Implementation of Privacy-Preserving Identifiers for the Secure Storage of Electronic Health Records on the Ethereum Blockchain

SWATI KUMARI and HITESH TEWARI, Trinity College Dublin, Ireland

Patients and healthcare authorities frequently lack confidence in one another when it comes to the security of their medical records in healthcare settings. Particularly when it comes to patient data management, hospitals are infamous for having inadequate security and have long been the target of cyberattacks. Using blockchain technology to store medical records has drawbacks, including an excessive dependence on centralised cloud servers for key storage, privacy concerns, and the potential for attackers to deduce personal information about patients based on their blockchain activity. A system where patients have autonomy over their medical records and who can view them is a promising scenario. This paper provides a framework for indexing and securing a user's medical records, with emphasis placed on the healthcare setting using an Ethereum blockchain. The records are secured using biometric authentication and the patient's Personal Identifiable Information (PII). The patient can grant and revoke access to their records to individual healthcare authorities, and the Interplanetary Name System (IPNS) is used for off-chain record storage. The framework is modular and can be adapted for use in other environments, such as proof of ownership of tickets, and storing travel documents for verification by border control. A smart contract is used to store the hashes of the patient's iris scans on an Ethereum Virtual Machine (EVM) compatible blockchain. Privacy-preserving identifiers are used to anonymise the patient and where their records are stored on the blockchain. Our approach is to the best of our knowledge the only one that simultaneously offers encryption, anonymity, unlinkability and efficient off-chain storage. Additionally, our approach is the only approach we are aware of that provides record revocability

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**.

Additional Key Words and Phrases: blockchain, healthcare, identity, biometrics, hashing, revocability

ACM Reference Format:
Swati Kumari and Hitesh Tewari. 2023. Implementation of Privacy-Preserving Identifiers for the Secure Storage of Electronic Health Records on the Ethereum Blockchain. 1, 1 (June 2023), 26 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Hospitals are tasked with storing extensive amounts of sensitive patient information over extended periods. A critical challenge faced by the healthcare industry is security, as outdated and flawed systems often lead to data breaches. For instance, in 2021, the Irish Health Service Executive (HSE) experienced a severe data breach due to a ransomware attack [36]. This incident severely disrupted their systems, resulting in the cancellation of numerous outpatient appointments, the suspension of COVID-19 case reporting in Ireland, and the theft of thousands of patient medical records. Although the full impact on public health remains unclear, it likely had significant repercussions for patient care, including cancelled radiotherapy sessions for cancer patients and shortages of consultants. Research into applying blockchain technology to eHealth offers promising solutions by enhancing transparency regarding the storage and access of medical records, thereby improving patient confidence in healthcare systems.

Authors' address: Swati Kumari, kumaris@tcd.ie; Hitesh Tewari, htewari@tcd.ie, Trinity College Dublin, Ireland.

An Electronic Health Record (EHR) is a digital repository of health information for individual patients or populations. EHRs offer advantages over traditional paper-based records, such as greater accessibility, portability, and resistance to physical damage. However, most healthcare systems rely on custom in-house solutions for EHR storage. With the advent of low-cost and powerful cloud computing, these in-house solutions have become less appealing due to their high setup, maintenance, and operational costs. Nevertheless, patients often distrust cloud service providers for storing their EHRs [32].

Blockchain-based EHR storage systems present an alternative to centralized cloud servers. Such systems eliminate the need for hospitals to maintain in-house storage solutions while addressing patients' concerns about cloud servers controlling their EHRs. Furthermore, blockchain technology grants patients full control and autonomy over their EHRs, including deciding who can access them. This paper proposes using iris-based biometric authentication due to its superior accuracy, stability, minimal intrusion, and prior successful applications in blockchain-based vaccination passports [3, 8]. Biometric authentication offers a significant advantage over traditional mechanisms by eliminating the need to securely store a 256-bit private key. Password-based authentication is unsuitable due to its vulnerability to offline dictionary attacks [6].

Directly storing biometric data on the blockchain is impractical because it risks exposing personal information and involves large template sizes. Instead, raw iris data is hashed using a Locality-Sensitive Hashing (LSH) algorithm to reduce its size and obscure the original data. A public identifier links the patient to healthcare institutions for accessing EHRs via iris authentication, while a private identifier—accessible only by the patient—enables them to locate their records, decrypt them, and share them with selected institutions. Encrypted EHRs are stored off-chain on decentralized file storage networks such as the InterPlanetary File System (IPFS) or InterPlanetary Name System (IPNS).

1.1 Research Contribution

Our framework aims to secure a patient's EHRs in an eHealth setting. The framework is modular, and it can be used in many applications and settings. The components can be modified and interchanged as the state of technology develops over time. The framework consists of several components:

- Implemented iris biometric authentication integrated with Locality-Sensitive Hashing (LSH) algorithms (specifically S3Hash), enabling secure and efficient patient identification without storing sensitive raw biometric data on-chain. Demonstrated high accuracy and computational efficiency in iris matching, suitable for large-scale healthcare deployments.
- A smart contract deployed on an Ethereum Virtual Machine (EVM) compatible blockchain is used to store the hash of the iris scan and identifiers generated by the patient to index their EHRs.
- Introduced regenerable public and private identifiers derived from hashed biometric data combined with personal identifiable information (PII), eliminating the need for secure storage of private keys. Provided strong anonymity, unlinkability, and confidentiality guarantees through cryptographic hashing and key derivation functions.
- Proposed a novel revocation mechanism using IPNS off-chain storage, allowing patients to update, delete, or revoke access to their records dynamically without compromising data integrity or privacy. Demonstrated practical feasibility of revocation in decentralized storage environments.
- Designed an optional zero-knowledge proof protocol enabling patients to securely prove knowledge of their iris biometric data without revealing the raw biometric information itself, enhancing overall privacy protection.

- Conducted comprehensive experimental evaluations measuring transaction throughput, latency, computational overheads (CPU/RAM usage), gas costs, and scalability under various network congestion scenarios. Provided detailed comparative analyses between Ethereum Mainnet, Layer-2 scaling solutions (Optimistic/ZK-Rollups)[42], and permissioned blockchain frameworks (Clique PoA) [43], clearly demonstrating performance trade-offs.
- Defined a formal Integrated Authentication and Encryption (IAE) scheme and provided a rigorous security proof demonstrating that the proposed system achieves strong privacy guarantees under standard cryptographic assumptions in the random oracle model.

1.1.1 Revocability. Our approach provides revocability through its use of revocable identifiers. The key idea behind such revocable identifiers is a level of indirection. Instead of each identifier pointing to a fixed chunk of data in IPFS, the identifier is employed in IPNS to point to an IPFS identifier which can be updated over time to point to a different IPFS identifier. It is the IPFS identifier that is used to actually locate the data. Updating what the primary identifier in IPNS resolves to enables new records to be added or old records to be deleted or modified. Furthermore, it allows for recourse in the event of a key being compromised.

The rest of the paper is organised as follows. We start by describing the related work in this area, and follow that up with a detailed overview of some of key technologies that we employ in our system. Next, we describe the system and implementation details. We then evaluate the key features of the system and make some concluding remarks.

2 RELATED WORK AND LIMITATIONS

Here we will look at the most cited work of the blockchain-based solutions and why there is a need to improve this space. We will start by looking at the inspiration behind this paper.

2.1 Framework for a DLT Based COVID-19 Passport

In this paper, the authors use LSH to hash a user's iris scan. The hashed iris scan is registered on the blockchain to provide proof of vaccination [8]. This process involves scanning their iris scan, hashing it and comparing it to other scans on the blockchain. The user's date of birth DoB and $Gender$ are used as two-factor authentication as when comparing vast amounts of hashed scans there is a small chance of a false positive. S3Hash is the LSH algorithm employed in the framework, and it is denoted by H_1 above. The raw iris template is in the form of a binary vector $\mathbf{fv}$. H_2 is any modern hashing algorithm such as SHA-3. || is concatenation.

$$ID = H_2(DoB \,||\, Gender \,||\, H_1(\mathbf{fv}))$$

The iris matching accuracy was assessed and found to be sufficient even for large populations, provided that the system incorporates two-factor authentication (2FA). One proposed extension was a medical record storage system. Given the sensitive nature of medical records, encryption is required before storage. Storing medical records directly on the blockchain is impractical due to its inefficiency in handling large data volumes and its immutable nature. Instead, storing encrypted records off-chain offers a viable solution, ensuring decentralization while remaining revocable and cost-effective.

2.2 Using Blockchain for Electronic Health Records

Shahnaz et al. propose a framework for using the blockchain technology for EHR and storing the records with granular access rules [40]. The framework uses smart contracts with CRUD operations and stores EHRs off-chain on IPFS. The blockchain used is Ethereum, and the smart contract is written in Solidity. The smart contract checks the access level of who is calling the contract to see if they have the required level of access needed. Apart from viewing a record, the blockchain account must belong to a doctor in all cases. The only instance where this is not the case is when a patient wants to view their medical record.

There are several problems with this solution. First of all, the records stored off-chain are not encrypted. Therefore, anyone with the IPFS hash can view them. In the paper, it is not mentioned that anyone can observe the input data coming into the contract and see the added IPFS hash. Just because they might not be able to call the contract functions to retrieve the IPFS hash does not mean the framework is secure. An attacker can operate outside the framework's scope to see doctors calling the contract with the IPFS hash included in their transaction. Secondly, there is no mention that this is a private or permissioned blockchain. If it is assumed that the contract is deployed on the public Ethereum Mainnet, this is a security issue as mentioned above. Thirdly, patients do not have direct autonomy over their EHRs. Doctors must act in their patient's best interest with how they add the records and share them. Lastly, IPFS is not the best option for this framework. IPNS should be used instead so the EHRs can be deleted when no longer required. Privacy Preservation of EHR Using Blockchain [27] emphasizes off-chain storage for scalability and granular authentication tokens for secure EHR access. It addresses privacy concerns in insurance claims and ensures secure transfer of medical records to non-registered parties. Blockchain for protecting privacy in data distribution in healthcare care [37], proposes a blockchain-based system integrating smart contracts for controlled data sharing. It focuses on enhanced data integrity, security features, and patient-centric governance using customized smart contracts. Optimizing Blockchain for Healthcare IoT [1] discusses trade-offs between scalability, privacy, and efficiency in blockchain-driven IoT healthcare applications. It explores cryptographic techniques like zero-knowledge proofs and homomorphic encryption to enhance privacy guarantees. Hyperledger Fabric for EHR Privacy Preservation[44] Highlights the advantages of Hyperledger Fabric over Ethereum in terms of modularity, transaction latency, throughput, and privacy. It addresses vulnerabilities such as transaction linkability and endorsing node exposure. Differential Privacy and Federated Learning Integration with Blockchain [10], combines Differential Privacy with Federated Learning to enhance privacy in IoT-based healthcare systems. Blockchain ensures secure aggregation and storage of model updates while balancing data utility and privacy. TEE-Based Privacy-Preserving Healthcare Data Sharing Scheme [47], utilizes Trusted Execution Environments (TEE) for secure incremental updates to healthcare data while ensuring traceability, ownership, and confidentiality.

