of H, given two blocks $\tau_{\widehat{v}}^{(k)}, \tau^{(n)}$ it is infeasible to decide whether $\tau^{(n)} = \tau_{\widehat{v}}^{(k+1)}$. □

Theorem 6. *Given a non-genesis block $\tau^{(n)}$ and a public key of a user, the problem of determining whether that user is involved in this block or not is infeasible.*

Proof. The adversary needs a decisional oracle $\mathcal{A}$ such that $\mathcal{A}(g^{\pi_t}, X) \neq \mathcal{A}(g^{\pi_r}, X)\ \forall r \neq t$ where $X \subset \tau^{(n)}$. However, previously, we proved in Theorems 4 and 5 that such a decisional oracle is infeasible unless a computational oracle $\mathcal{B}$ exists that can solve the computational version of the same problem. In the Theorems 2 and 3, we proved that the computational oracle is as difficult as the CDH problem. Hence, determining whether a user is involved in a given block as an active or passive user is as difficult as CDH. □

Theorem 7. *Given that all blocks in the blockchain and a public key g^x are associated with a user, it is not possible to partition the blocks such that one subset contains all the blocks associated with that user.*

Proof. We have already proven that given any random block, deciding whether a user A_x is involved in that block using the public key g^x is infeasible. We have also proven that even if it is known that a block is associated with user A_x, the problem of deciding whether any random block is participated in by the same user is infeasible. Hence, even if the genesis blocks of all users are known to decide the association of that user with any other block is also infeasible. □

Theorem 8. *Given a random block, its content cannot be read without compromising the active or the passive user involved in the block or any supervisor.*

Proof. To decrypt the contents of a block, the adversary must interpolate a linear equation and find out its value for a given x value. However, only one coordinate is visible to the adversary. To retrieve the other coordinate, the adversary needs to compute $g^{\pi_u r_u^{(n)}}$ or $g^{\pi_v r_u^{(n+1)}}$ as shown in Equation (29), which is as difficult as the CDH. Also, under the Random Model assumption, to compute $H_2(g^{\pi_u r_u^{(n)}})$ from $H(g^{\pi_u r_u^{(n)}})$ or to compute $H_2(g^{\pi_v r_u^{(n+1)}})$ from $H(g^{\pi_v r_u^{(n+1)}})$ is considered to be intractable, because both H, H_2 are different cryptographic hash functions.

The adversary can also retrieve the secret by computing $g^{(\theta+\phi)w\gamma_x}$, as shown in Equation (32). However, neither of $g^\theta, g^\phi, g^w, g^{\theta w}, g^{\phi w}$ is known to the adversary because we consider that the adversary has not compromised a supervisor or the TS. Hence, to construct that is also intractable. □

Theorem 9. *Custodian A_t, who knows that $\tau_s^{(n)}$ is the last block in which user A_s was active, cannot construct an access block in which A_s is active without knowing π_s unless the trusted server is compromised.*

Proof. To construct an access block, the custodian needs to communicate with the trusted server and send $(\tau_s^{(n)}, g^{\pi_s r_s^{(n)}}, g^{\pi_s})$. Although the values of $\tau_s^{(n)}$ and g^{π_s} are known by A_t, the value of $g^{\pi_s r_s^{(n)}}$ must be computed. As A_t knows $\tau_s^{(n)}$, the value of $g^{r_s^{(n)}}$ is also known. However, to compute $g^{\pi_s r_s^{(n)}}$ from $g^{\pi_s}, g^{r_s^{(n)}}$ is as difficult as solving the CDH problem. Now, let us consider the situation that A_t sends $g^{x r_s^{(n)}}$ to the trusted server where $x \neq \pi_s$. The trusted server computes $H\left(g^{x r_s^{(n)}} w g^{\pi_s}\right)$ as shown in Equation (20). This value is compared with $l_s^{(n)*} = H\left(g^{\pi_s r_s^{(n)}} w g^{\pi_s}\right)$. These two values do not match, and the trusted server is trusted to follow the protocol. Hence, unless the trusted server is compromised, the adversary custodian A_t cannot impersonate A_s and construct an access block in which A_s is active. □

5.2. Experimental Evaluation

In Table 3, we have summarized the number of different types of operations that we need to perform for active and passive traversals over the ledger. We have shown that the computational overhead of traversal is deterministic. Now, we check the performance of the system while performing different actions, including traversals.

