Storing COVID-19 Vaccination Records on a Blockchain

Philip Bradish

August 9, 2021

A dissertation submitted in partial fulfilment of the requirements for the degree of Master in Computer Science (MCS)
Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

Signed: ________________________  Date: ________________________
Abstract

As the world grows more globalised the threat of pandemics increases. Widespread international travel allows diseases to rapidly spread worldwide. The COVID-19 pandemic has highlighted how unprepared we are to combat this threat. One approach which shows promise at preventing the spread of diseases is the employment of vaccinations to produce immunity.

However, one problem with immunisation is that it can be difficult to verify whether someone has been vaccinated. This is exacerbated when travelling across different jurisdictions. An internationally recognised vaccination passport could solve this issue by allowing users to present their vaccination records when required. Using a blockchain to implement this passport would have several key advantages over the alternatives in terms of transparency, decentralisation and immutability.

This dissertation implements and assesses the viability of a previously proposed blockchain based vaccination passport. The system is permissioned so that only authorised entities can write to it. However, it is publicly readable so that users can present their vaccination records to whomever they wish. The blockchain makes use of an iris-based identification system. The iris scans are hashed before being stored on the blockchain so that they cannot be linked back to the user. A locality sensitive hashing algorithm is employed to ensure that different scans of the same iris produce similar hashes. A unique identifier is created for each user by combining their stored hashed iris scan with some of their personal information. This identifier is stored alongside their records. When a user wants to retrieve their records, they must first locate their stored hashed iris scan and then use it to recreate their identifier.

A prototype of the system is built using an Ethereum blockchain. An open-source iris recognition system is employed and adapted to the application. Improvements have been made to its speed and accuracy. The prototype has been tested using three different iris image datasets. It is found to achieve very promising results in terms of its speed and memory usage. Additionally, high matching accuracy is reported on all the datasets.
Acknowledgements

First and foremost, I would like to thank my supervisor Hitesh Tewari and my assistant supervisor Michael Clear for their support. I would also like to thank Sarang Chaudhari for helping me understand the iris extraction component of the project.

Portions of the research in this paper use the CASIA-IrisV4 collected by the Chinese Academy of Sciences’ Institute of Automation. Additionally, portions of the work are tested using the IITD Iris Database version 1.0.

I am grateful to my peers who have helped me survive the last five years and encouraged me to succeed. Finally, I would like to thank my parents for their constant support and endless patience.
Publications

Contents

1 Introduction 1
   1.1 Objective ............................................. 3
   1.2 Guide to Chapters .................................... 3

2 Background 5
   2.1 Iris Authentication Systems ........................... 5
      2.1.1 John Daugman .................................. 6
      2.1.2 Libor Masek .................................... 7
      2.1.3 NIST VASIR ..................................... 10
      2.1.4 Ordinal Measures ................................ 11
      2.1.5 Rotation Invariant Templates .................... 11
   2.2 Locality Sensitive Hashing ............................ 12
      2.2.1 Random Projection Hashing ....................... 13
      2.2.2 TLSH ............................................. 14
   2.3 Blockchains .......................................... 16
      2.3.1 Bitcoin ......................................... 16
      2.3.2 Ethereum ........................................ 17
      2.3.3 Hyperledger ...................................... 19
   2.4 Related Work ......................................... 20
      2.4.1 COVID-19 Antibody Test/Vaccination Certification .... 20
      2.4.2 Immunity Certificate ............................. 21

3 Design 23
   3.1 System Overview ....................................... 23
   3.2 Application Workflow .................................. 24

4 Implementation 27
   4.1 Iris Template Extraction ............................. 27
      4.1.1 Converting the Template to a Vector and Accounting for Rotational Inconsistencies .......... 27
List of Figures

2.1 CASIA-Iris-Interval image ‘S1001L01’ Segmented with Masek’s System . 8
2.2 Masek’s Method of Normalising an Iris . . . . . . . . . . . . . . . . . . . . . 9
2.3 TLSH Algorithm [1] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.1 High-Level Overview of System . . . . . . . . . . . . . . . . . . . . . . . . . 24
3.2 Flow Chart of System Workflow [1] . . . . . . . . . . . . . . . . . . . . . . . 26

4.1 Template Conversion and Rotation Process . . . . . . . . . . . . . . . . . . 28
4.2 Original and Updated Noise Detection Algorithms on CASIA-Iris-Interval
  Image ‘S1001L08’ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

5.1 Number of Performed Rotations Comparison for CASIA-Iris-Interval . 44
5.2 ROC curves of Different Approaches on CASIA-Iris-Interval . . . . . . 45
5.3 ROC curves of Different Approaches on CASIA-Iris-Syn . . . . . . . . 46

A1.1 Number of Performed Rotations Comparison for CASIA-Iris-Syn . . . 63
A1.2 Number of Performed Rotations Comparison for IITD Irises . . . . . 64
List of Tables

5.1 Irises Successfully Segmented in the Datasets ........................................ 42
5.2 Angular and Radial Resolution Comparison for CASIA-Iris-Interval ........... 43
5.3 Comparison of Different Approaches on CASIA-Iris-Interval ................... 44
5.4 FAR and FRR of CASIA-Iris-Interval Hashed Templates .......................... 45
5.5 Comparison of Different Approaches on CASIA-Iris-Syn ........................... 46
5.6 FAR and FRR of CASIA-Iris-Syn Hashed Templates ................................. 47
5.7 Comparison of Different Approaches on IITD Irises ................................. 47
5.8 FAR and FRR of IITD Irises Hashed Templates ........................................ 47
5.9 Decidability Index of Datasets when using Hashing Parameters Trained on Different Datasets ................................................................. 48
5.10 Time Taken to Extract Iris Templates from Datasets .............................. 49
5.11 Time Taken to Search Blockchain and Lookup Accuracy for Datasets ...... 50
5.12 Overview of Results .................................................................................. 53

A1.1 Angular and Radial Resolution Comparison for CASIA-Iris-Syn ............... 63
A1.2 Angular and Radial Resolution Comparison for IITD Irises ..................... 64
1 Introduction

The rate at which diseases can spread across the world has grown rapidly in recent decades due to increased globalism and travel. This greatly increases the likelihood of pandemics. The COVID-19 pandemic has highlighted how ill-equipped we are to prevent and counter this threat. However, the employment of vaccinations on a large scale is one approach that shows promise. Widespread immunisation has been one of the greatest achievements in modern medicine. It is estimated by the World Health Organisation (WHO) that immunisation prevents 2-3 million deaths every year and with improved coverage could prevent a further 1.5 million [2].

One open problem associated with immunisation is reliably proving whether someone has or has not been vaccinated. This is currently a major issue as COVID-19 vaccinations are being distributed across the world. The problem is further exacerbated when travelling across different jurisdictions. An internationally recognised vaccination passport could solve this issue. It would enable users to present their records to whomever they wish in a globally accepted and verifiable format.

A vaccination passport could be implemented in a variety of different ways. However, conventional approaches suffer from fundamental security and usability flaws. Traditional paper-based vaccination certificates are prone to getting lost and are susceptible to forgery. Smart cards are equally liable to get lost and can contain vulnerabilities that are difficult to fix once they have been deployed. Several organisations are currently utilising smartphones. For example, the Commons Project and the World Economic Forum are developing a smartphone application called CommonPass which allows users to access their records [3]. However, smartphone-based systems have their own security vulnerabilities and discriminate against financially disadvantaged people. Another project called the Vaccination Credential Initiative which is backed by companies such as Oracle and Microsoft attempts to counter the different disadvantages by combining various approaches [4]. Participants can choose between using a smartphone or a printed certificate with a scannable quick response (QR) code. However, this does not fully address the inherent issues and it increases the size of the system’s attack surface. All these methods have the additional disadvantage of being
reliant on a centralised body to administrate and control them. Such centralised bodies are susceptible to misuses of power and as a result are often widely mistrusted. They also act as a central point of attack and failure.

An online passport serves as a viable alternative to these approaches. It avoids a lot of the previously mentioned issues by circumventing the need for users to carry around a physical representation of their records. This makes the system less intrusive to users and alleviates the fear of it being used to track and monitor people. Furthermore, using a blockchain to implement the online passport removes the need for a centralised body to store and administrate the records. A blockchain based vaccination passport is proposed by Chaudhari et al. [1]. The blockchain is configured so that only trusted entities can write to it. This provides a guarantee that all vaccination records stored on the blockchain are authentic. However, it is readable by everyone so that users have the freedom to present their records to whomever they wish. Such a system does not require an energy inefficient proof-of-work (PoW) consensus mechanism. Instead, a far less computationally intensive proof-of-authority (PoA) consensus mechanism can be employed with the trusted entities acting as the block signers. Therefore, the system is trustworthy so long as the trusted entities behave in an honest manner. A blockchain also provides a high level of transparency so it would be easy to catch any misbehaving entities.

A blockchain based online vaccination passport has its own obstacles that need to be addressed. Predominant among these is the need for a mechanism which can facilitate users storing and accessing their records without leaking any personally identifiable information (PII). Authenticating using a password is unsuitable as the decentralised and immutable nature of blockchains makes it impossible for a user to manually reset it if they forget it. Using a cryptographic key would face the same issue if they lost it. A biometric identification system offers a more viable alternative since it does not require a reset mechanism provided the biometric is stable throughout life. Biometric authentication systems have seen a massive growth in popularity in recent years. There are many different types of biometric authentication systems. An iris-based system is particularly suitable for this use case. Iris patterns are stable throughout adult life and highly unique to individuals. Several studies have shown iris-based authentication systems can be very accurate [5, 6, 7, 8].

If the iris templates were stored directly on the blockchain they could be used to compromise other iris-based authentication systems. It would also make it easier for attackers to link users and their vaccination records. Instead, the iris templates are hashed before being stored. Different scans of the same iris will probably produce slightly different iris templates. A locality sensitive hashing algorithm is employed to ensure that similar iris templates will have similar hashes. When a user first
presents themself to the system, they will save a copy of their hashed iris scan on the blockchain. They will then create a unique identifier by hashing a combination of this iris hash and some personal information. This unique identifier is stored alongside their records. In subsequent interactions with the system the user provides a new iris scan. It is hashed and compared to the stored hashes to find similar ones. The user can hash their personal information with each of these similar hashes. If they generate a unique identifier that matches one stored on the blockchain, then they know it must be their identifier. At this point they can retrieve their records or let a verified entity add new ones.

1.1 Objective

The primary objective of this study is to create a prototype of the blockchain based vaccination passport proposed by Chaudhari et al. [1] and assess its viability. The system consists of several components. An iris template extraction algorithm is used to create iris feature vectors from images of irises. A locality sensitive hashing algorithm is employed to hash these templates. A blockchain is built to store the hashes and the vaccination records. Users are able to access the records and store new ones. The prototype will prove the system is viable if it adheres to the following properties.

- The memory required to store the fully synced blockchain should be feasible for an inexpensive machine even when the blockchain has been running for a significant amount of time and there are a large number of records stored on it.
- The time required to locate a user’s records on the blockchain should be reasonable. This includes the time to extract the user’s iris template, hash it, search for similar hashes and recreate the correct identifier.
- The blockchain should be able to correctly identify a user in the vast majority of cases.
- The blockchain should never misidentify a user.

1.2 Guide to Chapters

Chapter 2 of this dissertation will describe relevant background information and related state of the art research. It will discuss iris identification systems, locality sensitive hashing and blockchains. It will also investigate some other work focused on using blockchains to store vaccinations records. Chapter 3 will provide a high-level overview of the proposed blockchain based vaccination passport’s design. It outlines the major components of the system and the relationships between them. Chapter 4
will discuss how a prototype of the system was implemented. Chapter 5 will then present an evaluation of the performance and accuracy of this prototype. It will include an analysis of the speed of the system, its memory usage, its security and its matching accuracy. Finally, chapter 6 will conclude the dissertation by providing a summary of the research and identifying areas for future work.
2 Background

This chapter discusses relevant background information and the related state of the art. First iris authentication systems are examined. Then information is provided on locality sensitive hashing. An overview of the fundamentals of blockchain is supplied along with some information about different blockchain variants. Finally, some related work around using blockchains to store vaccination records is investigated.

2.1 Iris Authentication Systems

Developing reliable and secure authentication systems has been a long-standing problem in computer systems. Traditional password-based authentication systems have many known flaws. Foremost among these is the requirement for users to remember their passwords. This often results in users choosing weak passwords or continuously reusing the same ones across multiple domains. This can greatly weaken the security of systems. Passwords are especially unsuitable for publicly readable blockchains. This is because attackers can download the system and perform offline dictionary attacks against it. Additionally, the immutable and decentralised nature of blockchains makes it impossible for users to reset their password in the event they forget it. As a result, cryptographic keys are often used for blockchain authentication. However, cryptographic keys are lacking in portability and also cannot be recovered if they are lost.