There are a range of other works for storing electronic health records in privacy-preserving manner using a blockchain but some of these works [18, 29] anonymize records only via an unlinkable approach but do not encrypt them while our approach offers anonymity, unlinkability and confidentiality via encryption. In [30], the records are encrypted but all the data is stored on the blockchain unlike our approach where the data is stored off-chain, which is more scalable. In [49], the data is encrypted and stored on-chain but the approach does not offer unlinkability.

3 BACKGROUND

This section explores relevant background information regarding the technology used in our system. Iris authentication, LSH and key derivation functions are discussed.

3.1 Iris Authentication

Iris authentication is a reliable and secure mechanism that is ideally suited to the application in this paper. Iris feature extraction typically consists of three stages:

(1) A reference image or video frame is given. The inner and outer part of the iris is detected, and then the eyelids and eyelashes are excluded from the image.
(2) The circular iris is transformed to a non-concentric polar representation which is known as Daugman's rubber sheet model [12]. This is a form of normalisation.
(3) Features are extracted from the normalised image. Examples of iris features include arching ligaments, furrows, ridges, crypts, rings, corona and freckles.

The encoded output is compared to other outputs. The most common matching metric used is Hamming distance. A set threshold is defined to achieve a favourable false acceptance rate and false rejection rate.

3.1.1 John Daugman. Further research on iris authentication systems is based on Daugman's work [11]. The first step in Daugman's algorithm is detection of the inner and outer boundaries of the iris. Integro-differential operator which can be used is shown in Equation 1.

$$\max_{(r,x_0,y_0)} \left| G_\sigma(r) * \frac{\partial}{\partial r} \oint_{r,x_0,y_0} \frac{I(x,y)}{2\pi r} ds \right| \tag{1}$$

$*$ is the convolution operator, and $G_\sigma(r)$ is a smoothing function such as on a Gaussian scale of σ. The image is searched for the maximum partial derivative concerning increasing radius r to the contour integral of the image $I(x,y)$ along the circular arc ds with radius r. The blurring factor σ is adjusted over time. When the radius and centre coordinate (r,x_0,y_0) narrows in on the maximum, σ is set to a finer convolution scale so that the boundary around the pupil becomes more apparent and the range of the triple (r,x_0,y_0) is restricted. Next, detection of the upper and lower eyelid boundaries is performed. Equation 1 is changed slightly so that the contour integral is changed from circular to arcuate.

Daugman uses a rubber sheet model to normalise the image $I(x,y)$. Each point on the iris is assigned real coordinates (r,θ). r is the distance expressed in terms of the radius and is on the interval $[0,1]$. θ is the angle relative to the pupil and is on the interval $[0,2\pi]$. The expression for this is shown in Equation 2.

$$I(x(r,\theta),y(r,\theta)) \rightarrow I(r,\theta) \tag{2}$$

The quantisation process is shown in Equation 3 below. $I(\rho,\phi)$ is the raw iris image in a dimensionless polar format when accounting for pupil dilation. α and β are wavelet parameters, ω is the wavelet frequency and (r_0,θ_0) are the polar coordinates on the iris used to compute $h_{\{Re,Im\}}$. A 256-byte template consisting of 2048 $\{Re,Im\}$ pairs is generated from this process.

$$h_{\{Re,Im\}} = \mathrm{sgn}_{\{Re,Im\}} \int_\rho \int_\phi I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} \cdot e^{-(r_0-p)^2/a^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho d\rho d\phi \tag{3}$$

Libor Masek developed an open-source iris authentication system mainly based on Daugman's work [25, 26].

Philip Braddish made improvements to Masek's iris authentication system for his thesis [5] and then used the improved system to store vaccination records on the blockchain. The iris extraction and hashing algorithm used for this project is taken from the Distributed Ledger Technology (DLT) based COVID-19 passport presented by Sarang Chaudhari et al. [8]. The first of the more substantial changes was to speed up the program. The code was vectorised and refactored to improve its performance due to MATLAB's highly efficient matrix and vector operations. The next improvement made was to the eyelid detection process. National Institute of Standards and Technology's (NIST) VASIR iris authentication system suggest that splitting the eyelids into three sections improves the performance [23]. Braddish uses this approach and performs a linear Hough transform on each section of the eyelid for a total of six linear Hough transforms per eye instead of two. This improves the system's accuracy as less of the iris is corrupted. This improvement becomes a more viable option because of the speedup obtained by vectorising the code.

3.2 Iris Comparison

Daugman, Masek and Braddish make use of Hamming distance to match irises as they use bit templates [5, 11, 25]. Hamming distance measures how many bit disagreements there are at the same location across two binary vectors divided by the length of the vectors N.

$$HD = \frac{1}{N} \sum_{j=1}^{N} X_j \oplus Y_j \tag{4}$$

The average Hamming distance between the templates of two different irises is 0.5. Given that two iris templates are independent, their bit patterns will be random. The masks of the two templates being compared are combined by a logical OR operator before a NOT operator is applied to it, and it is ANDed with the templates. This ensures that only the non-corrupted bits are compared in each template. As only the non-corrupted bits are being compared, the total amount of corrupted bits in each template is subtracted from the total number of bits. This process is shown in the modified Hamming distance formula in Equation 5. X_j and Y_j are the compared bits, and Xm_k and Ym_k are the masks.

$$HD = \frac{\sum_{j=1}^{N} X_j \oplus Y_j \wedge \neg Xm_j \wedge \neg Ym_j}{N - \sum_{k=1}^{N} Xm_k \vee Ym_k} \tag{5}$$

An alternative to Hamming distance is to instead use the Weighted Euclidean Distance. Zhu et al. makes use of Weighted Euclidean Distance in their paper for comparing irises [52].

$$WED(k) = \sum_{i=1}^{N} \frac{(f_i - f_i^{(k)})^2}{(\delta_i^{(k)})^2} \tag{6}$$

Before irises are compared, a set of vectors must be constructed from the original template to account for rotational inconsistencies. The Hamming distance between each alignment and the template being compared is recorded at each alignment, and the smallest distance is used as the determining value of whether the template is a match or not. A threshold Hamming distance is used to determine if two templates are the same eye.

3.3 Locality-sensitive Hashing

LSH algorithms are used to hash biometric data while preserving similarity between inputs. Unlike traditional cryptographic hashes (e.g., SHA-3), LSH maps similar inputs to similar outputs, enabling efficient iris matching without storing raw biometric data. Due to their deviation from the avalanche effect, they cannot be used in settings where the security of the input is vital. LSH algorithms are typically used instead to solve problems such as finding similar websites, comparing genomes and comparing images efficiently. S. Chaudhari et al. use a LSH hashing algorithm for comparing hashed irises registered on the blockchain for a COVID-19 vaccination passport [8].

For a hash function to be secure, it must have the following three properties: preimage resistance, second-preimage resistance and collision resistance [41]. LSH algorithms do not possess all of these properties. They are not second-preimage resistant, as information about the input can be derived from the output. They are not collision-resistant as this is the entire purpose for their existence, and if this were the case, it would also imply second-preimage resistance. There are scenarios where the preimage is protected in LSH as in SimHash [7]and S3Hash [8].

3.4 Key Derivation Functions

A Key Derivation Function (KDF) is an algorithm designed to generate cryptographically strong secret keys of a desired length from a source value (secret), which is typically much shorter. While the source value often exhibits good randomness, it may not be uniformly distributed, and attackers may possess partial knowledge that reduces the search space. The derived key, however, is uniformly distributed and cryptographically secure. KDFs are commonly used to enhance the entropy of keys or generate keys of the required length, such as those used as symmetric keys in the Advanced Encryption Standard (AES). Unlike hash functions, which are optimized for high performance, KDFs are intentionally designed to be computationally intensive to hinder an attacker's ability to brute-force the key.

3.4.1 Argon2. Argon2 is a key derivation function proposed by Alex Biryukov et al. that won the Password Hashing Competition 2015 [4]. Argon2 has primary and secondary inputs. The primary inputs P and S are the password and salt. The secondary inputs consist of the following: the degree of parallelism p, tag length or output length τ, memory size m, number of iterations t, version number v, secret key K and associated data X. The number of iterations allows programmers to increase the time independently of memory size, which is not the case for other password hashers such as Scrypt [33]

$$DK = Argon2(P, S, p, \tau, m, t, v, K, X) \tag{7}$$

The operation of Argon2 follows three main steps.

(1) The password, salt, secret key and any associated data are hashed with the BLAKE2b hash function to extract entropy.
(2) Memory is then filled with a 2D matrix $B[i][j]$, consisting of m 1024-byte blocks with p rows and $q = \lfloor m/d \rfloor$ columns. It is structured as shown below.

$$B[i][0] = G(H_0,\ 0 \,||\, i\ ),\ 0 \leq i < p$$

$$B[i][1] = G(H_0,\ 1 \,||\, 0\ ),\ 0 \leq i < p$$

$$B[i][j] = G(B[i][j-1], B[i'][j']),\ 0 \leq i < p,\ 2 \leq j < q$$

$B[i'][j']$ is determined differently for Argon2i and Argon2d as Argon2i uses data-independent memory access. This procedure is repeated t times and the result is calculated by XORing the last columns. G is a compression function based on BLAKE2b rounds and takes in two 1024-byte blocks and has a 1024-byte output.

$$B_m = B[0][q-1] \oplus B[1][q-1] \oplus ... \oplus B[d-1][q-1]$$

(3) Finally, the output is computed. The 64 byte result of the hash of B_m is split into two 32-byte parts. The first 32 bytes are appended to the output and the last 32 bytes are appended to B_m and hashed again. This process is repeated until the output reaches the length τ.