For testing our proposed model, we have used the Crypto++ library to implement cryptographic functions in C++. A TCP server using the Boost Asio library was developed to represent the trusted server. Additionally, a TCP client was created to manage user private-key operations, as outlined in Table 2. We chose a PostgreSQL database for storing HD and employed a key-value store for both an event log and indexing purposes. To evaluate the performance of active or passive traversal, a reader application was developed that accesses the key-value store using the user's private key. The architecture of the implementation is shown in Figure 8. The source code of the implementation is available online https://github.com/neel/cbtl, accessed on 3 December 2024. We initialize the system with five managers, four supervisors, seven patients, and their genesis records in the database (We assume that the private keys are securely delivered to the users, as we do not focus on the key-distribution problem).

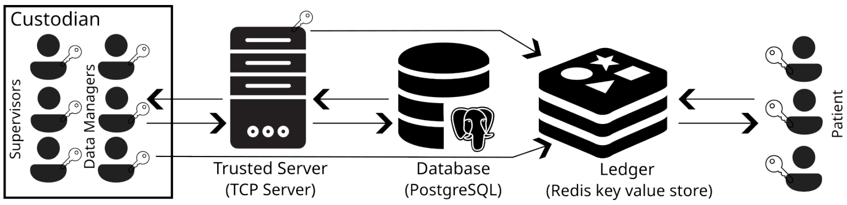


Figure 8. Architecture of the implementation.

5.2.1. Insertion Overhead

In the first experiment, we assess the performance of bulk insertion to assess the influence of the following factors on the performance of insertion.

1. Number of records already associated with the patient.
2. Total number of records in the database.
3. Number of access events in which the data manager was active.

In this experiment, we simulate the initial insertion of medical records by various data managers into a patient database. Each data manager, denoted as M_i, inserts a specific number of records for a designated patient P_i. For instance, M_0 inserts 10 records for patient P_0, M_1 adds 20 records for P_1, and similarly, M_4 adds 50 records for P_4. This is referred to as "incremental load", which is depicted using blue columns in Figure 9. We observe that CPU time consumed for insertion is proportional to the number of records inserted. Prior to this experiment, there were no existing records or previous access events by these five data managers, ensuring that past activities do not influence the results of this experiment. It is also important to note that this represents each manager's first insertion of data into the system. At the end of this procedure, the ledger will document a single event for each manager, highlighting their role as active users during their respective insertions.