There has been a huge surge of interest and research in biometric based authentication systems in recent years. They have several distinct advantages over traditional methods. Most notably, they do not require the user to remember or store anything. This makes them significantly more portable. Additionally, if the biometric is stable there is no need for a reset mechanism. Authentication systems have been built using many different biometrics including fingerprints, facial features and retina scans. Each biometric has its own accuracy and security guarantees.

An iris-based recognition system was found to be particularly suitable for this system. Irises are stable from a young age. The formation of an individual’s iris pattern
is random and not related to their genetics [5]. They are highly unique to individu-
als and there are reportedly $10^{72}$ possible iris patterns [9]. They are hard to steal as a
thief would need to acquire a close-up high-resolution image of someone’s eye. Addition-
ally, various iris authentication systems have been shown to be extremely accurate
[5, 6, 7, 8].

An iris authentication system typically consists of four stages. Either an image or a
video of an iris is inputted into the system. In the case of a video a single frame is
usually selected. The first stage of the authentication system involves segmenting the
image so that the iris can be located and separated from the pupil, sclera and eyelids.
During this stage sections of the iris that are blocked by eyelashes or corrupted by
reflecting light can be flagged. The iris must then be normalised to a form with con-
stant dimensions. Special care must be taken to ensure that the same iris tissue will be
mapped to the same part of the normalised form regardless of imaging conditions and
pupil dilation. The fact that the pupil is not always concentric to the iris further com-
plicates this process [5]. The third stage consists of extracting the most distinguishing
features of the iris. This should also convert it to a form that is independent of imaging
conditions such as lighting. These first three stages produce a template which repre-
sents the iris. A matching algorithm can then be employed to compare the similarity
of different iris templates.

### 2.1.1 John Daugman

John Daugman, a researcher at the University of Cambridge was the first person to
successfully create a working iris authentication system [6]. His system was initially
patented [10] which slowed down further research. However, the patent has now
expired. Daugman’s system is often considered the most successful and widely used
iris authentication system.

Daugman uses integro-differential operators to search the inputted image for the inner
and outer boundaries of the iris. These boundaries are represented by circles and are
placed at the locations where the maximum change in pixel value occurs. The outer
edge of the iris is located first. To speed up the process the search for the inner edge
can be performed within the previously identified iris region as the pupil will always
be inside the iris.

Daugman achieves normalisation by remapping each point in the iris based on its
angle relative to the centre of the pupil and its relative position between the iris’ inner
and outer boundaries. Each point is effectively remapped to a pair of dimensionless
real polar coordinates $(r, \theta)$ where $r$ lies on the interval $[0, 1]$ and $\theta$ on the interval
$[0, 2\pi]$. This remapping of the iris region from $(x, y)$ Cartesian coordinates to the new
normalised coordinate system can be represented as in Equation 1, where \( x(r, \theta) \) and \( y(r, \theta) \) are defined as the coordinates at an angle \( \theta \) and a percentage of \( r \) between the inner and outer iris boundaries. However, this process does not account for the fact that the angle between a region of the iris and the centre of the pupil can vary. This happens due to rotational inconsistencies which are caused by tilts of the head or camera or rotations of the eye itself within the eye socket. Rotational inconsistencies are instead handled in the matching stage.

\[
I(x(r, \theta), y(r, \theta)) \rightarrow I(r, \theta)
\]  

Daugman extracts the most distinguishing features of an iris using two-dimensional Gabor filters. Gabor filters are useful as they can represent a signal in terms of both space and frequency. The normalised image is convolved with two-dimensional Gabor filters. According to Bowyer et al. [11] Daugman quantises the filter’s phase response based on its real and imaginary components. Two bits are used to represent each complex coefficient. The first bit is equal to 1 if its real part is positive and otherwise is equal to 0. Similarly, the second bit is equal to 1 if its imaginary part is positive and otherwise is equal to 0. This process produces a binary template.

To compare the similarity of two templates the hamming distance between them is calculated [12]. Naturally the lower the value the more similar the templates. If the hamming distance is lower than a specified threshold the two templates are considered to be from the same iris. Each template has a corresponding mask which flags the areas that were affected by noise. This includes parts of the iris obstructed by eyelashes, eyelids and reflecting light. In order to accurately calculate the similarity, the comparison only considers the bits that are not corrupted in either template. This is done by combining both template’s masks and only comparing the bits which are not classified as invalid in either of them. The rotational inconsistencies are accounted for by rotating one of the templates in both directions along the angular dimension. Its hamming distance to the other template is taken at each orientation and the smallest of these values is used when deciding if the two templates are from the same iris. The reason for this is that the minimum value should correspond to the correct alignment between the two templates.

2.1.2 Libor Masek

Libor Masek developed his iris authentication system as part of his final year project in the University of Western Australia in 2002 [5]. He was motivated by the absence of an open source iris authentication system. Consequently he made his program publicly available [13]. Masek opted to write his code in Matlab so that he could utilise its
advanced image processing toolbox. Masek largely based his work on Daugman’s. He also used some ideas from other researchers and made some innovations of his own.

Masek locates the iris in the image using two circular Hough transforms to get its inner and outer boundaries [5]. To accomplish this, Canny edge detection is used to generate an edge map. First the boundary between the iris and sclera is located. Since the top and bottom of an iris are often blocked by eyelids and eyelashes, the gradients are biased in the vertical direction. This means that the outer border of the iris will be decided using its left and right edges which should be free of obstruction. The second circular Hough transform will get the border between the pupil and iris. As in Daugman’s system, this process is performed inside the previously identified iris region.

Areas of the iris that are corrupted by eyelashes, eyelids or specular reflections are marked as noise. The upper and lower eyelids are detected using linear Hough transforms. In both cases a horizontal line is drawn at the point closest to the pupil where the detected eyelid line intersects the outer edge of the iris. Everything on the outer sides of these lines is marked as noise. If the Hough transforms are accurate this process should mark most of the iris that is occluded by the eyelids as noise. However, it also misclassifies a large portion of the unobstructed iris as noise. Eyelashes and specular reflections are detected using simple thresholding as these values tend to be very dark and bright respectively. This involves manual experimentation to pick the optimal threshold values for a specific dataset. The segmentation process can be seen in Figure 2.1. Although the iris boundaries are perfectly located a large portion of the unobstructed iris is classified as noise while sections of the eyelids and eyelashes are not.

![Figure 2.1: CASIA-Iris-Interval image ‘S1001L01’ Segmented with Masek’s System](image-url)
Masek heavily based his approach for normalisation on Daugman’s [5]. Lines are outputted from the centre of the pupil to the outer edge of the iris at different angles. Several equally spaced points are selected along these lines within the iris region. The number of outputted lines is controlled by the angular resolution and the number of selected points along the lines is controlled by the radial resolution. The process is shown in Figure 2.2.

![Figure 2.2: Masek’s Method of Normalising an Iris](image)

Masek employs one-dimensional Log-Gabor filters to extract the discriminating features of an iris. They have the same desirable properties as a Gabor filter. However, they also guarantee the signal will not have a DC component. The corrupted parts of the iris are set to the average intensity of the uncorrupted iris pixels, so they do not affect the output. A Fast Fourier Transform is applied to each row. Recall the rows correspond to rings in the iris region. The resulting complex values are then convolved with the Log-Gabor filters. The output is phase quantised [5] in the same manner as proposed by Daugman [6]. The produced templates will be in the form of binary matrices. The columns will have length equal to the radial resolution and the rows will have length equal to double the angular resolution since the Log-Gabor filters produce a two-bit output for each pixel. A corresponding mask of the same dimensions will also be produced for each template.

The similarity of two templates is calculated in the same way as in Daugman’s system [6]. Masek uses Equation 2 to ensure only the bits that are not flagged as corrupted in either template’s mask are considered when calculating the hamming distance [5]. The templates to compare are denoted by $X$ and $Y$ while $Xm$ and $Ym$ denote their noise masks. $N$ is the size of the templates and masks. The equation finds the percentage of uncorrupted bits that match.
\[
HD = \frac{\sum_{j=1}^{N} (X_j \oplus Y_j) \land \neg(X_m \lor Y_m)}{N - \sum_{k=1}^{N} X_m \lor Y_m}
\]

2.1.3 NIST VASIR

Concerned over the lack of open source software for iris recognition, the National Institute of Standards and Technology (NIST) developed their own system called Video-based Automatic System for Iris Recognition (VASIR) [7]. They hoped their implementation could be used by other researchers to benchmark future iris authentication systems. The code is based off a C++ port of Masek’s work that was developed to allow researchers to benchmark their entries in the Iris Challenge Evaluation (ICE) in 2005 [14]. The code has been updated to include several improvements. The system is primarily designed so that it can identify users in unideal situations. For example, using video footage of a distant iris. However, it also works for close up stationary images of irises.

The most notable changes from Masek’s original implementation occur in the segmentation stage. The new process was designed to be more resilient to noise. Unlike in Daugman’s and Masek’s approaches, the inner boundary of the iris is located before the outer border. The radius of the pupil is then used as the minimum radius of the iris. This decreases the number of dataset specific parameters that must be selected. The pupil can be located by searching the darkest areas of the image since pupils usually have very low pixel intensity. A threshold value is automatically detected for the image. It is used to convert the image to a binary form. Noise is removed using a Gaussian filter and a morphological opening and closing. The biggest circle in the binary image is detected using contour processing and it is taken to be the pupil. If the pupil cannot be found the threshold can be readjusted before starting the process again. The iris’ outer boundary is detected using a circular Hough transform. The centre point of the pupil is used as a starting point for the search to help speed it up and improve its accuracy.

VASIR includes an improved version of Masek’s eyelid detection process. The upper eyelid is split in three. A linear Hough transform is applied to each third to find a straight line that represents the eyelid edge. A point is sampled from the edge in each third and the border of the eyelid is interpolated using them. This procedure is then repeated for the lower eyelid. This proves to be significantly more effective than Masek’s method since it can more accurately fit the shape of the curved eyelid.

The normalisation process is identical to Masek’s. The feature encoding stage is also similar. However, in the NIST system the real and imaginary components are exam-
ined before being quantised. If the amplitude of a convolved pixels is too large or too small, it is flagged as noise. The reason for this is that small values likely correspond to noise while large values are probably outliers.

The similarity between templates is again calculated by finding the hamming distance between them. However, in this case the matrix templates can be shifted vertically as well as rotated horizontally. This corresponds to shifting the radial lines up and down. It can be useful for noisy images where the detected iris boundaries are slightly off. However, it can also slow down the system as extra comparisons need to be made. Additionally, it can increase the likelihood of false positives.

2.1.4 Ordinal Measures

Sun and Tan propose an alternative approach to the feature encoding stage in the form of ordinal measures [8]. Ordinal measures focus on qualitative information rather than quantitative values. Instead of getting precise values for iris regions they order them relative to each other. This creates templates that are independent of lighting conditions since lighting should not affect the relative ordering of nearby regions. This makes them more robust. Multilobe differential filters are employed to extract the ordinal features. The resulting system produces very compact and accurate binary vector templates. A major advantage of this approach is that it removes the reliance on masks since the bits in the template are taken over a region of pixels instead of an individual pixel which could be corrupted. As in the other systems the similarity of templates is measured by calculating the hamming distance between them and rotational inconsistencies are accounted for by performing rotations.

Unfortunately, there is currently no open source implementation of the system.

2.1.5 Rotation Invariant Templates

Damer et al. propose a novel approach for handling rotational inconsistencies in the form of rotation invariant codes [9]. The ordinal measures approach proposed by Sun and Tan [8] is used to generated binary templates. It was chosen since it is highly robust to intraclass variations and does not have the same reliance on a noise mask as the other methods discussed. The binary vector can be thought of as a ring. The distance between points \(i\) and \(j\) in the template can then be calculated using Equation 3 where \(n\) is the size of the template.

\[
d(i, j) = \min (|i - j|, n - |i - j|)
\]

Each element in the rotation invariant form corresponds to a possible distance between
points. Let $v$ and $u$ denote the original and updated templates respectively. The value for element $k$ in the updated template is calculated as shown in Equation 4, where $\delta$ is the kronecker delta function which take a value of 1 when its argument are equal and otherwise takes a value of 0. Element $k$ represents the number of matching pairs of bits which are a distance of $k$ apart.

$$u_k = \sum_{i<j} \delta(d(i,j), k) * \delta(v_i, v_j)$$

(4)

Unfortunately, this method cannot be employed in systems that utilise a noise mask. This is because the noise masks will affect the conversion in different ways even when the templates come from the same iris. This severely restricts the use of this technique.

### 2.2 Locality Sensitive Hashing

According to Al-Kuwari et al. hash function should generally seek the avalanche effect [15]. This means flipping a single bit in the input should have a 50% chance of flipping each bit in the output. This makes it harder for an attacker to deduce the input to a hashing function given its output. It also means there is no correlation between changes to the input and changes to the output. This behaviour is exhibited in hash functions such as SHA-2 and SHA-3. However, when hashing the iris templates the avalanche effect is considered undesirable. This is because templates generated using different images of the same user’s iris are probably going to differ slightly. If they were hashed using a traditional hashing function, there would be no way to use the hashes to measure the similarity of the original templates.