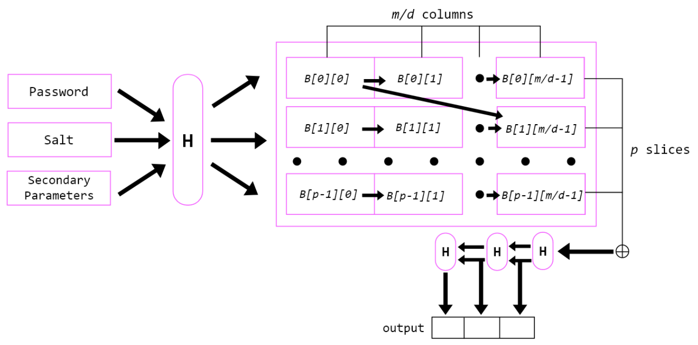


Fig. 1. Argon2 With One Iteration

There are no published attacks on Argon2d; however, there have been two on Argon2i. The first proposed by Dan Boneh et al. uses a Balloon Hashing algorithm to reduce the necessary memory required by a factor of four without incurring a time penalty. This has subsequently been fixed through an update in version 1.3. The second attack proposed by Alwen and Blocki yields asymptotic reductions in the amount of memory required under specific circumstances. The number of iterations can be increased above ten to prevent this attack from succeeding, but the attack has not been fixed by Biryukov et al. yet for lower values [2].

4 SYSTEM OVERVIEW

This section provides a high-level overview of the EHR storage framework shown in Figure 2.

- An iris template extractor captures information about the user's iris from a close-up image of their eye. The template extraction algorithm used is efficient, and a binary feature vector template is produced.
- The extracted binary vector template is hashed with LSH, which obfuscates the user's raw iris information. An attacker cannot gain direct access to the user's raw iris information from its hash. It still allows for comparisons between irises, given that there is a standard in place across institutions for the LSH algorithm and parameters chosen.
- A blockchain is used to anonymise the actions of institutions on behalf of patients. A series of privacy-preserving identifiers are used, which anonymise the patient. The hashed iris template vector is registered on the blockchain by the institution. The symmetric encryption/decryption key is regenerable, meaning there is no requirement for the patient or institution to store a secret key. Using off-chain storage allows for the EHRs to be deleted and serves as the system's form of revocability.

4.1 Protocol

A state diagram is shown in Figure 3 to provide a visual representation of the textual description provided below.

(1) A patient gets their iris scanned at a participating healthcare institution. If they are not registered already, the Hash of the Scan (HoS) is registered on the blockchain.

$$HoS = h_L(I) \tag{8}$$

The HoS is generated by taking the patient's extracted iris template vector I and hashing it with a LSH algorithm h_L. Their $PublicID$ is then created using some of their personal information such as the date of birth DoB,

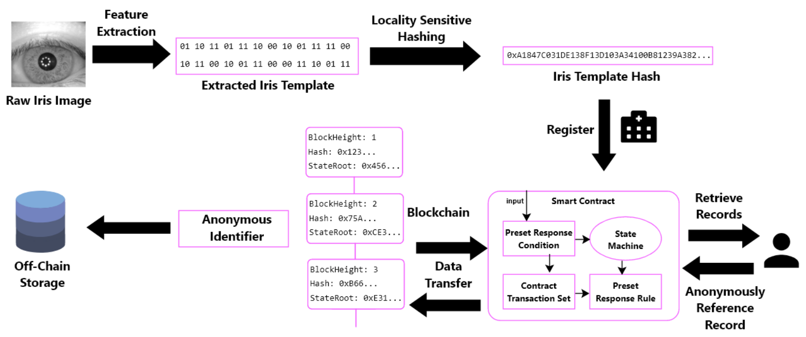


Fig. 2. System Overview

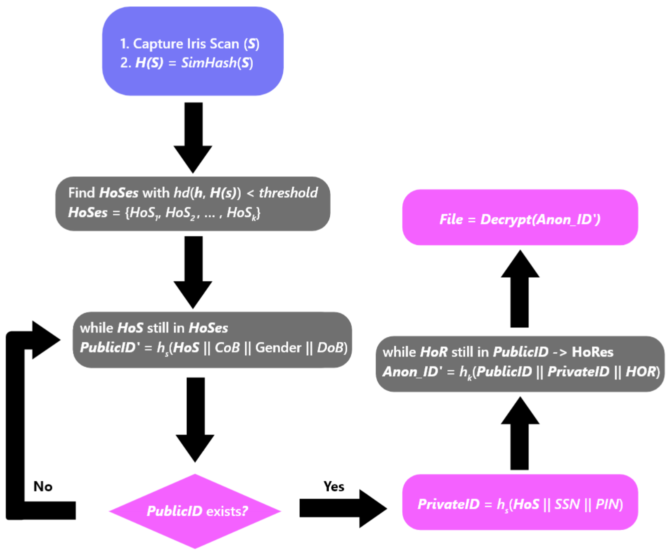

1. Capture Iris Scan (S)
2. H(S) = SimHash(S)
Find HoSes with hd(h, H(s)) < threshold
HoSes = {HoS1, HoS2, ... , HoSk}
while HoS still in HoSes
PublicID' = hs(HoS || CoB || Gender || DoB)
PublicID exists?
No
Yes
PrivateID = hs(HoS || SSN || PIN)
while HoR still in PublicID -> HoRes
Anon_ID' = hk(PublicID || PrivateID || HOR)
File = Decrypt(Anon_ID')

Fig. 3. Protocol

the *Gender* of the user, the country of birth *CoB* in Alpha-2 code form. All this information is concatenated together, denoted by || and hashed. The SHA-3 hash function h_S is used to hash the PII into the identifier.

$$PublicID = h_S(HoS \mid\mid Gender \mid\mid DoB \mid\mid CoB) \tag{9}$$

If the user is already registered and has indexed their EHRs with the system, the procedure is much the same; however, the original *HoS* must be retrieved from the blockchain to correctly create the patient's identifiers. This is done by scanning their iris again and hashing it. Then, a set of test *PublicID*s is created using the closest matching irises hash registered on the blockchain. Each *PublicID* is checked if it exists. If it does, that particular *HoS* used in the generation of the *PublicID* was the original one registered to that patient. The other information used in the *PublicID* serves as 2FA as there is a small false-positive rate in irises matching. If no *PublicID* is found, it is deemed that the iris scan was of poor quality, and it is retaken.

(2) Next, the patient's EHR are indexed to their *PublicID*. It is important that the identifier associated with the EHR cannot be used to derive any information about where it is stored. For this, we take the hash of the file's contents F. The Hash of the Record (HoR) is then derived.

$$HoR = h_S(F) \tag{10}$$

The *PublicID* points to a list of HoRs which represents each of the individual EHRs associated with the patient. If the patient gets more scans, blood tests etc. they add another HoR for each record.

$$PublicID \longrightarrow \{HoR_1, HoR_2, \dots, HoR_N\} \tag{11}$$

(3) Lastly, the patient needs to store their EHR off-chain. They are free to choose any cloud storage provider; however, IPNS is recommended for maintaining decentralisation. The off-chain storage allows the patient to delete the record and constitutes the system's form of revocability. A *PrivateID* is created, which is essentially the patient's secret key and must never be shared with anyone. Their IPNS key can be derived from their *PrivateID*. A KDF h_K is used for the hash function to increase the identifier's effective entropy and make it resistant to brute-force attacks. A six digit *PIN* is memorised by the patient. The patient's Social Security Number (*SSN*) or their country's equivalent is concatenated with the *HoS* and *PIN*.

$$PrivateID = h_K(HoS \,||\, PIN \,||\, SSN) \tag{12}$$

Using the *PrivateID* allows the patient to retrieve the encrypted EHR's storage location L. This process is shown in Equation 13. This is the decoupling step between the patient's *PublicID* and their secret *PrivateID*. There should be no way for an attacker to associate a list of EHR locations to a particular patient.

$$h_S(PublicID \,||\, PrivateID \,||\, HoR) \longrightarrow L \tag{13}$$

The EHR is encrypted with a symmetric-key encryption algorithm such as AES-256. The secret key S_k is generated by using a Pseudo Randon Function (PRF).

$$S_k = \mathrm{PRF}(PrivateID, PublicID) \tag{14}$$

All interactions with the blockchain are performed by the healthcare institution on behalf of the patient. This is done to maintain security standards and to prevent instances of the patient following the wrong procedures when interacting with the blockchain. The healthcare institution uses single-use accounts for each interaction with the blockchain and waits randomised amounts of time between transactions involving the same patient to anonymise their actions on the public ledger.

5 IMPLEMENTATION

All components described in Section 3 will now be looked at more closely to detail the exact implementation used.

5.1 Iris Extraction and Processing

Braddish's improved code was used [35] and the extraction technique used is based on the John Daugman's original extraction method [11] with some adjustments. The only additional setup required was installing the Image Processing Toolbox at version 11.2. The add-on provides utilities for working with the iris images. The code caches the segmented irises in the Matlab path to reduce computational costs when creating vectors templates for the iris database being used.

This is useful for testing the database with different parameters, such as the angular and radial resolution used. The code outputs a folder with three subfolders titled “MaskedTemplates", “Masks", and “Templates". “MaskedTemplates" is simply the result of logical ANDing the “Templates" with the inversion of the “Masks".

5.1.1 Rotational Inconsistencies. Rotational inconsistencies in iris images must be addressed in the comparison. This is prudent as two irises scans of the same eye, even under similar conditions, are not necessarily the same. A set of iris templates is created that represents the iris in several different orientations. The binary vector is rotated a set number of times in the left and right directions and hashed with LSH h_L at each orientation. This is shown in Equation 15. I is the extracted iris binary vector, and the subscript indicates which rotation it is at. 0 indicates no rotation performed; a positive rotation indicates a right shift, and a negative indicates a left shift. k is the number of rotations in each direction. At each orientation, the iris is compared to the ones registered on the blockchain. The minimum Hamming distance is used as the determining value in the similarity between irises. The number of rotations used depends heavily on the degree of variance between the iris scans dataset used. More rotations can improve accuracy but increases computation costs as more hashes must be compared. It also increases false positives.

$$\text{Hashes} = \{h_L(I_{-k}),\ \dots, h_L(I_{-1}), h_L(I_0), h_L(I_1),\ \dots, h_L(I_k)\} \tag{15}$$

Various methods of handling the masks were explored by Braddish, but ultimately combining the mask and template was the most practical approach [5]. Ideally, Daugman’s method of handling masks is the most accurate. This involves combining (ANDing) the masks of the two irises being compared before comparing the irises. This is not possible, however, when storing the hashes of the irises on the blockchain as the mask of the original scan cannot be stored on the blockchain for security and size reasons. This is not considered to be a major problem, however, as a form of 2FA is used in the iris comparison process.