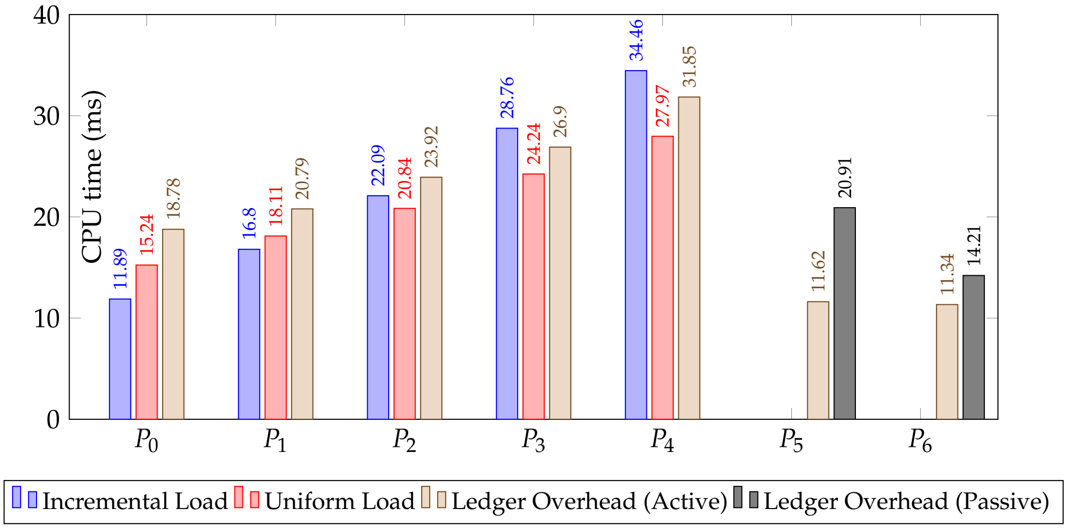


Figure 9. Insertion overhead by scenario. The overhead varies based on the number of existing records and access events associated with a passive user but not on the number of events tied to the active user performing the insertion.

The experiment is then repeated, with each data manager inserting an identical number of records (10 records), allowing us to observe performance under uniform input conditions. This case is referred to as "uniform load", which is depicted using red columns in Figure 9. We observe that CPU time consumed for insertion increases even though the number of records being inserted is identical. This shows that the overhead of insertion depends on the number of records that are already associated with that patient.

Furthermore, the same manager (M_0) inserts an identical number of records (10 records) for all patients. This allows us to observe the influence of existing access events associated with an active user on the overhead of the insertion operation. This case is referred to as "ledger overhead (active)", which is depicted using brown columns in Figure 9. We observe that the overhead of insertion of records associated with P_5 and P_6 is identical even though the number of access events associated with M_0 is not the same before this insert operation. Hence, the overhead of insertion is not influenced by the number of existing access events associated with the active user performing the action but by the existing records associated with that patient.

Lastly, we explore the influence of the number of past access events in which a patient was a passive participant. We already have 10 records associated with P_5 and P_6 corresponding to 1 access event related to each of these patients. We now perform the insertion of another 10 records by manager M_0, which implies that there will be a total of 20 records associated with P_6. Then, we perform 10 insertion requests, each consisting of a single record by the same manager M_0. At the end of this, we have 20 records associated with both P_5 and P_6 but 20 access events associated with P_5 while there are only 2 access events associated with P_6. Now we perform the insertion of one record for P_5 and one for P_6. This case is referred to as "ledger overhead (passive)", which is depicted using brown columns in Figure 9. It is observed that although there is the same number of records associated with both P_5 and P_6, the insertion overhead of P_6 is less than P_5. This is an expected behavior because the TS performs passive traversal in order to find the last block associated with the access event. This process is recursive, leading to the differences in our observation. This overhead can be eliminated by storing the last block of all passive users in a cache. However, that cache needs to be secured against adversarial access.

5.2.2. Retrieval Overhead

Following several insertion operations in the previous experiment, we now compile a table (shown in Table 4) detailing the number of records associated with each data manager and patient pair. In addition, each patient is linked to one genesis record, bringing the total to 299 records in our database. We also summarize the number of events associated with different managers and patients in Table 5. Additionally, we have 16 genesis events (5 managers, 4 supervisors, 7 patients) totaling 46 events in our ledger. To ensure accuracy, we perform a sanity check on our database and the Redis key-value store to verify that the total number of records and event blocks match our expectations.

Table 4. Record Insertion Summary.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	**Total**
M_0	30	10	10	10	10	21	21	112
M_1	0	30	0	0	0	0	0	30
M_2	0	0	40	0	0	0	0	40
M_3	0	0	0	50	0	0	0	50
M_4	0	0	0	0	60	0	0	60
Total	30	40	50	60	70	21	21	292

We then measure the computational overhead of retrieving these records. The data managers, M_0 and M_1, represent the extremes in terms of event activity, having executed the most and the least events, respectively. Our retrieval algorithm is independent of the retriever. Thus, we anticipate identical computational overhead, regardless of which data manager retrieves the records of a patient. The retrieval overhead for operations performed by M_0 and M_1 across all patients is illustrated in Figure 10. The results clearly demonstrate that the number of events in which the active user was previously involved does not influence the overhead of retrieval. Instead, it solely depends on the number of records associated with a patient.