Locality-sensitive hashing (LSH) is a type of hashing that generates similar outputs for similar inputs. If the iris templates are hashed using a LSH algorithm it will be possible to compare the resulting hashes to estimate the similarity of the templates that generated them. LSH algorithms will naturally have worse security properties than traditional hash functions since they inherently go against the avalanche effect. Therefore, LSH algorithms are typically only employed in situations where security is not considered important. For example, when estimating the similarity of files or when clustering data. However, there are instances where LSH algorithms can be employed without compromising the overall security of the system. This can occur when an application does not require all the properties that are usually considered essential in a cryptographically secure hash function.

According to Al-Kuwari et al. the three properties traditionally expected in a hash
function are preimage resistance, second preimage resistance and collision resistance [15]. However, it can be shown that none of these properties are essential when hashing the iris templates. Preimage resistance requires the hash function to be one way. This means that it should be infeasible for an attacker to learn any preimage that produces a specific hash. In this case it does not matter if the attacker can find a random preimage of a hash provided they cannot figure out any information about the specific preimage that generated it. Chaudhari et al. define a property known as input hiding which captures this idea [1]. A function is considered to be input hiding if it is computationally hard or information-theoretically impossible for an attacker to figure out any information about the input to the function given its output. Any LSH function that satisfies this property is appropriate for the proposed vaccination passport. Second preimage resistance and collision resistance are not considered important since it does not matter if the attacker can find two inputs that produce the same output. The attacker would be unable to accomplish anything using them unless they had permission to add entries to the blockchain. This permission is reserved for trusted entities. Even if an attacker obtained permission to add entries to the blockchain it is not apparent how they could use a collision to their advantage. The security of the system will be discussed in further detail in Chapter 5.

Several different LSH algorithms have been developed. Each one has its own performance and security guarantees. Two of them are discussed in this section and their suitability for hashing the iris templates is investigated.

### 2.2.1 Random Projection Hashing

Random projection hashing was proposed by Charikar in 2002 [16]. Charikar defines a LSH function as a function which probabilistically makes the outputs generated by two different inputs the same based on a similarity measure of the two inputs. Random projection hashing is a family of LSH functions where the inputs are vectors and the similarity measure is calculated using the angles between them. This can be shown in Equation 5 where \( h \) is the hash function and \( \theta(x, y) \) is the angle between vectors \( x \) and \( y \).

\[
P(h(x) = h(y)) = 1 - \frac{\theta(x, y)}{\pi}
\]

The algorithm is known as SimHash. Its similarity measure is closely related to the cosine similarity of the vectors. Since the angle between vectors can only be calculated when they are the same length, each SimHash function requires all preimage vectors to be the same size. Therefore, the vector size must be provided when creating a SimHash function. The hash function will generate a random parameter vector \( (r) \)
of the specified size. The preimage vectors are hashed by calculating the inner product between them and the parameter vector. The hash will be equal to 1 if the inner product is not negative and otherwise it will be equal to 0.

\[
h_r(v) = \begin{cases} 
1 & \text{if } \langle r, v \rangle \geq 0 \\
0 & \text{else if } \langle r, v \rangle < 0 
\end{cases}
\] (6)

Obviously, a hash that is just a single bit has very limited applications. However, it is trivial to generate a much larger hash by combining different SimHash functions. A series of SimHash functions can be generated each with its own parameter vector. A vector is hashed using each of them and the resulting hash is created by concatenating the results.

It can be shown that SimHash is input hiding. Provided the random parameter vector is properly constructed it should be impossible to deduce any information about the input vector after running the sign operation. Even when multiple SimHash functions are combined together the resulting hash function should still be input hiding so long as the length of the input vectors is sufficiently larger than the number of bits in the hash. This is because each function can be thought of as a linear equation with the input vector’s elements being the unknowns and the hash bit being the output. In this case there will be more unknowns than equations. Therefore, the system of equations will be underdetermined.

### 2.2.2 TLSH

Trend Micro Locality Sensitive Hash (TLSH) is an open source LSH algorithm proposed and implemented by Oliver et al. [17]. It was designed primarily for identifying the similarity of files. It takes a byte string as its input. A sliding window of five bytes is moved across this string. At each step six of the ten possible triplets in the window are selected and hashed. The Pearson hash function is used for this since it is highly efficient and can effectively produce an 8-bit hash [18]. Each possible output of the Pearson hash function maps to a bucket containing a count. For each generated hash the count of its corresponding bucket is incremented. Once the window has traversed the entire string quartile points are calculated using the buckets’ counts. The quartile points are selected so that they partition the counts approximately evenly. The hash is constructed by combining a header and a body. The header is calculated using a checksum, the length of the byte string and the quartile points. The body is created by iterating through the buckets and outputting two bits based on each one’s size relative.
to the quartile points. The values of the bits are chosen as shown in Equation 7.

\[
y = \begin{cases} 
00 & \text{if } x \leq q1 \\
01 & \text{else if } x \leq q2 \\
10 & \text{else if } x \leq q3 \\
11 & \text{else}
\end{cases}
\] (7)

An overview of the TLSH algorithm is shown in Figure 2.3. It can be seen that the six selected triplets all contain the first byte in the window. The other four triplets are not selected since they will all be chosen in subsequent windows. This guarantees that each possible triplet will only be hashed once. Every byte in the input will therefore contribute to a maximum of 12 buckets. The first and last four bytes will contribute to less since they are not included in as many windows. If a byte is changed it can at most affect the value of 24 buckets. However, since the final hash is calculated using the quartile points it is very unlikely to affect all 48 corresponding bits. TLSH works best when the differences between similar inputs are localised to a few adjacent bytes as this will minimise the number of buckets affected.

![Figure 2.3: TLSH Algorithm [1]](image)

TLSH is not considered to be as cryptographically secure as other hashing algorithms.
Since each input to the Pearson hash function is merely 24 bits long there are only 16,777,216 possible inputs. It is therefore feasible for an attacker to calculate the set of triplets which map to each bucket. Using a hash, the attacker would be able to deduce the relative frequency of each set of triplets. They could then estimate the likelihood of the different triplets appearing in the input. For example, if a bucket was represented by ‘00’ the attacker could conclude that the triplets which map to that bucket are uncommon. However, this will only give them a rough estimate of the likelihood of the triplets appearing. Additionally, it will not provide them with any information about the positions of the triplets.

2.3 Blockchains

Blockchains were first proposed by Nakamoto in 2008 as a fundamental component of the decentralised digital currency, Bitcoin [19]. A blockchain consists of a continuously growing list of blocks. Each block contains several recorded transactions and a timestamp. They also contain a hash of the previous block in the list. This means that in order to change a block all subsequent blocks must also be modified to update the chain of stored hashes. Nodes participating in a blockchain system communicate in a peer-to-peer (P2P) network. Transactions are propagated through the network and verified by the participating nodes. They can then be included in a subsequent block. A consensus mechanism is employed to control who adds the next block to the chain. A blockchain is immutable so long as the speed at which blocks are added by honest nodes is greater than the speed at which attackers can create blocks.

Blockchains have received a lot of attention since their inception. There has been considerable research into applying them to other fields besides cryptocurrencies including identification systems, internet voting and medical records management [20]. Several alternative blockchains to Bitcoin have been proposed and implemented including Ethereum [21] and Hyperledger Fabric [22]. This section will examine several different blockchains and assess their suitability for implementing the vaccination passport.

2.3.1 Bitcoin

Traditionally digital payment systems required a central authority to prevent tokens from being spent multiple times. This was a highly contentious issue. Critics argue that the involvement of a third party increases overhead costs, allows buyers to exploit refund mechanisms and endangers people’s privacy. According to Tewari the cryptographic community saw the central authority of digital payment systems as a central point of attack and failure [20]. Nakamoto proposed Bitcoin as a fully peer-to-peer
electronic cash system in 2008 [19] to address these problems. The most significant contribution of Bitcoin was the blockchain. It solved the double spending problem by storing the transactions on a public immutable distributed ledger. The current owners of tokens can be determined by examining the transactions that have been logged on the blockchain. The immutable nature of the blockchain provides a guarantee that the records cannot be changed later.

Bitcoin uses a proof-of-work (PoW) consensus mechanism. This means that the first node to solve a computationally challenging problem adds the next block to the blockchain. This ensures that the blockchain is immutable provided the majority of computational power is controlled by honest nodes. Participating nodes are incentivised to invest computational power into solving the problems as they receive a monetary reward for adding blocks. The more computational power invested by honest nodes the more secure the system will be against attacks. However, the energy usage of the system will also increase. Bitcoin is therefore fundamentally energy inefficient. More energy efficient hardware does not solve this problem since it will only enable both honest and attacking nodes to increase the computational power they invest into the system.

As well as the massive amounts of energy it consumes, Bitcoin has several other noteworthy shortcomings. Bitcoin’s PoW mechanism relies solely on the SHA256 hashing function [19]. Application specific integrated circuits (ASICs) can be built which can compute SHA-256 hashes significantly faster than ordinary general-purpose computers. Therefore, casual participants will struggle to compete against those who have bought specialised equipment. In 2019, Sharkey and Tewari calculated that the top three entities participating in the Bitcoin system controlled over half the computational power [23]. Another problem with Bitcoin is that the blockchain’s built-in scripting language is intentionally not Turning complete [21]. Most notably it does not support loops. This ensures all programs will terminate. However, it also makes it much harder to write applications which run on the blockchain. Finally, Bitcoin’s blockchain was constructed solely to facilitate the cryptocurrency. This makes it difficult to adapt it to other use cases. For these reasons, Bitcoin’s blockchain is unsuitable for implementing the vaccination passport.

2.3.2 Ethereum

In 2013 Buterin proposed Ethereum as an alternative blockchain to Bitcoin [21]. Ethereum seeks to address many of Bitcoin’s issues and provide a more modular and efficient blockchain. Similar to Bitcoin, Ethereum has a built-in cryptocurrency called Ether. The primary purpose of Ethereum is to allow online transactions to be made
using Ether. However, Buterin envisioned many additional uses for the Ethereum blockchain. The main innovation of Ethereum is that it has a built-in Turning-complete programming language. This programming language can be used to create applications which run on the blockchain. Buterin intended these applications to all run on the same instance of the blockchain. Developing applications this way has the advantage of utilising the security and computational power of the main Ethereum blockchain. However, running these applications costs Ether, which can be prohibitively expensive. Fortunately, the modular design of Ethereum makes it possible to create separate Ethereum blockchain networks for specific purposes.

Ethereum has traditionally used a PoW consensus mechanism. However, its modular design makes it feasible to utilise another consensus mechanism. There are currently plans to switch the main Ethereum blockchain network to a proof of stake (PoS) consensus mechanism [24]. In this new PoS consensus mechanism, nodes must pay a deposit of Ether to become a validator. Provided they correctly validate the chain they will be rewarded. However, if they go offline or incorrectly validate the chain, they will lose a portion of their deposit and may be removed as a validator. There are also other consensus mechanisms which have been implemented and can be employed when creating an Ethereum blockchain network.

Szilágyi proposed Clique as a Proof of Authority (PoA) consensus mechanism in 2017 [25]. It was designed primarily for test Ethereum networks. However, it also works well for private and permissioned networks. PoA consensus mechanisms have a list of trusted signers who are allowed to add blocks to the blockchain. Clique is designed in a manner so that it can be easily integrated into the Ethereum protocol. The mutable list of authorised signers is maintained using the blocks’ headers. However, it is not possible to change the structure of the header as this would make it difficult to incorporate it into the Ethereum protocol. Instead, certain fields of the header are repurposed when running Clique. The redundant metadata field is used to store the signature of the entity adding the block. This signature is generated using the block’s hash which contains everything in the block except the signature itself. The signature can be checked against the list of authorised signers.

The genesis block contains the initial list of authorised signers. Changes to this list are voted on by the signers. This process is handled using other redundant header fields. The node that added a block can be inferred from the block’s signature. Therefore, the miner field in the header is obsolete. Similarly, the nonce field is not required in a PoA consensus mechanism. Whenever a signer adds a block it is allowed to vote on one proposal. The miner field specifies the node they wish to vote about. The nonce field indicates whether they want to vote them onto or off the list. A vote succeeds and the list of signers is updated whenever a proposal is supported by a sufficient
percentage of the signers. The exact percent required can be configured when creating the blockchain. However, it must be at least 51%. This ensures the list cannot change without a majority consensus. After a specific number of blocks an epoch transition occurs. This resets all the currently placed votes which keeps the number of ongoing proposals manageable. It also contains the current list of authorised signers. When a node wants to obtain the list of authorised signers, they can retrieve the most recently published one. They then only need to process the subsequent votes which have been cast. This is significantly more efficient than having to process every vote since the start of the blockchain.