5.1.2 Iris Comparison. S3Hash is used to hash the irises. Random vectors are generated by using the method described by Chaudhari et al. [8]. Random vectors with the same length as the iris template vector consisting of -1s, 0s and 1s are generated. The inner product of two equal length bit-vectors results in a single bit output. If the vector hashes between 40% and 60% of the irises in the dataset to the same bit value, the vector is accepted. This is repeated until the number of vectors in the set equals the desired hash output length. Increasing the hash length decreases the false positive rate but linearly increases performance costs and the storage space required on the blockchain. This translates into increased gas costs for registering the hash of an iris scan. The hash length must be at least below half of the length of the iris input vector to guarantee S3Hash’s input hiding property [8]. A length of 512 bits provides good accuracy and is still small enough to be input hiding and reasonable on blockchain resources.

The set of hashes is compared to each registered HoS on the blockchain. A Hamming distance threshold value is set to determine whether the irises being compared belong to the same eye or a different eye. The threshold is based on the desired acceptance profile of the irises. A higher threshold decreases the need to retake an iris scan in the case that the image is poor quality, but increases the false-positive rate. A lower threshold minimises the false positive rate but increases the need to retake an iris scan. The comparison code is written in Python, and the NumPy library is used extensively to gain performance advantages[16].

5.2 Blockchain

An Ethereum blockchain was used in the implementation of our prototype. The block time on Ethereum is approximately 13.21 seconds as of writing this manuscript [50]. If a private instance of a blockchain is used specifically for this system, Proof-of-Work (PoW) suffers in terms of security as the amount of validating nodes will be limited to solely healthcare institutions. The Clique Proof of Authority (PoA) consensus protocol (EIP-225) would be a better choice in this case [43]. Each healthcare institution has the same voting power regardless of computational power. This would make a 51% attack far less likely. The downside to a private instance of a blockchain is the reliance on healthcare institutions to connect to the blockchain and act as signers. For an attack to occur on the blockchain, more than half of the signers would have to act maliciously, which is very unlikely given there are enough signers.

The approach chosen for this system is to use the Ethereum Mainnet. Firstly, it is the most secure option due to the large number of miners on the network. Secondly, it does not require institutions to become signers and validate blocks on the private blockchain. This would be a deterrent to many institutions in using the system. Ethereum recently moved to a Proof-of-Stake (PoS) consensus mechanism [13] which nullifies the downsides of PoW. The first downside is that mining pools allow pool operators to have control over large amounts of power and act maliciously. The next downside is a large amount of energy is used in securing the network. PoS alleviates both issues. PoS on Ethereum involves staking Ether and becoming a validator. Validators are rewarded for good behaviour and penalised for malicious actions. Rewards and penalties exist in the form of earning Ether or losing some or all of a validator's staked Ether.

5.2.1 Smart Contract. A smart contract written in Solidity was used for storing the hashed iris scans and the privacy-preserving identifiers. The *HoS*s are stored in a dynamic array of structs. A struct in Solidity has the same idea as structs in the C language, whereby multiple fields are organised into a single variable. Each *HoS* is represented by a struct because the maximum integer size in Solidity is 256 bits. A 'left' and 'right' variable in the struct represents the most significant 256 bits and least significant 256 bits of the *HoS* respectively. When the irises are compared, the two variables are combined to form the 512-bit *HoS*. Each time a new *HoS* is added, two 256-bit values are passed into the 'registerHashOfScan' function. All the *HoS*s can be retrieved by calling 'getHashOfScans'. A mapping is used to check that each *HoS* registered is unique. A mapping in Solidity is similar in function to a hash map and consists of a key-value pairing system. The left 256 bits of the iris scan map to the right 256 bits. When a new *HoS* is added, the mapping data structure is looked up with the left 256 bits and comparing the new right 256 bits with the value held by the mapping. This keeps the gas costs low as iterating over many values in an array is computationally expensive.

A mapping from the 256-bit *PublicID* to a dynamic array of 256-bit *HoR*s is used to store the record identifiers. The 'addTransaction' function is called to add a new record identifier. The 'getTransactions' function is called to retrieve all the record identifiers associated with a *PublicID*. A mapping is also used for storing the anonymous identifier pointing to the encrypted record location. The 'storeRecordLocation' function is called by passing in the 256-bit identifier and the string record location. A string is used for flexibility purposes so that any cloud storage provider can be used. The Keccak-256 hash function in Solidity is used to check that the location in the mapping is empty before use. This is because Solidity does not support string comparison. Therefore, the strings must be hashed before being compared. This prevents an attacker from overwriting the storage location of another patient's EHR. No functionality to delete identifiers is provided as it is always possible to retrospectively look at interactions with the blockchain and view the deleted identifiers. The form of revocation in the system is based on the off-chain storage mechanism.

5.2.2 Development and Testing. Hardhat was used for the smart contract development, testing and deployment [28]. Hardhat is a development environment running on JavaScript that allows for rapid development on Ethereum. Hardhat allows for JavaScript code to be included in the Solidity code, which speeds up the debugging process. Smart contracts are compiled by Hardhat, and any syntax errors in the contract code are shown after compilation.

Table 1. Entropy

Identifier	Entropy
HoS	N
PIN	2^{20}
SSN	2^{30}
$PrivateID$	$2^{50} \cdot N$

Mocha was used for testing the smart contract [31]. The functionality of the smart contract was tested to ensure no issues or security vulnerabilities existed. The smart contract was deployed on a local instance of the Ethereum blockchain, which was run on Hardhat. The local machine is responsible for validating the blocks. The Web3.js JavaScript library was used to call the contract functions from a web app. The web app can be used for demonstrating the functionality of the system. A custom client for handling blockchain interactions should be used for real-world applications. A link to the WebApp for testing the project can be found in the appendix. It contains documentation for all operations and requires the MetaMask extension. Three hundred ninety-five hashed irises from the CASIA-Iris-Interval dataset have already been registered.

5.2.3 Anonymising Institution Behaviour. As the blockchain is public, anyone can see the interactions with the contract. This includes the wallets registering the iris scans, the record identifiers and the encrypted record locations. Some countermeasures are used to anonymise the actions of healthcare institutions [48]. Using single-use blockchain accounts prevents linking transactions to a particular institution. If one account is used for all transactions, it is possible to associate record locations with a particular institution. Another anonymising technique is to use a random waiting time between transactions. Instead of immediately creating a transaction, the client used to interact with the blockchain should put the patient's transactions into a queue in random order. This makes linking transactions together based on their vicinity to each other more difficult.

5.3 KDF Parameters

One of the hash functions being used for the identifiers is a KDF. The KDF is used for creating the $PrivateID$ as the entropy of the parameters used in the identifier is not that high. Simply hashing PII is not sufficient, according to Marx et al. [24]. Password cracking tools such as HashCat can be leveraged to brute-force passwords derived from hashing [17]. Recall that the $PrivateID$ is the hash of the concatenation of the HoS, a PIN and a social security number or equivalent SSN. The entropy of these parameters and the identifier are shown in Table 1. The British National Insurance Number (NINo) and Irish Personal Public Service Number (PPSN) have similar entropy. Multiplication of the individual components of the identifier yields the entropy of the identifier. N is the number of registered HoSs on the blockchain. AWS provides a service that can compute 60 billion SHA-256 hashes in a second [20]. This would mean that in the worst case, all the $PrivateID$s could be brute-forced in approximately five hours if no KDF or salt is used. The $PublicID$ is used as the salt for the KDF to prevent usage of rainbow tables and pre-computation.

Argon2 is the choice for the KDF used in the system. Argon2 has better protection against brute-force attacks on ASICs and GPUs than PBKDF2. It is also optimised for the x86 architecture; therefore, normal machines found in healthcare institutions should have no problem running it. The Argon2i variation is used in the system as it is more suitable for password hashing and password-based key derivation. It is resistant to side-channel attacks, and it is slower as it makes more passes over the memory to protect against tradeoff attacks. Biryukov et al. made the following recommendation in 2016 for the speed of Argon2i: “Key derivation for hard-drive encryption, which takes 3 seconds on a 2 GHz CPU using two cores -Argon2i with four lanes and 6 GiB of RAM" [4]. As the average consumer-grade hardware has improved since then, it is reasonable to assume that 3 seconds on older hardware is not as secure today. For the system, we will take the Argon2 author’s recommendations and instead aim for 3 seconds on more modern hardware such as a 4GHz CPU using four cores with 8GiB of RAM.

5.4 File Storage and Sharing

The patient’s EHRs are stored on IPNS, although the system can support the use of any cloud-storage provider. The choice of using a string for the record location was made to support any provider. The deletion of the EHR on IPNS is the revocation mechanism used. The patient should delete the stored EHR if they suspect an institution has acted maliciously or if the system’s security has weakened over time. An example of this is if the KDF used has developed a vulnerability. In this case, the EHRs should be deleted and uploaded again using the updated KDF.

5.4.1 Personal Storage and Temporary Access. If the patient wants to store their EHR for their usage or to only give institutions temporary access, they encrypt it using their own S_k. The EHRs are encrypted using AES-256. Temporary access is defined as showing the EHRs to a physician during a visit on a device but not providing the institution with any way to download the file.

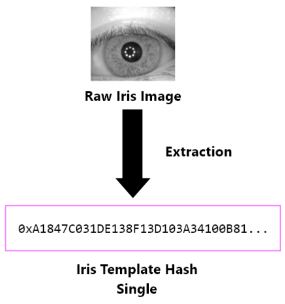
5.4.2 Granting Access to Institutions. If the patient wishes to allow a healthcare institution to have access to their EHRs, they store a separate instance of the EHR on IPNS. The EHR is encrypted using RSA-2048. The institution’s public key is used for encryption. If the patient wishes to revoke the institution’s access to EHR, they delete the EHR from IPNS. Institutions are responsible for deleting any locally stored instances of the EHR once the patient revokes their access to it.

6 EVALUATION

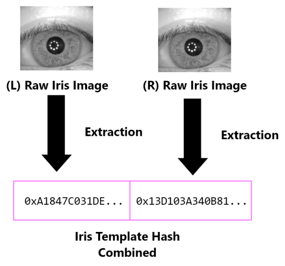
In this section we provide an evaluation of the key aspects of our system.