Table 5. Access event summary.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	**Total**
M_0	3	1	1	1	1	12	3	22
M_1	0	2	0	0	0	0	0	2
M_2	0	0	2	0	0	0	0	2
M_3	0	0	0	2	0	0	0	2
M_4	0	0	0	0	2	0	0	2
Total	3	3	3	3	3	12	3	30

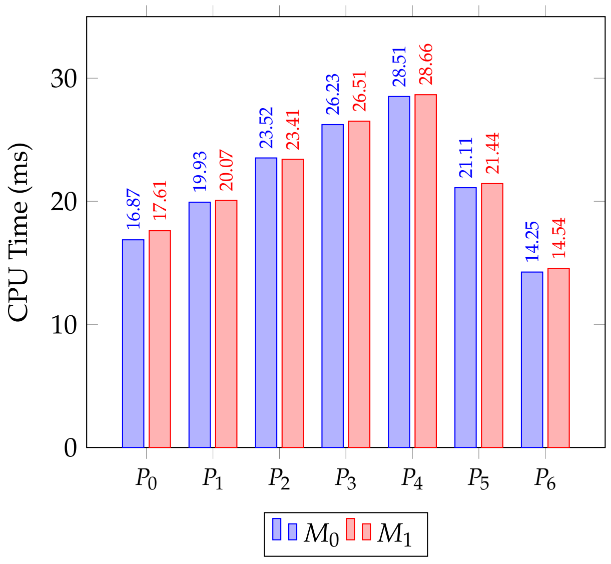


Figure 10. Retrieval overhead depends on the number of records associated with a patient only.

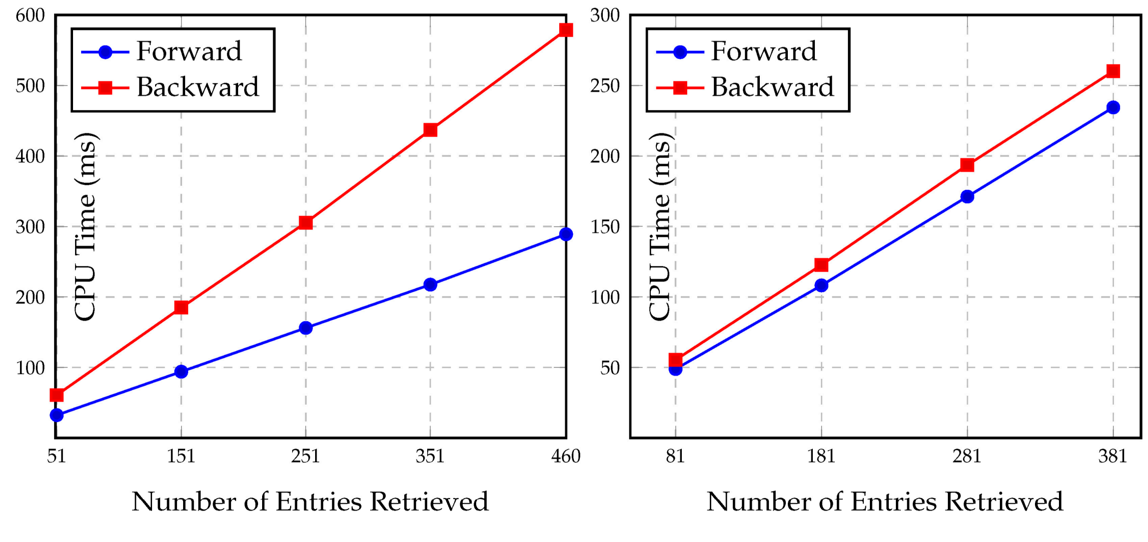
5.2.3. Traversal Overhead

Following the last experiment, our ledger now includes 7 entries for each of the data managers, M_0 and M_1, totaling 14 entries. These entries also involve the 7 patients as passive users. Currently, our ledger comprises 60 blocks. We have previously noted that the ledger traversal is an offline process that does not require communication with the trusted server (TS). Our traversal program directly interacts with the Redis key-value store to perform these traversals.