The rate at which signers are allowed to add blocks is limited. This minimises the damage a malicious signer can do before the other signers vote to remove them. Between a single signer adding two blocks it is required that at least half of the other signers add one. In this way the blockchain is secure provided the majority of the authorised signers are benevolent. A fork occurs when two signers attempt to add a block at the same time. The other nodes in the system will be more likely to accept the chain with a higher difficulty level as the canonical chain. The difficulty level of a chain is higher when signers follow a round robin schedule when adding blocks. This incentivise the signers to wait for their turn before adding blocks to increase the difficulty of the chain and make it more likely to be accepted as the canonical chain.

2.3.3 Hyperledger

Hyperledger is a family of open source blockchain frameworks and related tools [26]. It is run by the Linux Foundation. As of August 2018, 230 organisations were registered as members of the Hyperledger group. The idea is that companies can pool together their research and work around creating blockchains for enterprise purposes. All the work that is contributed is made open source so that everyone can benefit from it. Unlike Ethereum and Bitcoin, Hyperledger does not have a built-in cryptocurrency. However, it is still possible for developers to build a cryptocurrency on top of a Hyperledger blockchain. Hyperledger follows a very modular design so that developers can adapt it to their needs.

There are many different blockchain frameworks in the Hyperledger umbrella project including Burrow, Fabric, Indy, Iroha and Sawtooth [26]. Their modular designs allow components to be reused across them.

Hyperledger Fabric is a blockchain framework which focuses on allowing some transactions to be partially private [22]. Channels containing their own ledger of transactions can be setup between entities. Any transactions performed on these channels can only be seen by its members. Alternatively, entities can opt to only include the hashes
of transactions on the blockchain. The transactions can then be shared with other entities on a need-to-know basis and validated against their publicly stored hashes. Hyperledger Fabric follows a very modular approach which allows different consensus mechanisms to be utilised. Applications which run on the blockchain can be written in general-purpose programming languages [26].

2.4 Related Work

The COVID-19 pandemic has triggered a surge of interest in vaccination passports. This interest has mainly revolved around building smartphone applications which would enable users to easily access their records. For example, CommonPass [3] and the Vaccination Credential Initiative [4] are smartphone based vaccination passports which have a large amount of backing behind them. However, there has also been considerable research into creating a blockchain based vaccination passport [27]. This section will investigate some alternative approaches for building a blockchain based vaccination passport and compare them to the system described in this paper.

2.4.1 COVID-19 Antibody Test/Vaccination Certification

Eisenstadt et al. propose a blockchain based platform for validating test results and vaccination records [28]. The system utilises an Ethereum blockchain and a PoA consensus mechanism. The complexity of the blockchain is abstracted away from the users. Instead, all interaction with the system is performed through a mobile application.

A particularly interesting feature of this system is that users store their own records either on their phone or on a cloud server. However, it is worth noting that most records will end up being stored on a third party’s cloud server either for practicality or redundancy. Therefore, the user will not have complete control over them. The blockchain itself only stores hashes of the records. This allows a user to keep their information private and only share it with those they want to. They also have the option to completely delete their records. This is not possible in the system described in this dissertation since the transaction that added a record will be permanently stored.

Trusted medical entities are in charge of issuing the test results and vaccination records. When end users sign up to the application, they must provide a picture of an identification document. This is stored locally alongside their other information. However, its hash is sent to the blockchain to permanently link the document with the user’s account. The digital and physical versions of this document are shown alongside each other to a medical entity when the user is being tested or vaccinated. This proves their
identity and verifies the certificate is being issued to the right account on the system. The medical entity signs a certificate containing the test result or vaccination record. The user’s identifier is incorporated into this certificate to prevent it being used by others. The user will also sign this certificate and store it locally. A hash of the certificate is added to the blockchain. The certificate can be presented to other people who are using the application. The public keys and hashes can be retrieved from the blockchain to verify it.

2.4.2 Immunity Certificate

Bansal et al. propose an immunity certificate [29]. An immunity certificate is slightly different to a vaccination passport. It seeks to prove that users are immune to COVID-19 by demonstrating they have already been infected and have tested positive for COVID-19 antibodies. Their practicality is based on the assumption that those who have been previously infected are immune to COVID-19. However, according to WHO this might not always be the case especially with regards new COVID-19 variants [30]. Additionally, there is another major issue associated with them. A vaccination passport encourages users to get vaccinated. This is considered desirable as it will increase the uptake of vaccinations and help to curb the spread of the virus. Conversely an immunity certificate incentives people to get infected. This would greatly increase the spread of the disease. Nevertheless, the design of the immunity certificate could be easily adapted to a vaccination passport as all the requirements are the same.

The proposal is lacking in detailed technical information and only offers a very high-level overview of the system. A country’s government is in charge of running the blockchain. Similar to the system described in this dissertation biometric data is used to authenticate the users. In this case the biometric data acts as a private key. Ordinary people, testing facilities and hospitals participate in the system. When somebody is tested for COVID-19 antibodies a smart contract allocates them an anonymous temporary key pair. The public key is sent to the medical entity performing the test and used to encrypt the results. The smart contract checks the results. If they verify the user has COVID-19 antibodies it issues them an immunity certificate with a specified expiry date. Details are not supplied regarding how the user is anonymously linked with the temporary key pair.

As in this dissertation’s system, the blockchain is publicly readable. Anyone can see the immunity certificates. A user just needs to prove ownership of theirs by demonstrating they possess the associated biometric. Unfortunately, no information is provided about how the biometric system would be implemented.

Bansal et al. suggest incorporating contact tracing into the system [29]. They argue
this would fix the problem where users are incentivised to get infected. However, no justification is provided for this viewpoint.
3 Design

This chapter offers a high-level overview of the proposed blockchain based vaccination passport’s design. The design is based on the one described by Chaudhari et al. [1]. The system’s three major components and their interactions are outlined.

3.1 System Overview

The system consists of three major components. It follows a modular design so that each component can be changed with minimal interference to the overall system.

- An iris template extraction program is required to generate a binary iris template from an image of a user’s iris. This template will be used to help identify the user’s records. Any iris template extraction algorithm that produces a binary vector can be used here. Alternatively, a different biometric could be used instead of irises. However, it would need to be as stable as the iris and offer comparable security and accuracy. It would also need to be possible to convert the biometric into a binary vector.

- A locality sensitive hashing algorithm is employed to anonymise the iris templates. It will also ensure that similar templates have similar hashes. This allows a user to search for their stored hashed iris scan using another hashed iris scan. The stored hashed templates are used to help construct users’ unique identifiers. Any locality sensitive hashing algorithm can be used here provided that it generates similar outputs for similar inputs. For security reasons it is also required that no part of the input can be deduced from the output.

- A blockchain is utilised to store and access the vaccination records. The blockchain must be configured so that only verified entities can add vaccination records. This ensures that all the stored vaccination records are authentic provided that the trusted entities behave in an honest manner. Corrupt institutions can fabricate records regardless of how the vaccination passport is implemented. For this reason, ensuring the trusted entities behave in an honest manner is consid-
ered outside the scope of this project. However, it is noted that the high level of transparency provided by a blockchain would make it easier to detect and trace misbehaving entities. Anyone should be able to query the stored records. This enables a user to share their vaccination records with whomever they wish. Any blockchain which can provide everyone read access while restricting write access can be employed. Ideally, for security reasons the blockchain should allow some transactions to be made anonymously.

A high level design diagram of the system is shown in Figure 3.1. The user will provide some personal information and an image of their iris. The iris image will be converted into a binary template using the template extraction program. It will then be hashed using the locality sensitive hashing algorithm. At this point the hashed iris scan and personal information can be used by a local instance of the blockchain to either enrol the user onto the system or locate their existing records. The blockchain is replicated on several nodes which communicate across the internet.

![Figure 3.1: High-Level Overview of System](image)

### 3.2 Application Workflow

Figure 3.2 shows the workflow of the system. When a user wishes to interact with their records, they must provide an image of their iris and some personal information. This personal information could consist of the user’s date of birth (`DoB`) and their `gender`. `DoB` would be in the form dd/mm/yyyy and `gender` would be male, female or other. The iris scan will be converted to a binary feature vector (`fv`) using the iris template extraction program. It will then be hashed with the locality sensitive hashing algorithm (`H1`). A unique identifier can be created for a user by combining their `DoB`,
gender and hashed iris scan as shown in Equation 1. $H_2$ can be any traditional hashing algorithms such as SHA-2 or SHA-3.

$$ID = H_2(DoB \parallel Gender \parallel H_1(fv))$$ (1)

The system will search through the anonymous hashes that are stored on the blockchain for ones which are similar to the inputted hashed feature vector. The similarity of two hashes is calculated by finding the hamming distance between them. The system may find a set of hashes that are within a specified threshold. The system will then use Equation 1 to combine each of these hashes one at a time with the previously entered DoB and gender to create $\tilde{ID}$. It checks each $\tilde{ID}$ to see if it matches an ID stored on the blockchain.

If a match is found, then the ID must belong to the user as it has been construed using their personal information and biometric data that presumably belongs to them. In this way the system utilises a two-factor authentication mechanism where a user’s records can only be located using the biometric data they possess and the personal information they know. The ID can then be used to locate the user’s records. Additionally, an authorised entity may use it to update the user’s records.

If no match is found, then it must be the user’s first time presenting themselves to the system. At this point an authorised entity can enrol them onto the system. The authorised entity starts off by anonymously storing the inputted hashed iris scan on the blockchain. The user’s unique identifier created by Equation 1 is then stored as a separate transaction along with their vaccination details. The two transactions are broadcast on the peer-to-peer blockchain network so they can be verified by the other nodes in the system and added to a block on the blockchain. The transactions are uploaded at random intervals to ensure they are stored on separate blocks. This makes it harder for attackers to deduce the relationship between them.
Figure 3.2: Flow Chart of System Workflow [1]
4 Implementation

This chapter describes how the prototype of the vaccination passport was created. It will discuss the implementation of the three different components of the system outlined in the previous chapter.

4.1 Iris Template Extraction

Libor Masek’s code [13] was used for the iris template extractor. It was chosen primarily because of how well documented it is. Additionally, working in Matlab allows quick testing of experimental changes due to the highly functional debugger. The NIST C++ code might be more suitable for a deployed application due to its reported improvements [7]. However, it has not been well maintained since its creation. In the meantime, there have been major changes to the libraries it depends on and to C++ itself. This has caused significant compatibility issues. Additionally, the installation instructions are very complicated and out of date. Preliminary attempts to use it produced very poor results. Therefore, Masek’s code was deemed more appropriate for the initial prototype. However, as mentioned the system is highly modular so the NIST system or another program could be later used instead. The original version of Masek’s code was not entirely suitable for this application. This section will outline how it was changed to integrate it into the system and improve its performance.

4.1.1 Converting the Template to a Vector and Accounting for Rotational Inconsistencies

As previously discussed in Chapter 2 Masek’s program outputs an iris template in the form of a binary matrix. The rows of this matrix correspond to the relative distances between the inner and outer iris boundaries and the columns correspond to the angles relative to the centre of the pupil. The LSH algorithm requires this template to be in the form of a vector. There are two basic choices for converting the matrix into a vector. Either the rows or columns can be concatenated. Chaudhari et al. argue that column concatenation is preferable as it allows for easier rotations [1]. In Masek’s original
work each row would be rotated separately to account for rotational inconsistencies [5]. If column concatenation is used the same effect can be achieved by rotating the entire vector. Recall that each pixel sampled from the iris region is represented by two bits in the template. Therefore, the length of the rows is double the angular resolution. Thus, shifting a flattened vector by the length of two columns is equivalent to shifting each row by two bits or one pixel.

The template conversion and rotation process is shown in Figure 4.1. Each column is concatenated to produce the vector. The vector is then rotated right by double the length of a column which in this case is 6 elements. When the vector is converted back into a matrix each row has been shifted by exactly one pixel to the right.

Since the templates are hashed before being compared, Daugman’s method of handling rotational inconsistencies [6] cannot be directly leveraged. However, the same basic idea can be followed. An inputted feature vector is rotated a specified number of times left and right. At each of these rotations it is hashed. An inputted template will therefore have a set of hashes which will be compared to all the hashes stored on the blockchain. The iris scan’s similarity to each of the stored hashes will be measured by taking the minimum hamming distance between its hashes and the stored hash. This should correspond to the correct alignment between them and will thus account for any rotational inconsistencies in the templates. However, it is not an ideal solution as it greatly increases the number of comparisons. If four shifts are performed left and right, then the total number of comparisons is nine times larger than when doing no rotations. This will have a significant performance impact on the system. Additionally, the more hashes that are created for a template the more likely it will be that one of them is similar to a random hash. This means the likelihood of false positives increases along with the number of rotations considered. Therefore, a balance must be struck when determining how many rotations should be performed.

The technique of rotation invariant codes proposed by Damer et al. [9] could offer
an alternative solution to this problem. However, this would require the employment of a different segmentation system that does not have the same reliance on noise masks.