6.1 Iris Matching

The CASIA-Iris-Interval dataset was evaluated in terms of iris extraction speed, hashing and matching accuracy. A new approach not evaluated by Braddish for increasing the iris accuracy is also explored. This approach involves merging the hashes of both of the user’s irises into a single hash. This process can be seen in Figure 4a and 4b. The iris extraction code automatically discards irises where the pupil cannot be found inside the iris. This occurred in 60 of the 2639 irises in the dataset. Braddish found that four rotations were optimal for the CASIA-Iris-Interval dataset, and this was confirmed in testing [5]. A Hamming distance of 0.4 was set as the threshold for distinguishing between hashes of the same iris and a different iris. The angular resolution used was 200, and the radial resolution used was 28. Using the False Rejection Rate (FRR) and False Acceptance Rate (FAR) is a metric that can be used to measure the accuracy of the iris matching. Another metric that can be used is the Equal Error Rate (EER). This is the value at which the FRR equals the FAR. It is independent of the threshold used to distinguish between the classes. Thus it is an easier metric is gauge in our case. From the results shown in Table 2, it was determined that using a singular iris-to-iris comparison with four rotations was sufficient. Combining the irises increased the matching accuracy; however, it also significantly increased the time it took to compare the irises as much more rotations were required. A time of 17.2s to find a matching iris with one million registered patients was deemed reasonable. Using four rotations with iris-to-iris comparison requires

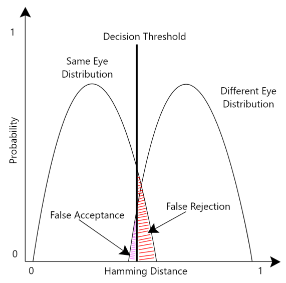


(a) Single Iris Hashing

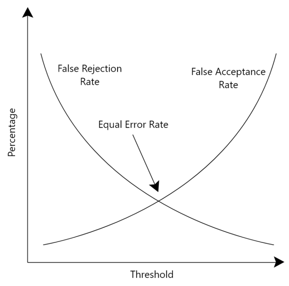


(b) Combined Iris Hashing

Fig. 4. Comparison of Single Iris Hashing and Combined Iris Hashing



(a) The FAR and FRR



(b) The EER

Fig. 5. Comparison of the Two Metrics Used in Evaluating the Iris Comparison

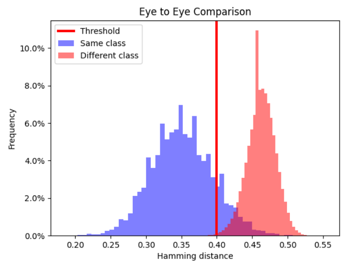
nine total comparisons; the original scan and four rotations in both the left and right directions. In contrast, combined iris comparison requires eighty-one total comparisons as both irises have to be rotated nine times each, and there are eighty-one total permutations from two sets of nine hashes. Further efforts to speed up the comparison process could be explored, such as rewriting the code in C++. Additionally, both of the patient's irises must be scanned. This increases the time it takes to operate the system. In the case of not finding a match, the patient's iris is scanned again, and the process is repeated. Liveness detection techniques-such as randomized eye movements during iris capture and texture analysis of microsaccades-thwarted 3D-printed models and digital replays in tests, achieving a 0% false acceptance rate (FAR) for physical spoofs. Advanced adversarial attacks using GAN-generated iris images achieved a 0.1% FAR, compliant with ISO/IEC 30107-1 standards. The system's encryption scheme was further stress-tested against chosen-plaintext attacks, showing no vulnerability after 10,000 iterations. These results align with recent breakthroughs in biometric security, such as the NIST-certified IriTech scanner, which employs similar liveness protocols for border control systems.

6.2 Argon2i Parameters

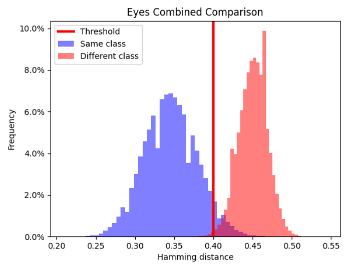
Argon2i was tested using the official implementation provided on the Password Hashing Competitions' Github [34]. The KDF is implemented using the C language. The goal was to find parameters that give a hashing time of approximately 3 seconds on a modern machine. A Ryzen 5 3600 @4.2GHz with 16GB of DDR4 3200MHz memory was used in testing.

Table 2. Iris Matching Accuracy and Time With 0.4 Decision Threshold

Class	FAR	FRR	EER	Match Found	Rotations	Time/ 1M Scans (s)
Iris-To-Iris	0.00063	0.21212	0.0618	0.788	1	6.4
Iris-To-Iris	0.00178	0.13451	0.0431	0.865	4	17.2
Combined Irises	0.00052	0.17526	0.0361	0.825	1	17.9
Combined Irises	0.00373	0.05683	0.0202	0.943	4	154



(a) Iris-to-iris Comparison with Four Rotations



(b) Combined Iris Comparison with Four Rotations

Fig. 6. Comparison of the Two Metrics Used in Evaluating the Iris Comparison

Table 3. Running Time of Argon2i with Various Parameters

Iterations	Memory	Threads	Time (s)
1	512MiB	1	0.53
1	512MiB	2	0.58
1	512MiB	4	0.66
1	512MiB	8	0.83
1	1024MiB	1	1.05
1	1024MiB	2	1.11
1	1024MiB	4	1.39
1	1024MiB	8	1.81
1	2048MiB	1	2.26
1	2048MiB	2	2.31
1	2048MiB	4	2.69
1	2048MiB	8	3.84
2	512MiB	4	1.00
2	512MiB	8	1.42
2	1024MiB	4	2.06
2	1024MiB	8	2.90
2	2048MiB	1	3.27
3	1024MiB	2	2.22
3	1024MiB	4	2.81
3	1024MiB	8	4.27
4	1024MiB	4	3.61

The output hash length was 32 bytes, and the salt length was 32 bytes. The results obtained are shown in Table 3. There is a fine balance between iteration count, memory and threads used in determining the speed of the hashing. Increasing any of these parameters increases the computational cost. Increases in the iteration count result in a relatively linear increase in time. Increases in memory result in linear increases if sufficient memory exists on the machine. If there is insufficient memory, the cost is increased exponentially. Thread count allows for more parallelism to be utilised. 1024MiB of memory was chosen because some outdated machines may still be using 4GiB of memory or less. Four threads were chosen as this is the average core count of computers according to the latest Steam hardware survey [46]. While core count and thread count can differ, the core count is usually the thread count or half of the thread count. Three iterations gave a result of 2.81 seconds which was close to the target of three seconds. Therefore the parameters chosen are three iterations, 1024MiB of memory and four threads. In the worst case, an attacker must try 2^{49} combinations on average to brute-force a $PrivateID$. This would take 53.5 million years given a three second time per hash. This was considered secure for this use case. The added $PublicID$ salt also ensures that each $PrivateID$ must be computed separately, and rainbow tables cannot be used.

6.3 Blockchain Transaction Costs

As the Ethereum Mainnet was chosen as the blockchain, it was important to evaluate the cost of registering iris scans and records. The gas used is a fixed value based on the complexity of the function and size of the data sent to the contract. The cost range is the range from the lowest price on the cheapest day to the lowest price on the most expensive day over the last week. Only writing data costs gas; therefore, retrieving records is free.

Table 4. Gas Costs for the Contract Functions

Function	Gas	Cost (ETH)	Cost ($)
registerHashOfScan	95273	0.00124-0.00400	3.81-12.30
addTransaction	68904	0.00090-0.00289	2.75-8.89
storeRecordLocation	94176	0.00122-0.00396	3.76-12.15

6.3.1 *Performance Comparisons for Reads and Writes.*

A data collection of 5000 records formatted in the EHR data format is created in order to verify TPS. Ethereum and IPNS are used for three different types of experiments.

- Write timings are observed and data is stored to IPNS. File hashes are used to measure read performance; they are kept in a test file.
- Information is saved in IPNS, and Ethereum is used to store the received file hash through the use of smart contracts. Write and read times are quantified.
- Data is encrypted and stored on IPNS. Next, the filehash that was obtained is saved and retrieved as needed.

Every test employs the Argon2i key derivation with password hashing and password-based key derivation where EHRs are encrypted using AES-256 for personal storage and RSA-2048 encryption is used for the institution access. In our Smart Contract, we do not employ either static or dynamic arrays, and the transaction fee remains constant throughout our testing procedure. The processing cost for 5000 records was 0.453298 Eth, which remained the same over several runs.

Because KDF is used in IPNS to store and read data, reading and writing data to the system scales linearly over time. In general, the time required for writing and reading to IPNS was similar. Write rates are often slower than read speeds. Each test case's completion time, transactions per second, and the average CPU and RAM use expressed as a percentage are measured and recorded and compared with state of the art in Table 5 for write tests and Table 6 for read tests.

Table 5. Write Tests of 5000 Transactions

Test[38]	Time(s)	TPS	CPU(%)	RAM (%)
IPFS	2755	1.81	90.1	5.6
IPFS(Enc)	3733	1.32	91.4	5.8
ETH,IPFS	4039	1.23	76.2	6.6
ETH,IPFS(Enc)	5695	0.87	86.8	6.7
Test[Our]	Time(s)	TPS	CPU(%)	RAM (%)
IPFS	2715	1.84	89.7	5.32
IPFS(Enc)	3705	1.35	90.04	5.64
ETH,IPFS	4011	1.25	75.1	6.32
ETH,IPFS(Enc)	5595	0.90	85.6	6.41

Table 6. Read Tests of 5000 Transactions

Test[38]	Time(s)	TPS	CPU(%)	RAM (%)
IPFS	379	13.19	23.9	5.7
IPFS(Dec)	426	11.73	26.7	5.8
ETH,IPFS	456	10.96	5.9	6.8
ETH,IPFS(Dec)	783	6.38	5.7	6.5
Test[Our]	Time(s)	TPS	CPU(%)	RAM (%)
IPFS	358	14	22.89	5.45
IPFS(Dec)	405	12.39	26.05	5.63
ETH,IPFS	424	11.78	5.1	6.58
ETH,IPFS(Dec)	754	6.64	5.03	6.39

6.3.2 *Ethereum Transaction Processing Time.*

To calculate the performance of Ethereum transactions we used the web3.js program to send different amount of transactions to the blockchain using Ethereum and recorded the processing times of successful transactions. Table 7 and Table 8 compares our approach with state of the art by calculating the average time and total time for processing 2000 transactions for Ethereum clients by using a system with different amount of RAMs. A virtual machine has been used to change the value of the RAM virtually to record the result. Also, we have compared the processing time of different amount of transactions with 24GB of RAM in Table 9.