We conduct both active and passive traversals through our ledger in forward and backward directions. We have previously shown that the computational overhead of backward traversal is more than forward traversal in Table 3. This experiment reconfirms the expected behavior: the CPU time taken for active forward traversal by M_0 through 30 blocks is 20.635 ms, whereas active backward traversal for the same took 44.774 ms. Similarly, passive forward traversal by patient P_5 through 15 blocks required 8.884 ms, while passive backward traversal took 10.31 ms.

In order to perform this experiment on a larger dataset, we do not necessarily need a large number of records. Instead, we need a large number of blocks in our ledger. We, therefore, execute a large number of retrieval operations involving a randomly selected data manager and patient, which results in 2000 access events. We then select the data manager and patient with the highest number of events for further active and passive traversal experiments to measure computational overhead. The results are shown in Figure 11. In

Figure 11a, it is observed that active backward traversal is significantly slower than active forward traversal. Figure 11b reveals that, although passive backward traversal is also slower than passive forward traversal, the difference is not as pronounced as in active traversal. However, both cases confirm that backward traversal consistently incurs higher computational overhead than forward traversal.



(a) Computational overhead of Active Traversal **(b)** Computational overhead of Passive Traversal

Figure 11. Computational overhead of ledger traversal.

6. Discussion

In the previous section, we assessed the efficiency and performance of the proposed system, demonstrating its potential for real-world applications. Now, we present a summary of our findings and interpretations from Section 5. In Section 5.1, we have provided theoretical evidence to show that our proposed approach is secure against adversarial access to the storage and the ledger while considering adversaries incapable of solving CDH and DDH problems as defined in Definitions 1 and 2. First, we have shown that the storage system is secure unless an adversary has compromised a custodian while also compromising the trusted server at the same time. We have also shown that the ledger is secure against adversarial knowledge gains from malicious actors inside or outside the system. For this, we first show in Theorem 2 that the passive users cannot perform active traversal without compromising the target data manager. Then, we show in Theorem 3 that the active users cannot perform passive traversal without compromising the target patient. Based on these two theorems, we show in Theorem 7 that it is difficult to partition the blocks into subsets associated with a specific actor without gaining access to the credentials of that specific actor. Hence, given the ledger, an adversary cannot find the set of blocks belonging to a patient or a data manager. The confidentiality of the contents of a block is also demonstrated to be safe in Theorem 8. We have also shown that with the proposed framework, the ledger is safe against impersonation attacks by the data managers in Theorem 9.

Section 5.2 demonstrates the system's ability to handle various operations like insertion and retrieval without significant performance degradation, even as the number of records grows. This scalability is vital in healthcare settings where data volume can increase rapidly. For the insertion of health records associated with a patient, our proposed system requires linear traversals across all records of that patient, contributing to the complexity of $\mathcal{O}(n)$, where n is the number of existing records associated with the patient. Consequently, the overhead for adding a record varies between patients and increases over time. Therefore, assessing the anticipated number of records per patient and their growth rate is crucial for practical implementation. Furthermore, for the creation of a new block in the ledger, the TS needs to find out the last block associated with the passive user who is not actively participating in the access event. In our implementation, the TS performs traversal from the genesis block to the last block associated with the passive user in order to fetch the ID of the last block. This leads to the complexity of $\mathcal{O}(n)$, where n is the number of existing blocks associated with that patient. This overhead can be mitigated if the IDs of the last block of each user are cached in server memory. However, storing additional information requires secured access, presenting a trade-off between safety and performance. A practical application of our proposed system must carefully evaluate these design choices to balance efficiency and security effectively.

However, the system's reliance on a centralized trusted server (TS) poses potential risks, such as single points of failure or targeted attacks. While the paper mentions the use of a semi-honest model for the TS, suggesting that it operates correctly under normal circumstances but could be curious or malicious, this model may not sufficiently mitigate all potential internal threats. Future enhancements could include decentralized approaches or additional checks within the system to further reduce these risks. Additionally, in this paper, we do not discuss adding or removing a new data manager or a new supervisor, which could be necessary for real-world implementation. Therefore, in the next section, we discuss the limitations of our proposed approach.