### 4.1.2 Masks

Masks proved to be a major obstacle for the system. Recall that when two templates are compared in Daugman’s [6] and Masek’s [5] systems their masks are combined. This shared mask is then used to ensure the comparison only involves bits which are not corrupted in either template. However, in this case an inputted template and mask must be compared against hashed templates. Therefore, it is not possible to combine both templates with a shared mask before comparing them.

The hashed templates could keep their masks. The inputted template could be combined with each stored hash’s mask before being hashed and compared to it. This would be very computationally intensive as a separate hash would have to be generated for each comparison. Additionally, storing the masks would greatly increase the memory usage of the system since they are much larger than the hashes. Furthermore, following this approach only half solves the problem as it is impossible to retroactively combine the hashed template with the mask of the inputted template.

Chaudhari et al. propose two other possible solutions to this problem [1]. Each template can be combined solely with its own mask. This will remove the corrupted bits in each template. The hashes will then be generated using only the uncorrupted bits. The accuracy of this method is dependent on the assumption that different templates generated for the same user will have similar masks. The main contributor to the mask will be the areas of the iris blocked by the eyelids. It seems plausible that these areas would be relatively consistent for a user. The other parts of the mask that correspond to areas of the iris obstructed by reflections and eyelashes are more likely to be random. However, these tend to be comparatively small. Nevertheless, it seems likely that this method will incur some kind of accuracy penalty.

Alternatively, a global mask can be created. Chaudhari et al. suggest it could be constructed by combing all the masks generated using a training database [1]. This makes some sense as it will remove portions of the iris which are often blocked by the upper and lower eyelids. However, it will also remove large portions of the iris which are only corrupted in a single template. If a large training dataset is used it is liable to flag the entire template as corrupt since all parts of the iris will probably be obstructed in at least one template. Instead, the global mask could be created by only flagging bits which are regularly marked as corrupted or by trying to predict the parts liable to be occluded by eyelids. Even still this causes issues as some templates will have uncor-
ruptured parts of the iris needlessly discarded while others will have corrupted areas influence the hash generation.

The first of these proposals was judged to be more appropriate as testing and analysis suggested that the areas of the iris flagged as corrupted were relatively consistent for each user. Results are reported in Chapter 5 and the incurred accuracy penalty is examined.

4.1.3 Parameter Tuning

Masek’s code contains many parameters which can be tweaked to improve performance on a specific dataset. The effectiveness of Masek’s code was found to be highly dependent on tuning these parameters for each dataset. This meant that the program had to be run differently for each dataset. This is not considered desirable as it means imaging conditions must be kept consistent across different uses of the system to obtain optimal results. Consistency could feasibly be maintained across the various trusted entities. However, it would be impossible to maintain consistency across ordinary users of the application. Therefore, this might pose an issue to users who are independently trying to access their records.

There were many parameters which had to be selected separately for each dataset. A minimum and maximum radius size for the iris and pupil had to be chosen. This is used to narrow down the search for the iris boundaries. This greatly speeds up the process. It also decreases the likelihood of the system mistakenly detecting other circular objects instead of the iris edges. The values are chosen by manually inspecting the iris images and attempting to select the narrowest range which encompasses all of them. The angular and radial resolutions must also be decided. They determine the number of points sampled from the iris and by extension the size of the input vectors. Their optimal values depend on the resolution of the iris. Therefore, they can be partly inferred by examining the images. However, precise values must be chosen by testing various combinations. Thresholding values must be selected to help determine the edges in the image. Large values are liable to miss the boundary between the iris and sclera. Conversely small values cause significant performance penalties and increase the likelihood of the system mistakenly finding other circular objects. There are plenty of other values that could be changed in the code to try to improve the results. However, they were judged to be less critical for the system and therefore, were not fully explored. Most notably the parameters of the Log-Gabor filters were left unchanged since the default values Masek selected performed well on all the datasets.
4.1.4 Code Improvements

Masek’s program has not been maintained since it was first released in 2002. Fortunately, it still achieves good results on the recent versions of Matlab. Nevertheless, it contains some minor mistakes which have not been corrected. Furthermore, there is plenty of scope for optimising the code. Therefore, several improvements were made to Masek’s code to improve its efficiency and reliability.

Some minor mistakes were detected and fixed in Masek’s program. For instance, a problem was found in the code which normalises the iris into a fixed dimension rectangle. Recall that each column in this rectangle corresponds to an angle relative to the centre of the pupil. One of these angles was being sampled twice. This meant that two of the columns in the normalised form were identical as they corresponded to the same region of the iris. This happened because angles were taken in the inclusive range of 0 to $2\pi$. However, since circle angles follow modular arithmetic the pixels at an angle of 0 and $2\pi$ are the same. This will cause an issue when trying to fix rotational inconsistencies as two scans of the same iris could have different regions sampled twice. This would make it impossible to perfectly line up the two templates. This problem was easily fixed by making it a half-open interval. Another mistake was discovered where the corrupted pixels are set to the average intensity of the uncorrupted pixels. This takes place immediately before the feature encoding stage so that the corrupted pixels do not interfere with the Log-Gabor filters. Masek’s original code was incorrectly calculating the average value of the uncorrupted region. Again, this problem was easily remedied. Although, these mistakes were minor, correcting them led to an improvement in the accuracy of the iris recognition system.

Matlab is an interpreted language which means that it compiles code while running it. It uses just-in-time (JIT) compilation to produce highly optimised machine code [31]. However, the process of simultaneously compiling and executing the code means that it typically runs substantially slower than compiled languages such as C. Matlab is especially optimised for operations involving matrices and vectors [32]. Therefore, code can be made much more efficient by vectorising it. Masek’s program was not optimised to take advantage of this. Thus, the code was rapidly sped up by refactoring it.

The weakest part of Masek’s code is the segmentation phase. The normalisation and feature encoding stages were examined and found to be very effective. This is supported by the fact that the NIST system reused Masek’s normalisation process and only made minor changes to the feature encoding stage [7]. The most significant deviations in the NIST protocol occur in the segmentation phase. Therefore, efforts in this research were also concentrated on improving this stage. Attempts were made to
implement Daugman’s differential-integro process [6] in Matlab. This code was substituted in place of Masek’s segmentation functions and used alongside the rest of his system. However, it was not any more accurate at segmenting the iris than Masek’s original system. It did have the advantage of requiring fewer dataset specific parameters. Nevertheless, it was decided to continue using Masek’s segmentation process as it had more potential for improvement.

Efforts were then concentrated on adjusting Masek’s code in an attempt to improve the segmentation process. Since the code was running much faster it was possible to conduct a more thorough search for the iris’s inner and outer boundaries. This increased how often the iris could be found. The eyelid detection algorithm was an especially weak part of Masek’s system as mentioned in Chapter 2. Recall in NIST’s VASIR system the eyelid detection process was greatly improved by splitting the upper and lower eyelid in three [7]. The shape of the eyelids could then be interpolated by sampling an edge point from each section. This method was ported into Matlab and used in place of Masek’s eyelid detection system. It does not introduce any noticeable slowdown since it is essentially just performing six quick linear Hough transforms. The eyelash and reflection detection process were also updated to make them more accurate and less reliant on dataset specific parameters. A comparison between the original and updated segmentation systems is shown in Figure 4.2. It is clear that the updated method is significantly more accurate than Masek’s.

4.2 Locality Sensitive Hashing Algorithm

The primary candidates for the LSH algorithm are TLSH and SimHash. TLSH has better performance and could potentially generate a rotation invariant hash. However,
SimHash is considered preferable primarily because it offers better security guarantees. Additionally, there is another major issue with using TLSH in this domain. As previously mentioned TLSH is most effective at mapping similar inputs to similar outputs when the differences between them are localised to small areas. Masek’s iris extraction process is more susceptible to frequent small differences between templates rather than large, concentrated ones. This means that TLSH cannot effectively generate similar hashes for different iris templates that belong to the same user.

The prototype employs a variant of SimHash known as S3Hash. However, the modular design means that other LSH functions could be used instead provided they offer sufficient security and accuracy guarantees.

### 4.2.1 S3Hash

S3Hash is a variant of SimHash that was proposed by Chaudhari et al. [1]. S3Hash follows the same procedure as SimHash. A series of hash functions which output a single bit are combined to create a hash function with a much larger output. Since each of these sub functions generate a single bit, a hash of length $m$ is created using an equal number of sub functions. As in SimHash each of the sub functions has a random parameter vector $(r)$. The primary difference is that the random vector is sampled from the finite field $\{-1, 0, 1\}$. Additionally, all the input vectors are binary. For simplicity, the sub functions can all be combined into a single function which has $m$ random parameter vectors. The formula is shown in Equation 1.

$$S3Hash_R(v) = (\text{sgn}(\langle v, r_1 \rangle), \text{sgn}(\langle v, r_2 \rangle), ..., \text{sgn}(\langle v, r_m \rangle))$$

Where sgn is a function which returns 1 if its argument is not negative and 0 if it is. It is shown in Equation 2.

$$\text{sgn}(x) = \begin{cases} 
1 & x \geq 0 \\
0 & x < 0 
\end{cases}$$

A balance must be struck when choosing the length of the hashes. Larger hashes will have greater entropy. This decreases the probability of templates from different users matching. However, the time required to hash a template is directly proportional to the size of the hash since an inner product operation is required to generate each bit. Furthermore, larger hashes also require more space. A length of 512 bits was chosen as this was judged to provide sufficient entropy without overly damaging performance.
The program was written in Python. Python was deemed a suitable choice since it is a high-level language which allows rapid prototyping. It also supports many useful libraries which simplify the task of coding. Furthermore, Python is very portable which is important as the vaccination passport needs to be able to run on a large number of different systems. The disadvantage of Python is that it is an interpreted language which makes the code run significantly slower. However, there are libraries which can make it much faster.

The templates are encoded in the form of binary vectors and the hashes in the form of integers. Although Python supports both lists and high precision integers, the default implementations are highly inefficient. Libraries are utilised to make the code substantially more performant. The NumPy library [33] is used to encode the templates and random parameter vectors as arrays. NumPy contains a built-in function to calculate the inner product between two arrays. This function is precompiled, so it runs significantly faster than code written in native Python. The hash is represented using the gmpy2 library. The gmpy2 library is written in C and is used for arbitrary precision arithmetic [34]. One of the major advantages of gmpy2 is that it includes a hamming distance function that is highly performant. This allows the similarity of hashes to be compared very quickly.

4.2.2 Selecting Parameters

Several observations were made when examining the hashes produced by the S3Hash function. Ideally each bit in the produced hashes should have an equal probability of being a 1 or a 0. This will maximise the inter user variability as it will limit the correlation between hashes except in cases where the input vectors are similar. This in turn will make it easier for the system to identify which hashes come from similar templates. However, it was found that some parameter vectors were much more likely to hash input vectors to a particular bit. No issues were detected in the random generation of the parameter vectors. They were examined and found to have a roughly equal number of -1, 0 and +1 elements. Instead, the problem lay in the distribution of the templates themselves. It was found that the upper and lower regions of the iris were liable to be occluded by eyelids and therefore flagged as corrupted. This meant they would be set to 0 in most templates and by extension have no impact on the generated hashes. Any parameter vector that had a disproportionate number of -1 or +1 elements in these areas would be more likely to hash a template to a 1 or a 0 respectively. Additionally, recall that each pixel sampled from the iris is represented using two bits. This means a pixel can be represented by four different values. It was found that some of these values were more likely to occur than others. Furthermore, the values assigned to adjacent pixels are not independent. This can also influence how often
a parameter vector hashes templates each way.

The parameter vectors that usually hash templates the same way were ascertained to be having a detrimental effect on the system’s matching accuracy. Therefore, the accuracy of the system can be improved by discarding these parameter vectors and replacing them with ones that are approximately as likely to hash a template to a 1 as a 0. A set of parameters is constructed for each dataset. The selected parameter vectors must meet two criteria. Firstly, they must hash at least 40% of the templates each way. This fixes the issues described above and helps ensure that each parameter vector has a roughly equal chance of hashing a template to a 1 as a 0. Secondly, all the selected parameter vectors should hash between 40% and 60% of the templates the same way. This guarantees there is not a strong positive or negative correlation between the parameter vectors.

4.3 Permissioned Blockchain

Ethereum was deemed to be the best choice for the blockchain. Its Clique PoA consensus mechanism is ideal for the vaccination passport. The trusted entities can act as the signers. Additionally, the Ethereum blockchain framework has been heavily utilised and it is therefore a very tried and tested solution. This is especially true in the field of vaccination passports. According to a study on blockchain based vaccination passports conducted by Abd-alrazaq et al. Ethereum was used in nine out of the ten systems that reported the employed blockchain [35]. There is also a huge amount of resources and tools available to help developers work with Ethereum. Bitcoin was found to be too unmalleable. Hyperledger Fabric offered a promising alternative. Its modular design might enable greater freedom to implement the blockchain as desired. However, it is predominately used by companies and therefore does not have the same available ecosystem of tools and resources as Ethereum. As with the other components the blockchain could be substituted with minimal interference to the overall system.