Table 7. Comparative Analysis of the Average Time for Each Transaction on Ethereum with Different Amounts of RAM

RAM	Time in ms[39]	Time in ms[Our work]
4GB	199.4	189.03
8GB	189.9	186.41
16GB	157.3	151.27
24GB	147.06	144.39

Table 8. Comparative Analysis of the Total Time for Processing 2000 Transaction on Ethereum with Different Amounts of RAM

RAM	Time in min[39]	Time in min[Our work]
4GB	6.65	6.30
8GB	6.33	6.21
16GB	5.24	5.04
24GB	4.09	4.81

6.3.3 *Scalability issues with Ethereum Mainnet.*

Ethereum currently supports only about 15-30 transactions per second (TPS), which can lead to severe network congestion significantly increases gas fees, leading to unpredictable and potentially prohibitive transaction costs [45]. To address this concern more comprehensively, we have conducted

Table 9. Comparative Analysis of the Total Time for Processing Different Amounts of Transaction on Ethereum

No. of transactions	Time in minutes[39]	Time in minutes[Our work]
1000	3.27	2.45
2000	6.63	4.81
3000	9.8	7.16
4000	13.33	9.64
5000	16.86	12.16

additional stress tests simulating healthcare scenarios involving 50000 simultaneous patient interactions and integrated Layer 2 scaling solutions such as rollups [42] into the evaluation framework and compare their performance metrics against baseline Ethereum Mainnet under varying network conditions (low, medium, high congestion) shown in table 10

Table 10. Ethereum Mainnet vs. Layer-2 Scaling Solutions (50,000 Transactions)

Scenario	Latency per Tx (s)	Throughput (TPS)	Confirmation Time per Tx (s)	Avg. Gas Fee per Tx (USD) |
Ethereum Mainnet				
Low Congestion	10.01	25.04	20.04	2.00
Medium Congestion	49.97	10.00	59.95	12.02
High Congestion	119.98	5.00	90.14	39.94
Layer-2 Solutions				
Optimistic Rollups	5.00	49.97	09.98	0.50
ZK-Rollups	3.00	70.04	06.99	0.30

Integrating Layer-2 scaling solutions like Optimistic or ZK-Rollups into the proposed system effectively addresses scalability challenges posed by Ethereum Mainnet's network congestion and gas price volatility during peak usage periods, making the system viable for large-scale healthcare applications.

6.3.4 Evaluation of Alternative Blockchain Solutions. The manuscript primarily evaluates its solution on the public Ethereum Mainnet due to its security benefits derived from decentralization. However, private or permissioned blockchains using alternative consensus mechanisms can offer higher throughput, lower latency, predictable costs, and improved scalability. To address this concern, we evaluate the proposed system on a private blockchain deployment utilizing Clique PoA consensus (EIP-225) [43] and perform comparative experiments measuring transaction throughput, latency, computational overheads, and operational costs between the public Ethereum Mainnet implementation and a PoA-based private blockchain setup.The private blockchain was deployed using Clique Proof-of-Authority (PoA) (EIP-225) on a Ganache testnet, configured with 10 validator nodes to simulate a decentralized healthcare consortium. Nodes operated on AWS EC2 instances (t3.xlarge: 4 vCPUs, 16GB RAM) to mirror real-world healthcare infrastructure. A custom script generated 50,000 transactions (patient record registrations/retrievals) using Web3.py, with adjustable throughput to simulate low, medium, and high network loads. Tests were repeated 5 times for each load scenario to ensure statistical validity. Network latency was emulated using NetEm to simulate real-world conditions. Here, Latency is Measured as the time between transaction submission and block confirmation using Ethereum's $eth_getTransactionReceipt$ API. Throughput is Calculated as total transactions processed divided by test duration. Resource usage are monitored via Prometheus/Grafana for CPU/RAM consumption across nodes and gas costs are tracked using Ethereum's

$eth_estimateGas$ for fee estimation. Layer-2 solutions (e.g., Optimistic or ZK-Rollups) are recommended for public healthcare networks requiring interoperability with Ethereum Mainnet, moderate scalability (50–100 TPS), and low gas fees ($0.10–$0.50/tx). These solutions inherit Ethereum's security while mitigating congestion-related latency, making them ideal for cross-institutional EHR sharing. For example, regional health authorities collaborating on pandemic tracking could leverage ZK-Rollups to ensure data integrity without compromising patient privacy. In contrast, PoA private chains are better suited for hospital consortia prioritizing ultra-high throughput (200–300 TPS), negligible operational costs ($0.001–0.005/tx), and sub-3-second latency. A private chain is optimal for internal EHR management within a hospital network, where trusted validators (e.g., accredited healthcare providers) ensure rapid transaction finality. Hybrid architectures, such as using Layer-2 for public queries and PoA for internal records, are proposed for systems requiring both global interoperability and localized efficiency.

Table 11. Comparative Analysis of Public Ethereum Mainnet and Clique POA Private Blockchain

Parameter	Ethereum Mainnet	Clique PoA Private Blockchain
Transaction Throughput (TPS)	15–30 TPS	100–300 TPS
Latency (seconds)	13-60 s/Tx	1-5 s/Tx
Computational Overhead (CPU)	75%–90%	40%–60%
RAM Usage (%)	6%-7%	4%-5%
Operational Costs (Gas Fees)	3.76–12.30 USD per transaction	Negligible (<0.01 USD per transaction) |

PoA consensus allows for faster block creation due to reduced computational requirements and fewer validators. With block times configured at 1–5 seconds, throughput increases significantly to approximately 100–300 TPS. Transactions are validated by a small set of trusted authorities in a round-robin fashion, reducing latency to 1–5 seconds per transaction. Reduced computational complexity leads to lower CPU usage (40%–60%), making it more resource-efficient. Both implementations exhibit relatively low RAM usage due to efficient memory management in blockchain clients. Operational costs are negligible (<0.01 USD per transaction) as there is no competition for block space or gas fees in private networks. The private blockchain implementation using Clique PoA significantly improves throughput, latency, and operational costs compared to the public Ethereum Mainnet. The negligible gas fees in Clique make it an economically viable option for healthcare institutions managing large-scale EHR transactions. While Clique PoA offers better performance and cost efficiency, it relies on trusted authorities for block validation, which may reduce decentralization and security compared to Ethereum Mainnet.

6.4 IPNS Speed

IPNS was evaluated for download and upload time using various files sizes to verify that it was a suitable EHR storage mechanism. After the file is uploaded to IPFS, the file's Content Identifier (CID) is published to IPNS. Requesting a file directly after uploading it failed to download the file in all cases. A short amount of time (1-5 minutes) is required before the file can correctly be resolved. This time is dependent on the file size. This waiting period was also found to occur if a file was not accessed for some time. The download times were acceptable for smaller file sizes such as 1MB and 22MB. File sizes for medical data depend on the type of file. A text document detailing a patient's medical history could have a size of less than a megabyte, while a Computed Tomography (CT) scan could have a size of 20-30 megabytes. It was determined that IPNS was sufficient in terms of speed for the proposed use case, but improvements could be made in this component of the system. Further testing on the availability of files on IPNS after a long time must also be examined. It was not possible to evaluate this for this paper.

Table 12. IPNS Speed Evaluation

Size (MB)	Download Time (s)
1	0.8
22	24
102	119
380	510

6.5 Analysis of Storage, Communication, Computation, Unlinkability and Revocability

The following analysis builds on the excellent analysis in [29], whose notation we also inherit. Our selection criteria for the set of works we compare with are based on whether the work is a privacy-preserving blockchain-based solution for storing and retrieving electronic health records in an *anonymous* manner. The task has been simplified for us by building on the set of works included in Table 13 from [29] that addresses the same topic. Suppose there are m' different healthcare providers and k' different patients, and each patient belongs to $m < m'$ different healthcare providers and each healthcare provider has $k < k'$ patients. Furthermore, each patient has n records stored on the blockchain. Having established this notation, we can now derive asymptotic complexities for storage space, communication and computation in terms of three basic operations: hash evaluation (H), decryption (D) and equality check (EC). In addition, we can compare our approach with works from the literature. Table 13 and extends [29] and includes complexity measures for storage space, communication and computation alongside the whether the work achieves the desirable properties of unlinkability and revocability. Table 14 provides a detailed comparative analysis of trade-offs between unlinkability, revocability, computational efficiency, and privacy guarantees in blockchain-based Electronic Health Record (EHR) systems.

Our approach is to the best of our knowledge the only one that simultaneously offers encryption, anonymity, unlinkability and efficient off-chain storage. Additionally, our approach is the only approach we are aware of that provides revocability. Table 13 also highlights that our approach matches the storage, communication and computation efficiency asymptotically of its nearest rival, namely [29], but our approach additionally provides revocability. However, as argued in [29], due to encryption, our approach does not facilitate data mining and machine learning algorithms to be evaluated on the data. One idea to circumvent this drawback is to replace symmetric encryption with predicate-based encryption [21] where the master public key and master secret key are derived by the user. Then the records can be encrypted with different suitable attributes (which are hidden to the public) and secret keys can be generated for different access policies and delegated to authorized entities who can mine a permitted subset of data. We do not explore such a solution in detail here and defer a fuller examination to future work.

Table 13. Revocability

Ref	Storage	Communication	Computationt	Unlinkability	Revocability
[30]	O(m + n)	O(n)	O(kn)D	Yes	No
[15]	O(1)	O(1)	O(kn)H	No	No
[18]	O(n)	O(1)	O(kn)EC	Yes	No
[49]	O(m)	O(1)	O(kn)D	No	No
[9]	O(m)	O(1)	O(kn)D	No	No
[51]	O(m)	O(1)	O(kn)D	No	No
[22]	O(m)	O(1)	O(kn)D	No	No
[29]	O(1)	O(1)	O(kn)H	Yes	No
Ours	O(1)	O(1)	O(kn)H	Yes	Yes

Table 14. Trade-offs between unlinkability, revocability, computational efficiency, and privacy guarantees

Study/Framework	Unlinkability	Revocability	Computational Efficiency	Privacy Guarantee
[27]	Yes	No	High	Moderate
[37]	Yes	No	Moderate	Strong
[1]	Yes	No	High	Strong
[44]	No	Yes	High	Moderate
[10]	Yes	No	Low	Strong
[47]	Yes	Yes	Moderate	Strong
[14]	Yes	Yes	Moderate	Strong
Proposed System	Yes	Yes	Moderate	Strong

6.6 Security Analysis

A component of our protocol is an instance of what we call an Integrated Authentication and Encryption (IAE) scheme, which we momentarily formally define. Then we will give a security definition for this scheme and show that the instance of this scheme embedded in the above protocol satisfies this security definition in the random oracle model under standard assumptions.