6.1. Limitations

Despite the advancements offered by the proposed system, there are inherent limitations that could affect its effectiveness and security. These limitations are crucial for understanding the potential vulnerabilities and areas for future improvement. Here, we outline some of the primary challenges and limitations that warrant consideration.

Re-Identification Attacks: In Theorem 1, we have shown that adversarial access to sensitive information may not lead to the identification of the patient unless the adversary can solve the CDH problem. However, this work does not focus on a re-identification attack, which is possible for a polynomial adversary. There have been various works on protection against such attacks, which can be employed along with this work in order to improve the security of the system. The existing solutions that have been proposed to solve re-identification attacks include but are not limited to the anonymization and encryption of sensitive information. Anonymization, as discussed in Section 2, may lead to the degradation of data quality, which may be problematic for the internal researchers of healthcare registries such as cancer registries. On the other hand, the encryption of sensitive information adds computational overheads that internal researchers may encounter while decrypting. Moreover, the effectiveness of techniques such as encryption depends heavily on the overall design and integration of the system.

Interaction with Manipulated TS: In Theorem 1, we demonstrate that compromising the trusted server (TS) does not facilitate the adversarial identification of sensitive information, as secrets θ or g^{θ} are not stored but recomputed during each interaction with an active user, according to the RSI protocol described in Section 4.3. However, this work does not account for a scenario where the TS is replaced by a malicious version that retains the recomputed secrets for malicious purposes because this violates the initial assumptions on the trusted server. Such an event would compromise the trustworthiness of the TS, posing a significant security risk. Potential attack vectors could include DNS spoofing, installation of backdoors, or exploitation by advanced persistent threats (APTs). This work does not address this limitation; future research should explore the integration of additional defensive measures against such attacks.

Quantum Adversary: Although we have demonstrated the strength of our proposed scheme in Section 5.1.3, the entries in the ledger are not immune from adversarial attacks from a quantum adversary. Because our paper depends on the assumptions of Diffie–Hellman, a quantum computer can efficiently compute discrete logarithms, thus breaking that assumption. However, there have been many recent works on post-quantum cryptography, which can be incorporated for long-term protection against such strong adversaries.

7. Conclusions

In this paper, we have proposed a secure, responsible, privacy-preserving document-storage and -retrieval technique. The solution can be used for different business processes other than healthcare. Even in healthcare, the applications of such systems are not limited to health registries. However, a generalized use case may involve more than two users in one event.

Our proposed solution adds a constant time-storage overhead with each medical record, which is three integer values of encryption bit length. Additionally, each event block consists of eight integers for traversability, three of which are for the active user and another three for the passive user. The size of the signature and block ID may vary depending on the signature algorithm and platform usage. It also requires maintaining an offchain index that will have two entries for each block. Future research works may investigate whether it is possible to accomplish the same objective with less storage overhead.

Our proposed system considers the trusted server as the only writer, whereas the other users of the system are readers. However, if multiple registries participate in the system, then there will be multiple writers. The only mechanism of verification in our proposed system is a checksum and signature verification. The traversability of the blocks in the blockchain cannot be verified by any entity because then that entity will also be able to partition the blockchain into blocks associated with one particular user or the other. Also, the verifier will be able to understand the sequence of events associated with any user inside the system. This prompts another important problem that future work may address.