4.3.1 Creating the Blockchain Network

To run an Ethereum node, a computer requires an Ethereum client. The client implements the Ethereum protocol and thus enables it to be run. There are many different clients available. They are written in a variety of languages and often provide additional features. Go Ethereum (Geth) is one of the official implementations of the Ethereum protocol [36]. It is written in the programming language Go. It is a suitable choice for running the vaccination passport’s blockchain as it is efficient, very widely used and fully supports Clique.
Geth has a tool called Puppeth which helps users construct new Ethereum networks. It is used to create the genesis block of the vaccination passport's blockchain. The genesis block contains the configuration of the blockchain. The consensus mechanism is set to Clique. A list of initial signers is specified. Other trusted entities can be made signers if the majority of the current signers vote to add them. Similarly, nodes can have their status as a signer revoked if most of the signers vote to remove them.

For security reasons blockchains must constantly add new blocks regardless of whether transactions are being made. This is important as it allows transactions to be validated by future hashes in the chain and by extension become immutable. In Clique a time is specified in the genesis block that should elapse between the blocks. The longer the time the longer it will take for transactions to be included in the blockchain and be confirmed by future blocks. Conversely, a shorter time is more likely to cause forks. It is also worth noting that blocks require storage space even when they do not include any transactions. Therefore, a balance must be struck when choosing the time between blocks.

Trusted entities make transactions on the blockchain network when they want to update the stored data. These transactions cost Ether. This means they must pay some Ether whenever they want to change the vaccination records and iris hashes. However, no Ether is required to retrieve the stored information so anyone with access to the blockchain network can look at it. The initial signers are supplied with a starting amount of Ether. Additional Ether must be earned by adding blocks to the chain. This means that only verified signers can earn Ether. However, it is still possible for a non-signer to acquire Ether. This could occur if a signer opted to transfer Ether to a non-signer. This would be regarded as suspicious behaviour and given the circumstances might result in the signer having their status revoked. Alternatively, a former signer might still have some Ether which they earned from adding blocks before they were removed. However, even with Ether a non-signer would not have permission to edit the vaccination records and hashed iris scans. They would only be able to use it to make superfluous transactions. This could add congestion to the network though they would probably expend all their Ether before it became a serious issue. This situation would be unlikely to occur and its consequences are limited. Nevertheless, it would still be preferable if non-signers could be prevented from making any transactions.

Since the trusted entities require Ether to add and update vaccination records and iris hashes, they must keep earning it. This ensures that all entities participating in the system help to add blocks to the blockchain.
4.3.2 Storage of Vaccination Records and Iris Hashes

The vaccination records and iris hashes are stored directly on the blockchain. Blockchains do not provide the most efficient storage due to the large amount of transaction overhead and the lack of built-in compression. Therefore, an alternative solution would be to only store a hash of each record and a pointer to its actual storage location. However, given the vaccination records tend to be quite short and the hashes are restricted to a few hundred bits it is feasible to store them directly on the blockchain. This avoids adding an extra layer of indirection and helps to simplify the system. The memory required by the system will be examined in detail in Chapter 5.

A smart contract is used to store the vaccination records and iris hashes on the blockchain. It is written in Solidity which is a programming language designed specifically for developing smart contracts [37]. According to Abd-alrazaq et al. Solidity is becoming the dominant language for writing smart contracts and was by far the most commonly used language among the examined systems that specified one [35].

The iris hashes are stored in a dynamic array. When a new entry is added it is put at the end of this array. Hashes can be removed by clearing the corresponding entry in the array. This allows the system to support a form of revocation. Without a hash it is not possible to reconstruct the corresponding identifier and locate the associated records. This means that entities running the protocol will be unable to locate a revoked user’s records. However, it is worth noting that a hash cannot be completely deleted since the transaction that added it to the blockchain will be permanently recorded. Therefore, it is always possible to locate a user’s records by working outside the application’s protocol.

The vaccination records are stored in a mapping. A mapping is Solidity’s equivalent of a hash map, but it contains some noticeable differences. The keys are not stored in a mapping, instead they are mapped to a 256-bit address using a SHA3-256 hashing function. These addresses are used to index into a conceptual array. Every value in this array that has not been explicitly set contains a default value. In this way a mapping does not distinguish between whether a key is set or not. In reality most of the array is not initialised as this would waste a lot of memory. However, the underlying implementation of the array is abstracted away from the programmer. The users’ unique identifiers serve as the keys to this mapping and their vaccination records are stored as the values.

Running Solidity code is not very efficient. Therefore, methods are supplied in the contract to retrieve the data so that it can be easily processed using other programming languages.
The contracts are permissioned. They are set so that anyone can read the data stored in them. However, only the trusted entities are allowed to add new records or edit existing ones. A mapping of addresses to Booleans is used to indicate which entities are allowed to write to the contract. When a node attempts to access a function that writes to the blockchain this mapping is checked and it will be denied access if it is not specified as having permission. This check is omitted when accessing functions that only read from the blockchain. This provides an efficient and secure way of restricting write access to the blockchain. Ideally, the accounts with permission would mirror the list of signers. However, for simplicity the list is updated separately in the prototype.

4.3.3 Connecting to the Blockchain Network

The main application is written in Python. It connects the different subcomponents of the system. Iris feature vectors that have been created by the Matlab iris extraction code are manually inputted into the system. The S3Hash code can be directly leveraged to hash these vectors into an integer form. The Web3.py library [38] is used to connect to the blockchain.

The iris hashes are periodically read in from the blockchain. They are stored in a Python list. This list can be rapidly searched for hashes that are similar to an inputted iris hash. Solidity cannot directly return a mapping. Therefore, an array of all the unique identifiers is stored separate to the vaccination records mapping. This array can be retrieved from the smart contract by the Python application and converted to a set. This set can be quickly checked to determine if a unique identifier is present on the blockchain. The vaccination records themselves can be retrieved on demand directly from the Solidity contract. This is viable because this operation will probably only be performed once when a user is interacting with the blockchain. Therefore, the cost of reading all the records into the Python application is deemed to be much larger than just accessing them as needed. Additionally, it avoids the issue where a different entity could update someone’s vaccination records on the blockchain leading to the local copy in the Python application becoming stale.

4.3.4 Running the Blockchain Network

Docker is a tool used to provide isolated environments in which programs can run [39]. In this case it is useful as different nodes can be run in separate containers. They will each have their own IP address and are completely isolated from each other. Therefore, they are only able to communicate with each other across the blockchain network. This allows a simulation of the blockchain network to be built on a single computer.
In order for the different nodes to connect to the same blockchain network several resources must first be shared out-of-band between them. Every Ethereum blockchain network has a numerical identifier which is partially used to identify it. Since the identifier does not solely indicate the network it is possible to reuse one. However, it is considered preferable to pick a unique number. To facilitate this a list of network identifiers is maintained by the community [40]. The blockchain network is also identified by its genesis block. Therefore, both these resources must be shared across the Docker containers. A genesis block is constructed using Puppeth as described above. Each container initialises its blockchain using this same genesis block and connects to the network. One of the Docker containers acts as a bootstrap node. Its IP address is shared with the other nodes to allow them to easily connect to it and by extension each other. Finally, all nodes must use the same hashing parameters. This is essential as it ensures that similar templates map to similar hashes regardless of which node added them. A parameter set is selected as previously outlined and sent to the containers.

Once all nodes have initialised the blockchain they connect to the same blockchain network. They take turns adding blocks to the chain using the Clique protocol described in Chapter 2. One node is assigned the task of uploading the smart contract to the blockchain. The other signers are added to the list of trusted entities so they can write to the contract. The nodes can run instances of the Python application. Using this program, they can read in data from the smart contract and make transactions to edit the information. All the nodes have access to the same data stored on the blockchain.
5 Evaluation

The system was evaluated using three different datasets. Information was collected about the speed of the system, its memory usage and the accuracy of the iris recognition system. The results are presented and discussed in this chapter. An overview of some security and privacy considerations is also provided.

5.1 Testing Data

Three different datasets were used to evaluate the system. To avoid the experiments being biased by a single source, datasets are collected from two different entities. Performing the evaluation on multiple separate datasets prevents the system being overly tuned to a single dataset. This means that the results achieved should be generalisable to other iris datasets.

The CASIA-Iris-Interval images were taken at a close distance under near infrared conditions [41]. Therefore, the images are very high quality. There are a total of 2639 images taken from 249 people. For some users images of the left and right eye are provided. These can be treated as separate classes since someone’s two irises are completely independent of each other [5]. Thus, there are 395 distinct classes in the dataset. The images have a resolution of 320x280.

The CASIA-Iris-Syn dataset contains 10,000 images synthesised from real irises [41]. Intra-class variations were introduced by deforming, blurring and rotating the synthesised irises. This makes the dataset especially challenging as some of the images are very poor quality. The synthesised irises have been found to be comparable to real irises in both appearance and results. There are 1000 distinct classes. The images have a resolution of 640x480. However, the iris only occupies a very small amount of the overall image compared to the other two datasets. This makes the iris detection process harder as the system has to search a larger area.

The IITD iris images were taken close up [42] and are very high quality. However, the lighting conditions under which the pictures were photographed can make it dif-
difficult to distinguish between the iris and the sclera. 2240 images were taken from 224 users. In most cases images are provided for both the left and right eye resulting in 435 distinct classes. The images have a resolution of 320x240.

5.2 Segmentation Accuracy

The accuracy with which the system can detect the iris within an image is very important. If the system incorrectly detects either the inner or outer boundaries of the iris, then the created template will not be properly representative of the iris. Any hashes generated using it will likely be very different to the hashes generated from other images of the same user’s iris. This will greatly affect the matching accuracy of the system. However, in practice misidentified irises are very unlikely to be inputted into the system. This is because it is easy to provide a user with feedback regarding whether their iris has been successfully detected within an inputted image. For example, the system can draw circles around the detected boundaries of the iris in the inputted image. The user can then examine it and decide whether they want to proceed with the image or input a different one.

Most of the incorrectly segmented irises were removed from the datasets to prevent them from affecting the matching results. Any image in which the detected pupil was not entirely contained within the detected iris was automatically removed. The remaining pictures were manually examined and any other incorrect ones were discarded. Exceptions were made in cases where the iris border was only partially incorrect. It was determined that in these cases the accuracy would not be affected too much since most of the generated template would be correct. The percentage of irises that are correctly segmented is reported to assess the performance of the segmentation system. This is not considered as critical as the other aspects of the system. However, maximising the segmentation accuracy will minimise how often users have to input multiple iris scans. Therefore, it is still very important for the system’s usability.

The results for the three different datasets are shown in Table 5.1. As discussed in Chapter 4 separate segmentation parameters needed to be selected for each dataset to try to maximise the results. The accuracy of the segmentation system was especially good in the CASIA-Iris-Interval dataset owing to its high quality images and uniform imaging conditions. The mediocre results for the CASIA-Iris-Syn dataset are likely due to the poor quality images and the deliberately introduced noise. Masek reported that his system successfully segmented 62% of the LEI database and 83% of the CASIA eye image database [5]. Unfortunately, it was not possible to obtain either of these datasets. Therefore, a direct comparison cannot be made. However, it is noted that the updated system achieves better results on the datasets it was tested on with the
exception of the difficult CASIA-Iris-Syn dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total Irises</th>
<th>Irises Removed</th>
<th>Remaining Irises</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASIA-Iris-Interval</td>
<td>2639</td>
<td>196</td>
<td>2443</td>
<td>0.9257</td>
</tr>
<tr>
<td>CASIA-Iris-Syn</td>
<td>10000</td>
<td>2660</td>
<td>7340</td>
<td>0.7340</td>
</tr>
<tr>
<td>IITD Irises</td>
<td>2240</td>
<td>323</td>
<td>1917</td>
<td>0.8558</td>
</tr>
</tbody>
</table>

Table 5.1: Irises Successfully Segmented in the Datasets

5.3 Similar Iris Scan Searching Accuracy

The accuracy of the system when searching for similar iris scans is considered paramount. It is the principle step when recreating a user’s unique identifier. Ideally, the system should always be able to locate a user’s record when presented with their iris scan and personal information. However, it is essential that the system never mistakenly identifies a user as someone else. Several metrics are used to evaluate the matching accuracy.

The decidability index is a metric for evaluating how separable two distributions are [12]. In this case the two distributions represent the distance between hashes of irises from the same users and from different users. It is useful as it provides a single number which summarizes the matching accuracy. It is also independent of the selected acceptance threshold. The higher the decidability index the easier it is to separate the two classes. The decidability index can be calculated as shown in Equation 1.

\[
d' = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}} \tag{1}
\]

The equal error rate (EER) is another metric which can summarise the effectiveness of the system in a single number. Again, it is independent of the selected acceptance threshold. It is the value at which the false acceptance rate (FAR) and the false rejection rate (FRR) are equal [11]. It is desirable to minimise the FAR to decrease the number of unique identifiers which must be generated and checked. Similarly, the FRR should be minimised to maximise how often the system can locate a user’s records. Therefore, the smaller the EER the better.