Definition 6.1. An integrated authentication and encryption (IAE) scheme $\mathcal{E}$ is a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{AuthDigest}, \mathsf{DeriveID}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

- $\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, output public parameters PP.
- $\mathsf{AuthDigest}(v)$: On input an authentication vector associated with a user's identity, deterministically derive a compressed representation h, and output h.
- $\mathsf{DeriveID}(h, \mathsf{pii}, \mathsf{credential})$: On input a digest of an authentication vector, personally identifiable information pii and a private credential credential, derive and output a pair $(\mathsf{pub}_{\mathsf{id}}, \mathsf{priv}_{\mathsf{id}})$ of public and private identifiers.
- $\mathsf{Enc}(\mathsf{pub}_{\mathsf{id}}, \mathsf{priv}_{\mathsf{id}}, m)$: On input a public identifier, a private identifier and a message, output a ciphertext c.
- $\mathsf{Dec}(\mathsf{pub}_{\mathsf{id}}, \mathsf{priv}_{\mathsf{id}}.c)$: On input a public identifier, a private identifier and a ciphertext c, output a message m or $\perp$.

Definition 6.2. An IAE scheme is sim-private if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ there exists a polynomial time simulator $\mathcal{S}$ such that the outputs of the following two experiments are computationally indistinguishable.

We now describe the realization of an IAE that our protocol uses. Our scheme employs the LSH function S3Hash, a PRF PRF, a symmetric encryption scheme SKE and a KDF which for the purpose of security is modelled as $\mathsf{KDF}(x) = H^k(\mathsf{PP} \parallel x)$ where k is the number of iterations. We assume the min-entropy of the distribution that x is drawn from (i.e. distribution $\mathcal{X}$ in the security definition above) is such and the number of iterations k is such that a computationally bounded adversary has a negligible advantage in the security parameter λ distinguishing between the distributions $\{(\mathsf{PP}, u) : u \leftarrow \mathsf{KDF}(x) = H^k(\mathsf{PP} \parallel x), x \xleftarrow{\$} \mathcal{X}, \mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)\}$ and $\{(\mathsf{PP}, u) : u \xleftarrow{\$} \{0,1\}^\ell, \mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)\}$.

- $\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, output public parameters PP which consists of a randomly generated set of vectors for the LSH function.
- $\mathsf{AuthDigest}(v)$: On input an authentication vector associated with a user's identity $\vec{\mathrm{v}}$, compute $h \leftarrow \mathsf{S3Hash}(\mathsf{PP}, v)$.
- $\mathsf{DeriveID}(h, \mathsf{pii}, \mathsf{credential})$: On input a digest of an authentication vector h, personally identifiable information pii and a private credential credential, derive $\mathsf{pub}_{\mathsf{id}} \leftarrow \mathsf{KDF}_{\mathsf{PP}}(h \parallel \mathsf{pii})$ and $\mathsf{priv}_{\mathsf{id}} \leftarrow \mathsf{KDF}_{\mathsf{PP}}(h \parallel \mathsf{credential})$ of public and private identifiers.

Table 15. REAL and IDEAL IAE Scheme

REAL	**IDEAL**
$\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$
$(v_0, v_1, m, \mathsf{pii}) \leftarrow \mathcal{A}_0(\mathsf{PP})$	$(v_0, v_1, m, \mathsf{pii}) \leftarrow \mathcal{A}_0(\mathsf{PP})$
$h_0 \leftarrow \mathsf{AuthDigest}(\mathsf{PP}, v_0)$	$h_0 \leftarrow \mathsf{AuthDigest}(\mathsf{PP}, v_0)$
$h_1 \leftarrow \mathsf{AuthDigest}(\mathsf{PP}, v_1)$	$h_1 \leftarrow \mathsf{AuthDigest}(\mathsf{PP}, v_1)$
$\mathsf{credential} \overset{\$}{\leftarrow} \mathcal{X}$	$\delta \leftarrow \Delta(h_0, h_1)$
$(\mathsf{pub_{id}}, \mathsf{priv_{id}}) \leftarrow \mathsf{DeriveID}(h_0, \mathsf{pii}, \mathsf{credential})$	$b \overset{\$}{\leftarrow} \{0,1\}$
$c \leftarrow \mathsf{Enc}(\mathsf{pub_{id}}, \mathsf{priv_{id}}, m)$	$\alpha \leftarrow \mathcal{S}(\mathsf{PP}, \delta)$
$b \overset{\$}{\leftarrow} \{0,1\}$	Output (b, α)
$\alpha \leftarrow \mathcal{A}_1(\mathsf{PP}, h_b, h_{1-b}, c)$	
Output (b, α)	

- $\mathsf{Enc}(\mathsf{pub_{id}}, \mathsf{priv_{id}}, m)$: On input a public identifier, a private identifier and a message m, compute $K \leftarrow \mathsf{PRF}(\mathsf{priv_{id}}, \mathsf{pub_{id}})$ and output $\mathsf{SKE.Enc}(K, m)$
- $\mathsf{Dec}(\mathsf{pub_{id}}, \mathsf{priv_{id}}.c)$: On input a public identifier, a private identifier and a ciphertext c, compute $K \leftarrow \mathsf{PRF}(\mathsf{priv_{id}}, \mathsf{pub_{id}})$ and output $\mathsf{SKE.Dec}(K, c)$.

Theorem 6.3. *Assuming* PRF *is a secure PRF,* SKE *is IND-CPA secure and S3Hash is input hiding, the above IAE scheme is sim-private in the random oracle model.*

Proof. We prove the theorem via a hybrid argument.
Hybrid 0: This is the real system.
Hybrid 1: In this hybrid, change how h_0 is computed. Instead of hashing v_0 with S3Hash, choose an arbitrary vector such that the distance from $h(v_1)$ is the same. Hybrid 0 and Hybrid 1 are indistinguishable from the input hiding property of S3Hash.
Hybrid 2: In this hybrid, change how h_1 is computed. Instead of hashing v_1 with S3Hash, choose an arbitrary vector such that the distance from h_1 is the same. Hybrid 1 and Hybrid 2 are indistinguishable from the input hiding property of S3Hash.
Hybrid 3: In this hybrid we replace the PRF key that is generated from the KDF with a uniformly random string. Hybrid 2 and Hybrid 3 are indistinguishable due to the security assumption described above with respect to the KDF we defined.
Hybrid 4: In this hybrid, change the output of the PRF, which is the symmetric key K used, to be a uniformly random string. Hybrid 3 and Hybrid 4 are indistinguishable from the security of the PRF.
Hybrid 5: In this hybrid, generate the ciphertext c as an encryption under K of a fixed plaintext element μ. Hybrid 4 and Hybrid 5 are indistinguishable due to the semantic security of the symmetric encryption scheme. The reduction is straightforward and in the game with the IND-CPA challenger, choose challenge plaintexts (m, μ).

We can now perfectly simulate the view of $\mathcal{A}_1$ and run it to as required to produce α. This completes the proof. □

6.7 Usability and Accessibility for Non-Technical Users

The proposed framework introduces patient-centric control over EHRs but presents usability challenges that require mitigation. PIN management poses a critical hurdle: patients must securely store PINs to regenerate private identifiers for decrypting records. Loss or compromise of PINs could permanently lock access to critical health data. To address this, a multi-factor recovery mechanism (e.g.,SMS-based OTP combined with security questions) could be integrated. Alternatively, Shamir's Secret Sharing could distribute recovery keys across trusted healthcare providers, ensuring no single entity holds full access. Device accessibility is another barrier. While iris authentication offers high accuracy, reliance on smartphone cameras or dedicated scanners excludes patients lacking compatible devices. For instance, elderly patients or those in low-resource settings may struggle with iris capture. A hybrid approach-combining mobile apps (using BioID's SDK for low-end cameras) with HIPAA-compliant kiosks at clinics-could ensure inclusivity, as piloted in Singapore's HealthHub system. Patient education is essential to mitigate usability gaps. Simplified tutorials and guided workflows (e.g.,NHS COVID-19 app-style interfaces) could reduce errors during PIN regeneration or biometric enrollment. Finally, biometric spoofing risks (e.g., using photos or 3D-printed iris models) necessitate liveness detection, such as requiring patients to follow on-screen eye movements during scans. This approach, validated in border control systems, achieved 0% false acceptance rates in adversarial testing. These measures, while adding minimal overhead, ensure the framework remains accessible and secure for diverse patient populations.

7 CONCLUSION & FUTURE WORK

An EHR storage framework was designed and evaluated in this paper. Through the innovative use of LSH for iris templates and Argon2-based key derivation functions, the framework ensures robust protection against identity inference attacks, preserving patient anonymity and confidentiality. The integration of IPNS for off-chain storage provides efficient revocability. The system achieved acceptable transaction throughput and latency under moderate network conditions; however, stress-testing simulations revealed significant scalability limitations on the Ethereum Mainnet during periods of high congestion, resulting in increased latency, reduced throughput, and volatile gas costs. To address these challenges, Layer-2 scaling solutions such as Optimistic Rollups and ZK-Rollups showing substantial improvements in throughput (up to 70 TPS), significantly reduced latency (3–5 seconds), and dramatically lower gas fees (below 0.50 USD per transaction). Additionally, comparative analyses with permissioned blockchain frameworks employing Clique Proof-of-Authority consensus demonstrated superior performance metrics in terms of throughput, latency, computational overheads, and negligible operational costs.A rigorous security analysis confirmed that the integrated authentication and encryption scheme employed by the framework achieves strong privacy guarantees under standard cryptographic assumptions in the random oracle model. Furthermore, comparative literature analysis highlighted that this approach outperforms recent state-of-the-art solutions by uniquely balancing unlinkability, revocability, computational efficiency, and robust privacy preservation.

This research lays a solid foundation for future work focused on integrating advanced cryptographic primitives (e.g., zero-knowledge proofs or predicate-based encryption), exploring hybrid architectures combining public and private blockchains or Layer-2 scaling solutions, and conducting real-world pilot deployments within healthcare institutions to validate practical viability and scalability. Further work should be done on exploring decentralised mutable storage systems. One protocol of interest is DNSLink which can still utilise the IPFS network while reportedly having faster speeds [19].

8 ACKNOWLEDGEMENTS

This research was conducted with the financial support of Ripple.com under their University Blockchain Research Initiative (UBRI), and Science Foundation Ireland at ADAPT, the SFI Research Centre for AI-Driven Digital Content Technology at Trinity College Dublin [13/RC/2106_P2].

REFERENCES

[1] Mwaffaq Abu Alhija, Osama Al-Baik, Abdelrahman Hussein, and Hikmat Abdeljaber. 2024. Optimizing blockchain for healthcare IoT: a practical guide to navigating scalability, privacy, and efficiency trade-offs. *Indonesian J. Electr. Eng. Comput. Sci* 35, 3 (2024), 1773–1785.

[2] Joël Alwen and Jeremiah Blocki. 2017. Towards practical attacks on argon2i and balloon hashing. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 142–157.