Author Contributions: Conceptualization, S.B. and D.M.; methodology, S.B.; formal analysis, S.B.; writing and editing, S.B. and D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research Council of Norway, grant number 288106.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: This article is a revised and expanded version of a paper entitled Secure Traversable Event Logging for Responsible Identification of Vertically Partitioned Health Data, which was presented at TrustCom, Exeter, UK, and 2023. This work is supported by the Research Council of Norway, grant number 288106. We also acknowledge Jan F. Nygård from the Cancer Registry of Norway (CRN) for discussing the internal operations of CRN regarding this problem.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chatterjee , S.; Chattopadhyay, A.; Senapati, S.N.; Samanta, D.R.; Elliott, L.; Loomis, D.; Mery, L.; Panigrahi, P. Cancer registration in India—Current scenario and future perspectives. *Asian Pac. J. Cancer Prev.* **2016**, *17*, 3687–3696. Available online: https://pubmed.ncbi.nlm.nih.gov/27644602/ (accessed on 8 December 2024).
2. Laugesen, K.; Ludvigsson, J.F.; Schmidt, M.; Gissler, M.; Valdimarsdottir, U.A.; Lunde, A.; Sørensen, H.T. Nordic health registry-based research: A review of health care systems and key registries. *Clin. Epidemiol.* **2021**, *13*, 533–554. [CrossRef]
3. Bouchardy, C.; Rapiti, E.; Benhamou, S. Cancer registries can provide evidence-based data to improve quality of care and prevent cancer deaths. *Ecancermedicalscience* **2014**, *8*, 413. [CrossRef]
4. Pukkala, E.; Engholm, G.; Højsgaard Schmidt, L.K.; Storm, H.; Khan, S.; Lambe, M.; Pettersson, D.; Ólafsdóttir, E.; Tryggvadóttir, L.; Hakanen, T.; et al. Nordic Cancer Registries–an overview of their procedures and data comparability. *Acta Oncol.* **2018**, *57*, 440–455. [CrossRef] [PubMed]