The matching accuracy of the system is shown for the different datasets using receiver operating characteristic (ROC) curves. This curve measures how the true acceptance rate (TAR) and FRR change as the acceptance threshold is varied. Ideally the TAR should be maximised while the FAR is minimised. Therefore, the closer the curve can get to the top left corner of the graph the better. The bottom left corner of the graph
corresponds to when the threshold is so high that everything is rejected. Conversely
the top right corner of the graph corresponds to when the threshold is so low that
everything is accepted. The ROC curves can be used to help pick threshold values
which offers the best trade-off between maximising the TAR and minimising the FAR.
Traditionally, a line is drawn from the bottom left corner to the top right corner. This
represents how a random classifier with no domain knowledge would perform. Any
classifier which performs worse than this is considered useless.

Optimal parameter values were chosen separately for each dataset. The normalisation
parameters were found by experimenting with various combinations and choosing
the one that had the highest decidability index. For each combination a set of hashing
parameters was constructed as described in Chapter 4. These parameters were used
to hash the templates before comparing them and calculating the decidability index.
Unfortunately, there was an element of randomness in this experiment since some pa-
rameters produce better results than others. However, since all the combinations were
allowed “choose” their own hashing parameters this was minimised. Four rotations
were performed in the initial experiment since Masek found this was a sufficient num-
ber to account for most rotational inconsistencies in his datasets [5]. Having selected
normalisation parameters, the generated templates are used to pick the number of ro-
tations to perform. The same hashing parameters are used since they have previously
produced good results. The decidability index is compared when different numbers
of rotations are performed. The number of rotations that produces the highest decid-
ability index is selected.

5.3.1 CASIA-Iris-Interval Results

Various combinations of radial and angular resolutions were tested for the CASIA-
Iris-Interval dataset as shown in Table 5.2. The optimal combination was found to be
an angular resolution of 200 and a radial resolution of 28. These values produced a
template of 11200 bits. The results of performing different numbers of rotations using
these templates are shown in Figure A1.2. Four rotations was found to achieve the
best results.

<table>
<thead>
<tr>
<th></th>
<th>120</th>
<th>160</th>
<th>200</th>
<th>240</th>
<th>280</th>
<th>320</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>3.00384</td>
<td>3.46766</td>
<td>3.68508</td>
<td>3.65697</td>
<td>3.60521</td>
<td>3.40515</td>
</tr>
<tr>
<td>20</td>
<td>3.01977</td>
<td>3.46636</td>
<td>3.65974</td>
<td>3.70412</td>
<td>3.58479</td>
<td>3.43697</td>
</tr>
<tr>
<td>24</td>
<td>3.01650</td>
<td>3.45520</td>
<td>3.68907</td>
<td>3.68395</td>
<td>3.59778</td>
<td>3.43938</td>
</tr>
<tr>
<td>28</td>
<td>3.04191</td>
<td>3.49453</td>
<td>3.74200</td>
<td>3.74098</td>
<td>3.58534</td>
<td>3.41924</td>
</tr>
<tr>
<td>32</td>
<td>2.99448</td>
<td>3.43398</td>
<td>3.68506</td>
<td>3.68892</td>
<td>3.57489</td>
<td>3.34415</td>
</tr>
</tbody>
</table>

Table 5.2: Angular and Radial Resolution Comparison for CASIA-Iris-Interval
The recognition accuracy of the system using the extracted templates is tested and compared using three different approaches. The first approach employs Daugman’s joint mask method [6]. This is expected to achieve the highest accuracy since only the valid bits in the templates are compared. The second approach compares the templates without combining their masks. Finally, the third approach compares the templates after they are hashed using the S3Hash algorithm described in Chapter 4. All three approaches use the previously picked radial and angular resolutions and are rotated in both directions four times. The ROC curves of the three approaches are shown in Figure 5.2. The approach where the templates masks are combined achieves almost perfect results and manages to virtually reach the top left corner. However, it is apparent that there is an accuracy penalty when not using the shared mask. Additionally, the accuracy of the system decreases when the templates are hashed. Table 5.3 shows the decidability index and EER for the three different approaches.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Decidability Index</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Templates with Shared Mask</td>
<td>5.33896</td>
<td>0.00415</td>
</tr>
<tr>
<td>Templates with Individual Mask</td>
<td>4.57145</td>
<td>0.0151</td>
</tr>
<tr>
<td>Hashed Template</td>
<td>3.742</td>
<td>0.0276</td>
</tr>
</tbody>
</table>

Table 5.3: Comparison of Different Approaches on CASIA-Iris-Interval

However, despite the incurred accuracy penalty the results for the hashed templates still show promise. Recall that the system uses two factor authentication so it is not vital that the FAR be 0. Furthermore, if the user fails to locate their records they can always try again using a different scan. Therefore, the system can also tolerate the
FRR being greater than 0. The FAR and FRR values are shown for different hamming distance acceptance thresholds in Table 5.4. A hamming distance value of 0.4 offers a particularly promising FAR of 0.0014 and FRR of 0.0949.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>FAR</th>
<th>FRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0</td>
<td>0.9904</td>
</tr>
<tr>
<td>0.275</td>
<td>0</td>
<td>0.9687</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0.8846</td>
</tr>
<tr>
<td>0.325</td>
<td>0</td>
<td>0.7064</td>
</tr>
<tr>
<td>0.35</td>
<td>0</td>
<td>0.4567</td>
</tr>
<tr>
<td>0.375</td>
<td>0</td>
<td>0.2237</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0014</td>
<td>0.0949</td>
</tr>
<tr>
<td>0.425</td>
<td>0.0269</td>
<td>0.0283</td>
</tr>
<tr>
<td>0.45</td>
<td>0.2237</td>
<td>0.0064</td>
</tr>
</tbody>
</table>

Table 5.4: FAR and FRR of CASIA-Iris-Interval Hashed Templates

5.3.2 CASIA-Iris-Syn Results

An angular resolution of 120 and a radial resolution of 32 were selected for the CASIA-Iris-Syn dataset. These values produce a compact 7680 bit template. It was decided that 8 rotations would be performed. The graph and table used to select these values are included in the appendix. The large number of rotations required can probably be explained by the fact that the irises were deliberately rotated in the images to introduce intra-class variations. The system is evaluated using the shared mask template, individual mask template and hashed template approaches. The ROC curves of the
system using the three different approaches are shown in Figure 5.3. Again it is clear that there is a penalty incurred from not using the shared mask and from hashing. However, the results for the hashed templates are still viable. The decidability index and EER of the different approaches are shown in Table 5.5.

![ROC Curves for CASIA-Iris-Syn](image)

**Figure 5.3: ROC curves of Different Approaches on CASIA-Iris-Syn**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Decidability Index</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Templates with Shared Mask</td>
<td>3.97859</td>
<td>0.02475</td>
</tr>
<tr>
<td>Templates with Individual Mask</td>
<td>3.60874</td>
<td>0.03735</td>
</tr>
<tr>
<td>Hashed Template</td>
<td>3.10538</td>
<td>0.0474</td>
</tr>
</tbody>
</table>

**Table 5.5: Comparison of Different Approaches on CASIA-Iris-Syn**

The FAR and FRR of the system for various acceptance thresholds are shown in Table 5.6. In this case a threshold of 0.375 or 0.4 could be selected depending on whether it is considered more important to minimise the FAR or the FRR.

**5.3.3 IITD Irises Results**

For the IITD dataset the optimal angular and radial resolutions were found to be 240 and 24 respectively. This results in a 11520 bit template. It was found that the best results were achieved when just one rotation was performed in each direction. The graph and table used to select these parameters are included in the appendix. A comparison of the shared mask templates, single mask templates and hashed templates approaches is shown in Table 5.7. In this case there is no point in showing the ROC curve since all approaches were too close to the top right corner. This is probably be-
Table 5.6: FAR and FRR of CASIA-Iris-Syn Hashed Templates

<table>
<thead>
<tr>
<th>Threshold</th>
<th>FAR</th>
<th>FRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0</td>
<td>0.9141</td>
</tr>
<tr>
<td>0.275</td>
<td>0</td>
<td>0.8224</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0.6728</td>
</tr>
<tr>
<td>0.325</td>
<td>0</td>
<td>0.4925</td>
</tr>
<tr>
<td>0.35</td>
<td>0.0002</td>
<td>0.3135</td>
</tr>
<tr>
<td>0.375</td>
<td>0.0015</td>
<td>0.1686</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0126</td>
<td>0.0827</td>
</tr>
<tr>
<td>0.425</td>
<td>0.1067</td>
<td>0.0296</td>
</tr>
<tr>
<td>0.45</td>
<td>0.4716</td>
<td>0.0076</td>
</tr>
</tbody>
</table>

Table 5.6: FAR and FRR of CASIA-Iris-Syn Hashed Templates

cause of the high level of detail and low intra-class variations in the images. Interestingly the approach with the hashed templates achieved almost as high a decidability index as the approach with templates that were just combined with their own masks and outperformed it in terms of EER. This shows that the performance penalty from hashing the templates is not that significant when high quality images are used.

Table 5.7: Comparison of Different Approaches on IITD Irises

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Decidability Index</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Templates with Shared Mask</td>
<td>7.85861</td>
<td>0.0002</td>
</tr>
<tr>
<td>Templates with Individual Mask</td>
<td>5.67514</td>
<td>0.007</td>
</tr>
<tr>
<td>Hashed Template</td>
<td>4.917</td>
<td>0.0043</td>
</tr>
</tbody>
</table>

Table 5.7: Comparison of Different Approaches on IITD Irises

Table 5.8 shows the FAR and FRR of the system when various acceptance thresholds are used. In this case a threshold value of 0.425 achieves almost perfect results with a FAR of 0.0150 and FRR of 0.0018.

Table 5.8: FAR and FRR of IITD Irises Hashed Templates

<table>
<thead>
<tr>
<th>Threshold</th>
<th>FAR</th>
<th>FRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0</td>
<td>0.9641</td>
</tr>
<tr>
<td>0.275</td>
<td>0</td>
<td>0.8606</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0.6346</td>
</tr>
<tr>
<td>0.325</td>
<td>0</td>
<td>0.3654</td>
</tr>
<tr>
<td>0.35</td>
<td>0</td>
<td>0.1677</td>
</tr>
<tr>
<td>0.375</td>
<td>0.0001</td>
<td>0.0640</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0006</td>
<td>0.0173</td>
</tr>
<tr>
<td>0.425</td>
<td>0.0150</td>
<td>0.0018</td>
</tr>
<tr>
<td>0.45</td>
<td>0.1768</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

Table 5.8: FAR and FRR of IITD Irises Hashed Templates
5.3.4 Risk of Overfitting

It is worth noting that since the same data was used to select the parameters and test the system there is a risk that the results are overly optimistic and that the system will not perform as well with other data. However, this is not considered a major problem. Reasonable values were chosen for all the angular and radial resolutions. These values were observed to produce representative templates for all the irises in the datasets. Additionally, it was noted that a range of values achieved similarly good results. Therefore, it seems unlikely that the chosen values would preclude other images taken under the same conditions from getting good results. Regarding the number of rotations performed it was observed that a slight change in the value would not have a massive impact on the results. Therefore, the selected values should perform well on other iris pictures taken under the same conditions as the datasets. However, a more thorough test could be conducted by trialling the system with larger datasets and using separate subsections for the parameter selection and testing stages.

The risk is slightly more credible in the case of the hashing parameters. To alleviate this worry, the selected templates are evaluated using hashing parameters that were constructed using the other datasets. The results are shown in Table 5.9. Although there was a noticeable decrease in the decidability indexes when employing the hashing parameters that were generated using other datasets it is not massive. Furthermore, the three datasets are very distinct. It would be expected that pictures taken under similar conditions to the datasets would have less variations.