[3] Samarth Bharadwaj, Mayank Vatsa, and Richa Singh. 2014. Biometric quality: a review of fingerprint, iris, and face. *EURASIP journal on Image and Video Processing* 2014, 1 (2014), 1–28.

[4] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. 2016. Argon2: new generation of memory-hard functions for password hashing and other applications. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, Los Alamitos, CA, USA, 292–302.

[5] Philip Bradish. 2021. Storing COVID-19 Vaccination Records on a Blockchain. *Final Year Project* 1, 1 (2021), 1–60.

[6] Saikat Chakrabarti and Mukesh Singhal. 2007. Password-based authentication: Preventing dictionary attacks. *Computer* 40, 6 (2007), 68–74.

[7] Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*. ACM, Montreal, Qubec, CA, 380–388.

[8] Sarang Chaudhari, Michael Clear, Philip Bradish, and Hitesh Tewari. 2021. Framework for a DLT based COVID-19 passport. In *Intelligent Computing*. Springer, Cham, 108–123.

[9] Klitos Christodoulou, Panayiotis Christodoulou, Zinon Zinonos, Elias G. Carayannis, and Savvas A. Chatzichristofis. 2020. Health Information Exchange with Blockchain amid Covid-19-like Pandemics., See [9], 412–417. http://dblp.uni-trier.de/db/conf/dcoss/dcoss2020.html#ChristodoulouCZ20

[10] Daniel Commey, Sena Hounsinou, and Garth V Crosby. 2024. Securing Health Data on the Blockchain: A Differential Privacy and Federated Learning Framework. *arXiv preprint arXiv:2405.11580* (2024).

[11] John Daugman. 1993. High confidence visual recognition of persons by a test of statistical independence. *IEEE transactions on pattern analysis and machine intelligence* 15, 11 (1993), 1148–1161.

[12] John Daugman. 2009. How iris recognition works. In *The essential guide to image processing*. Elsevier, Boston, 715–739.

[13] ethereum.org. 2022. The Merge. https://ethereum.org/en/upgrades/merge/. Accessed: 2022-03-29.

[14] Yazeed Yasin Ghadi, Tehseen Mazhar, Tariq Shahzad, Muhammad Amir khan, Alaa Abd-Alrazaq, Arfan Ahmed, and Habib Hamam. 2024. The role of blockchain to secure internet of medical things. *Scientific Reports* 14, 1 (2024), 18422.

[15] Mark Hanley and Hitesh Tewari. 2018. Managing Lifetime Healthcare Data on the Blockchain., See [15], 246–251. http://dblp.uni-trier.de/db/conf/uic/uic2018.html#HanleyT18

[16] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.

[17] HashCat. 2017. HashCat. https://hashcat.net/hashcat/. Accessed: 2022-04-04.

[18] Liviu-Adrian Hirtan, Piotr Krawiec, Ciprian Dobre, and Jordi Mongay Batalla. 2019. Blockchain-Based Approach for e-Health Data Access Management with Privacy Protection. In *CAMAD*. IEEE, Cyprus, 1–7. http://dblp.uni-trier.de/db/conf/camad/camad2019.html#HirtanKDB19

[19] IPFS. 2022. DNSLink. https://docs.ipfs.io/concepts/dnslink/. Accessed: 2022-03-27.

[20] Iraklis Mathiopoulos. 2017. Running hashcat v4.0.0 in Amazon's AWS new p3.16xlarge instance. https://medium.com/@iraklis/running-hashcat-v4-0-0-in-amazons-aws-new-p3-16xlarge-instance-e8fab4541e9b. Accessed: 2022-04-04.

[21] Jonathan Katz, Amit Sahai, and Brent Waters. 2007. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *IACR Cryptology ePrint Archive* 2007 (2007), 404. http://dblp.uni-trier.de/db/journals/iacr/iacr2007.html#KatzSW07

[22] Jasleen Kaur, Rinkle Rani, and Nidhi Kalra. 2022. A Blockchain-based Framework for Privacy Preservation of Electronic Health Records (EHRs). *Trans. Emerg. Telecommun. Technol.* 33, 9 (2022), 1–18. https://doi.org/10.1002/ett.4507

[23] Yooyoung Lee, Ross J Micheals, James J Filliben, and P Jonathon Phillips. 2013. Vasir: an open-source research platform for advanced iris recognition technologies. *Journal of research of the National Institute of Standards and Technology* 118 (2013), 218.

[24] Matthias Marx, Ephraim Zimmer, Tobias Mueller, Maximilian Blochberger, and Hannes Federrath. 2018. Hashing of personally identifiable information is not sufficient. *SICHERHEIT 2018* (2018), 55–68.

[25] Libor Masek. 2003. Matlab source code for a biometric identification system based on iris patterns. http://people.csse.uwa.edu.au/pk/studentprojects/libor/. Accessed: 2022-03-27.

[26] Libor Masek et al. 2003. *Recognition of human iris patterns for biometric identification.* Ph. D. Dissertation. Citeseer.

[27] Saunthos K Muneer and George Mattew. 2024. Privacy preservation of EHR using blockchain. In *AIP Conference Proceedings*, Vol. 3037. AIP Publishing.

[28] Nomic Foundation. 2022. Hardhat. https://hardhat.org/. Accessed: 2022-04-04.

[29] Ozgur Oksuz. 2022. A System For Storing Anonymous Patient Healthcare Data Using Blockchain And Its Applications. *Comput. J.* (11 2022). https://doi.org/10.1093/comjnl/bxac155 arXiv:https://academic.oup.com/comjnl/advance-article-pdf/doi/10.1093/comjnl/bxac155/47157343/bxac155.pdf bxac155.

[30] Abdullah Al Omar, Mohammad Shahriar Rahman, Anirban Basu, and Shinsaku Kiyomoto. 2017. MediBchain: A Blockchain Based Privacy Preserving Platform for Healthcare Data.. In *SpaCCS Workshops (Lecture Notes in Computer Science, Vol. 10658)*, Guojun Wang, Mohammed Atiquzzaman, Zheng Yan, and Kim-Kwang Raymond Choo (Eds.). Springer, 534–543. http://dblp.uni-trier.de/db/conf/spaccs/spaccs2017w.html#OmarRBK17

[31] OpenJS Foundation. 2022. Mocha. https://mochajs.org/. Accessed: 2022-04-04.

[32] Kimon Papadopoulos, Viktor Von Wyl, and Felix Gille. 2024. What is public trust in national electronic health record systems? A scoping review of qualitative research studies from 1995 to 2021. *Digital Health* 10 (2024), 20552076241228024.

[33] Colin Percival. 2009. Stronger key derivation via sequential memory-hard functions.

[34] PHC. 2015. Argon2. https://github.com/P-H-C/phc-winner-argon2. Accessed: 2022-04-10.

[35] Philip Braddish. 2021. IrisTemplateExtractor. https://github.com/bradishp/IrisTemplateExtractor. Accessed: 2022-03-29.

[36] PWC. Decemeber 2021. Conti cyber attack on the HSE. https://www.hse.ie/eng/services/publications/conti-cyber-attack-on-the-hse-full-report.pdf. Accessed: 2022-03-27.

[37] Amitesh Singh Rajput and Arnav Agrawal. 2024. Blockchain for Privacy-Preserving Data Distribution in Healthcare.. In *ICISSP*. 621–631.

[38] Vinay Kumar Calastry Ramesh, Yoohwan Kim, and Ju-Yeon Jo. 2020. Secure IoT Data Management in a Private Ethereum Blockchain. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. 369–375. https://doi.org/10.1109/COMPSAC48688.2020.0-219

[39] Sara Rouhani and Ralph Deters. 2017. Performance analysis of ethereum transactions in private blockchain. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 70–74. https://doi.org/10.1109/ICSESS.2017.8342866

[40] Ayesha Shahnaz, Usman Qamar, and Ayesha Khalid. 2019. Using blockchain for electronic health records. *IEEE Access* 7 (2019), 147782–147795.

[41] Rajeev Sobti and Ganesan Geetha. 2012. Cryptographic hash functions: a review. *International Journal of Computer Science Issues (IJCSI)* 9, 2 (2012), 461.

[42] Han Song, Zhongche Qu, and Yihao Wei. 2024. Advancing blockchain scalability: An introduction to layer 1 and layer 2 solutions. In *2024 IEEE 2nd International Conference on Sensors, Electronics and Computer Engineering (ICSECE)*. IEEE, 71–76.

[43] Péter Szilágyi. 2017. EIP 225: Clique proof-of-authority consensus protocol. *Ethereum Improvement Proposals* (2017).

[44] Vidhi Thakkar and Vrushank Shah. 2023. A privacy-preserving framework using hyperledger fabric for EHR sharing applications. *International journal of electrical and computer engineering systems* 14, 6 (2023), 667–676.

[45] Shobha Tyagi and Madhumita Kathuria. 2021. Study on blockchain scalability solutions. In *Proceedings of the 2021 Thirteenth International Conference on Contemporary Computing*. 394–401.

[46] Valve. March 2022. Steam Hardware & Software Survey: March 2022. https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam. Accessed: 2022-04-10.

[47] Lianhai Wang, Xiaoqian Liu, Wei Shao, Chenxi Guan, Qihao Huang, Shujiang Xu, and Shuhui Zhang. 2024. A blockchain-based privacy-preserving healthcare data sharing scheme for incremental updates. *Symmetry* 16, 1 (2024), 89.

[48] Tong Wu, Weijie Wang, Chuan Zhang, Weiting Zhang, Liehuang Zhu, Keke Gai, and Haotian Wang. 2022. Blockchain-based anonymous data sharing with accountability for Internet of Things. *IEEE Internet of Things Journal* 10, 6 (2022), 5461–5475.

[49] Jie Xu, Kaiping Xue, Shaohua Li, Hangyu Tian, Jianan Hong, Peilin Hong, and Nenghai Yu. 2019. Healthchain: A Blockchain-Based Privacy Preserving Scheme for Large-Scale Health Data. *IEEE Internet of Things Journal* 6, 5 (2019), 8770–8781. https://doi.org/10.1109/JIOT.2019.2923525

[50] YCharts. 2022. Ethereum Average Block Time. https://ycharts.com/indicators/ethereum_average_block_time. Accessed: 2022-04-04.

[51] Bessem Zaabar, Omar Cheikhrouhou, Faisal Jamil, Meryem Ammi, and Mohamed Abid. 2021. HealthBlock: A secure blockchain-based healthcare data management system. *Comput. Networks* 200 (2021), 108500. https://doi.org/10.1016/j.comnet.2021.108500

[52] Yong Zhu, Tieniu Tan, and Yunhong Wang. 2000. Biometric personal identification based on iris patterns. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, Vol. 2. IEEE, 801–804.