5. Chaudhry, K.; Luthra, U.K. Cancer Registration in India. *Cancer* **2002** , 14–26. Available online: https://mohfw.gov.in/sites/default/files/Cancer%20Registration%20In%20India.pdf (accessed on 8 December 2024).
6. Bose, S.; Marijan, D. Secure Traversable Event logging for Responsible Identification of Vertically Partitioned Health Data. In Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Exeter, UK, 1–3 November 2023; IEEE: Piscataway, NJ, USA, 2023. [CrossRef]
7. Bose, S.; Marijan, D. A Survey on Privacy of Health Data Lifecycle: A Taxonomy, Review, and Future Directions. *arXiv* **2023**. . [CrossRef]
8. Xia, Q.; Sifah, E.B.; Asamoah, K.O.; Gao, J.; Du, X.; Guizani, M. MeDShare: Trust-less Medical Data Sharing Among. *IEEE Access* **2017**, *5*, 14757–14767. [CrossRef]
9. Gajanayake, R.; Iannella, R.; Sahama, T. Privacy oriented access control for electronic health records. *Electron. J. Health Inform.* **2014**, *8* , e15. Available online: https://eprints.qut.edu.au/63620/ (accessed on 8 December 2024).
10. Thummavet, P.; Vasupongayya, S. A novel personal health record system for handling emergency situations. In Proceedings of the 2013 International Computer Science and Engineering Conference, ICSEC 2013, Nakhonpathom, Thailand, 4–6 September 2013; pp. 266–271. [CrossRef]
11. Jose, J.T.; Anju, S. Threshold Cryptography Based Secure Access Control for Electronic Medical Record in an Intensive Care Unit. *Int. J. Eng. Res. Technol. (IJERT)* **2013**, *2*, 457–464.
12. Eskeland, S.; Oleshchuk, V.A. *EPR Access Authorization of Medical Teams Based on Patient Consent*; Gesellschaft für Informatik e. V: Bonn, Germany, 2007; Volume P-118, pp. 11–21. Available online: https://subs.emis.de/LNI/Proceedings/Proceedings118/article1928.html (accessed on 8 December 2024).
13. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 11. [CrossRef]
14. Yuliana, M.; Darwito, H.A.; Sudarsono, A.; Yofie, G. Privacy and security of sharing referral medical record for health care system. In Proceedings of the Proceeding—2016 2nd International Conference on Science in Information Technology, ICSITech 2016: Information Science for Green Society and Environment, Balikpapan, Indonesia, 26–27 October 2016; pp. 232–237. [CrossRef]
15. Sudarsono, A.; Yuliana, M.; Darwito, H.A. A secure data sharing using identity-based encryption scheme for e-healthcare system. In Proceedings of the Proceeding—2017 3rd International Conference on Science in Information Technology: Theory and Application of IT for Education, Industry and Society in Big Data Era, ICSITech 2017, Bandung, Indonesia, 25–26 October 2017; pp. 429–434. [CrossRef]
16. Liu, J.; Li, X.; Ye, L.; Zhang, H.; Du, X.; Guizani, M. BPDS: A Blockchain Based Privacy-Preserving Data Sharing for Electronic Medical Records. In Proceedings of the 2018 IEEE Global Communications Conference, GLOBECOM 2018—Proceedings, Abu Dhabi, United Arab Emirates, 9–13 December 2018. [CrossRef]
17. Ge, C.; Yin, C.; Liu, Z.; Fang, L.; Zhu, J.; Ling, H. A privacy preserve big data analysis system for wearable wireless sensor network. *Comput. Secur.* **2020**, *96*, 101887. [CrossRef]
18. 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]
19. Domadiya, N.; Rao, U.P. Improving healthcare services using source anonymous scheme with privacy preserving distributed healthcare data collection and mining scheme with privacy preserving distributed healthcare data. *Computing* **2021**, *103*, 155–177. [CrossRef]
20. Li, H.; Guo, F.; Zhang, W.; Wang, J.; Xing, J. (a,k)-Anonymous Scheme for Privacy-Preserving Data Collection in IoT-based Healthcare Services Systems. *J. Med. Syst.* **2018**, *42*, 56. [CrossRef] [PubMed]
21. Machanavajjhala, A.; Gehrke, J.; Kifer, D.; Venkitasubramaniam, M. L-diversity: Privacy beyond k-anonymity. In Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), Atlanta, GA, USA, 3–7 April 2006; IEEE: Piscataway, NJ, USA, 2006; p. 24. [CrossRef]
22. Li, N. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering, Istanbul, Turkey, 15 April 2006–20 April 2007; pp. 106–115. Available online: https://ieeexplore.ieee.org/document/4221659 (accessed on 8 December 2024).
23. Oh, S.R.; Seo, Y.D.; Lee, E.; Kim, Y.G. A comprehensive survey on security and privacy for electronic health data. *Int. J. Environ. Res. Public Health* **2021**, *18*, 9668. [CrossRef]
24. 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]
25. Tian, H.; He, J.; Ding, Y. Medical Data Management on Blockchain with Privacy. *J. Med. Syst.* **2019**, *43*, 26. [CrossRef] [PubMed]
26. Panko, R. Mobile App Usage Statistics 2018. Available online: https://themanifest.com/app-development/blog/mobile-app-usage-statistics (accessed on 8 December 2024).
27. Dennis, R.; Disso, J.P. An Analysis into the Scalability of Bitcoin and Ethereum. In *Proceedings of the Third International Congress on Information and Communication Technology*; Yang, X.S., Sherratt, S., Dey, N., Joshi, A., Eds.; Springer: Singapore, 2019; pp. 619–627.
28. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
29. Biham, E.; Boneh, D.; Reingold, O. Breaking generalized Diffie-Hellman modulo a composite is no easier than factoring. *Inf. Process. Lett.* **1999**, *70*, 83–87. [CrossRef]

30. Bresson, E.; Chevassut, O.; Pointcheval, D. Provably Authenticated Group Diffie-Hellman Key Exchange—The Dynamic Case. In *Advances in Cryptology—ASIACRYPT 2001*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 290–309. [CrossRef]
31. Al-Kuwari, S.; Davenport, J.H.; Bradford, R.J. Cryptographic Hash Functions: Recent Design Trends and Security Notions. *Cryptol. ePrint Arch.* **2011**. Paper 2011/565. Available online: https://eprint.iacr.org/2011/565 (accessed on 8 December 2024).