<table>
<thead>
<tr>
<th></th>
<th>CASIA-Iris-Interval Params</th>
<th>CASIA-Iris-Syn Params</th>
<th>IITD Irises Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASIA-Iris-Interval Templates</td>
<td>3.74200</td>
<td>3.65506</td>
<td>3.48675</td>
</tr>
<tr>
<td>CASIA-Iris-Syn Templates</td>
<td>2.72533</td>
<td>3.10538</td>
<td>2.64708</td>
</tr>
<tr>
<td>IITD Irises Templates</td>
<td>4.55106</td>
<td>4.6517</td>
<td>4.917</td>
</tr>
</tbody>
</table>

Table 5.9: Decidability Index of Datasets when using Hashing Parameters Trained on Different Datasets

5.4 Efficiency and Speed of the System

The speed of the system is important for evaluating its viability. Users cannot be expected to wait more than a few seconds to access their vaccination records. Therefore, if the time required to locate a user’s records is too long than the system cannot be used in practice. The faster this process runs the better.
5.4.1 Speed of the Iris Extraction Process

The code optimisations discussed in Chapter 4 not only allowed a more thorough search for the iris boundaries but also substantially decreased the runtime of the iris extraction process. Results were collected on a new upper-middle powered laptop with an Intel i7 processor and 16GB of RAM. Table 5.10 shows the time taken to run the iris extraction process for each dataset. Based on these results it is apparent that even on a less powerful machine the system should only take a few seconds to extract an iris template from an image. This is considered a reasonable wait time for the user. In contrast, testing Masek’s original code with one of the CASIA-Iris-Interval images and the same segmentation parameters took 10.2883 seconds. Therefore, the system is almost an order of magnitude faster than Masek’s.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total Irises</th>
<th>Time Taken (s)</th>
<th>Time per Iris (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASIA-Iris-Interval</td>
<td>2639</td>
<td>3090</td>
<td>1.17</td>
</tr>
<tr>
<td>CASIA-Iris-Syn</td>
<td>10000</td>
<td>10829</td>
<td>1.0829</td>
</tr>
<tr>
<td>IITD Irises</td>
<td>2240</td>
<td>3512</td>
<td>1.5679</td>
</tr>
</tbody>
</table>

Table 5.10: Time Taken to Extract Iris Templates from Datasets

5.4.2 Speed and Efficiency of the Blockchain

The blockchain was setup as described in Chapter 4. Six nodes including the bootstrap node were connected to the blockchain. The system was run using the three different datasets. They were each configured using the previously selected parameters. The speed and efficiency with which the system could locate the user’s records was measured. The evaluation was performed on an Ubuntu Virtual machine running on the aforementioned laptop. It had access to 4 processors and 8GB of RAM. However, these resources were shared across the six docker containers.

For each user in the dataset one iris was stored on the blockchain. Additionally, a unique identifier was constructed for each user using a randomly generated date of birth, gender and 3 digit pin number along with the stored iris. A fake vaccination record was stored on the blockchain with the identifier. The user’s other irises were then inputted into the system to try to retrieve their fake vaccination record. An acceptance threshold of 0.4 was used for all the datasets when deciding which hashes could have come from the same iris. The number of matches that were successfully made is reported alongside the average time taken to search the blockchain.

The results for the three datasets are shown in Table 5.11. Although in some cases the system failed to locate a user’s records it never misidentified a user as someone else. It is apparent that the time taken to locate a user’s records is effected by both the number
of irises stored on the blockchain and the number of rotations performed. The IITD dataset had the lowest lookup time since it only performed one rotation in each direction. The CASIA-Iris-Interval dataset had a slightly longer average search time since it performed more rotations. The CASIA-Iris-Syn dataset had the longest lookup time owing to its large size and the fact it performed the most rotations. Unfortunately, the number of users in each dataset was too small to allow a thorough evaluation. However, results for a larger number of users can be extrapolated from this table. The results for the CASIA-Iris-Interval dataset are used as a basis. It can be estimated that performing four rotations and searching a million hashes would take approximately 124 seconds. However, the time would probably be much less than this since certain operations are independent of the number of irises stored on the blockchain. For example, the time taken to create a set of hashes for a template. Additionally, these tests were run on low powered Docker images. A more powerful machine should run the system significantly faster. There is also the potential to add threading to the system since most of the code can be run in parallel.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Records Stored</th>
<th>Searches Performed</th>
<th>Matching Accuracy</th>
<th>Average Time per Iris (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASIA-Iris-Interval</td>
<td>389</td>
<td>2054</td>
<td>0.8992</td>
<td>0.05047</td>
</tr>
<tr>
<td>CASIA-Iris-Syn</td>
<td>947</td>
<td>6393</td>
<td>0.906</td>
<td>0.07519</td>
</tr>
<tr>
<td>IITD Irises</td>
<td>407</td>
<td>1510</td>
<td>0.9795</td>
<td>0.01615s</td>
</tr>
</tbody>
</table>

Table 5.11: Time Taken to Search Blockchain and Lookup Accuracy for Datasets

5.5 Memory Usage

The memory usage of the system is also crucial in assessing its viability. If the system requires too much memory low power machines will be unable to run it. This will greatly damage the flexibility and portability of the system. Additionally, it will discourage uptake of the system. The memory was examined by studying the size of individual blocks and extrapolating the cost. These results assume a block is added every 15 seconds since this is the default time. This could be adjusted although it would affect the speed at which records could be uploaded. In this case the vaccination records are restricted to 150 bytes.

An empty block uses 606 bytes of space. Given there are 86,400 seconds in a day, there will be 5760 blocks added each day. Therefore, the system will use a minimum of 3,490,560 bytes in one day. This is equivalent to 3.49056 MB. This can be extrapolated to 1.27405 GB a year. This serves as a minimum memory usage for the system in a year.

The cost to add a million users to the blockchain is now examined. A block with 30
iris hash transactions was found to be of size 5742 bytes. This means each iris hash transaction is approximately 191.4 bytes. Therefore, if one million hashes were stored the estimated space used would be 0.1914 GB. A block with 26 record transactions was found to be of size 10102 bytes. This means each record is roughly 388.5385 bytes. Therefore, the estimated space required to store a million records is 0.38854 GB. The total space required to add a million users can thus be estimated at 0.57994 GB.

The maximum space required to add a million users over the course of a year can be calculated to be 1.85399 GB. This seems a reasonable amount. The actual value would probably be lower since this calculated amount assumes there will be 2,102,400 empty blocks as well as the blocks required to store the details of the users. In reality these blocks will overlap which will decrease the memory usage.

5.6 Security and Privacy Considerations

Given the complexity of the vaccination passport there are many ways a malicious actor could attempt to attack it. This section will look at some of the system’s security and privacy issues. The most important countermeasure is to ensure that no sensitive information is included in the vaccination records. They must be completely anonymous and should only include the bare necessities required to assess someone’s vaccination status. In this way, even if an attacker manages to link a user to their records, they should only be able to deduce their vaccination status. However, this is still considered undesirable. Therefore, several mitigation measures are proposed to minimise the risk of an attacker successfully linking a user to their records.

5.6.1 Unique Identifier

The fact that the records are stored on a publicly accessible blockchain, enables attackers to download them and perform offline attacks. This makes brute force attacks against the system a lot more viable. Therefore, the unique identifier must be constructed in a manner that renders any potential brute force attack impractical. However, there is a trade-off involved when deciding what personal information should be used to construct the identifiers. Using more information increases the entropy and therefore decreases the likelihood of brute force attacks succeeding and false positives occurring. However, the more personal information that is used the more details about a user will be leaked if an attack succeeds.

For example, if as described in Chapter 3 the personal information solely consists of a user’s date of birth and gender. It would be feasible for an attacker to combine every possible date of birth and gender with every stored hashed iris scan. For every match
that is found the attacker would know the associated user’s date of birth, gender and vaccination records. If the attacker was targeting a specific user and already knew their date of birth and gender it would be easy to locate their records given sufficient time even without their iris scan.

If some additional personal information was added such as a user’s mother’s maiden name it would make a potential brute force attack much more computationally expensive. However, attackers could guess particularly common second names. In this case if an attack succeeds, they will have learned a user’s date of birth, gender, mother’s maiden name and vaccination records. This would usually be enough information for the attacker to identify the person. Additionally, adding the mother’s maiden name would not prevent targeted attacks where the attacker already knows it.

An alternative countermeasure would be to utilise a pin number. This could be in place of or in addition to the personal information. However, the system would then be reliant on users remembering their pins. This could lead to people losing access to their records. Nevertheless, this may be considered preferable to risking users’ personal information.

### 5.6.2 Iris Anonymity

It is essential that the system does not leak any information about the users’ irises. This will make it much more difficult for an attacker to locate any user’s records. Furthermore, the iris itself can be considered sensitive information. Therefore, it is essential that the LSH function be input hiding. This will ensure that information about the users’ irises cannot be leaked from the stored hashes. Chaudhari et al. include a proof to show S3Hash is input hiding provided the entropy of the input vector is sufficiently larger than the size of the hash [1]. Therefore, this must be taken into account when choosing sizes for the hashes and templates.

There are still some ways an attacker could attempt to use the system to learn information. For example, if they had access to a high-resolution image of someone’s iris, they could enter it into the system to try to detect if they are enrolled. This can be countered by ensuring the FAR is high enough to disguise whether a user’s iris has been entered into the system. Therefore, it may be preferable to choose a higher threshold which does not completely minimise the FAR.

### 5.6.3 Malicious Signers

Since the list of signers is restricted to trusted entities it is unlikely that any of them would be bad actors. Recall that new signers can only be added if the majority of ex-
isting signers vote to add them. However, there are some scenarios where a malicious entity could become a signer. For instance, a signer’s node operating the blockchain could be hijacked by a bad actor. In this case the malicious entity could attempt to corrupt the data by adding fake records or removing legitimate records. Fortunately, the blockchain provides a high level of transparency which should make this behaviour easy to detect. The other signers can quickly remove any bad actors from the system. The changes they made could then be rolled back. This would of course be disruptive to the system. However, it would be less dangerous than if a traditional system was compromised. This is because the system does not store any personally identifiable information. Furthermore, the high level of transparency in the blockchain would enable nodes to easily detect if someone made changes in their name. They could then take appropriate actions to secure their account.

5.7 Results Summary

This chapter has evaluated the system with regards its speed, efficiency, accuracy, memory usage and security. The presented results prove that the system is viable when parameters are specifically chosen for a dataset. A more thorough evaluation is needed to assess whether the results will be generalisable to new data. However, analysis indicates that they should. An overview of the different datasets’ selected parameters and achieved results is shown in Table 5.12. It is clear that the imaging conditions under which the pictures are taken have a large impact on the parameters required to maximise the results. It is also apparent that the quality of the images can greatly affect the results. The IITD dataset lacked significant intra-class variations so the results achieved are very good. Conversely the results for the CASIA-Iris-Syn dataset were much lower since they had major intra-class variations. The large number of rotations required for this dataset is probably due to the fact that the irises were rotated in the images to try to increase intra-class variations. The CASIA-Iris-Interval dataset is perhaps the most balanced. The results achieved on it probably give the best indication of the general performance of the system. Additionally, its selected parameters serve as good default values.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Irises</th>
<th>Template Size</th>
<th>Rotations</th>
<th>Decidability Index</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASIA-Iris-Interval</td>
<td>2443</td>
<td>11200</td>
<td>4</td>
<td>3.742</td>
<td>0.0276</td>
</tr>
<tr>
<td>CASIA-Iris-Syn</td>
<td>7340</td>
<td>7680</td>
<td>8</td>
<td>3.10538</td>
<td>0.0474</td>
</tr>
<tr>
<td>IITD Irises</td>
<td>1917</td>
<td>11520</td>
<td>1</td>
<td>4.917</td>
<td>0.0043</td>
</tr>
</tbody>
</table>

Table 5.12: Overview of Results
6 Conclusion

The primary objective of this study was to create and test a prototype of the blockchain based vaccination passport proposed by Chaudhari et al. [1]. With this in mind the study can be deemed a success. A prototype consisting of three major components has been successfully constructed. It has been tested and the results verify the system is viable. This chapter will summarise the research and identify potential areas for future work.

The system uses an iris-based biometric authentication system as suggested by Chaudhari et al. [1]. Masek’s Matlab code [13] is employed to generate feature vectors from iris images. Several enhancements were made to this program to improve its accuracy and performance. A few limitations were identified in Masek’s system. They will be discussed alongside potential future work.

The templates are hashed using the S3Hash algorithm proposed by Chaudhari et al. [1]. This is done to anonymise them. The algorithm is implemented in Python. The NumPy [33] and gmpy2 [34] libraries are used to make the code efficient. The parameter vectors were examined, and it was determined that some work better than others. Therefore, the parameter vectors are not selected randomly. Instead, parameter vectors that have an approximately equal probability of hashing a template to a 1 as a 0 and have a low correlation with each other are chosen. It is also noted that larger hashes produce better results.

The blockchain itself uses the Ethereum framework. It employs the Clique [25] PoA consensus mechanism to maximise efficiency. The vaccination records and iris hashes are stored using a smart contract. Read access to the contract is universally provided. However, write access is restricted to certain entities. Unfortunately, the blockchain does have some noticeable shortcomings which will be discussed in the future work section.

Despite the problems encountered, the evaluation results presented in Chapter 5 show promise. A set of objectives was specified in Chapter 1. The system is found to meet all these objectives. The amount of memory needed to run the system is deemed to be
acceptable. The time required to locate a user’s records is also judged to be reasonable. Finally, the blockchain is able to correctly identify most users and never misidentifies a user. This proves that the system is viable.

6.1 Future Work

6.1.1 Iris Extraction System

The major bottleneck preventing better recognition proved to be the iris extraction system. If a user’s iris templates are not similar than their hashes will not be either. Therefore, the system will be unable to identify them. Thus, a better iris template extraction system should significantly improve the results. It was determined that the weakest part of Masek’s code is the segmentation phase. Although this process was greatly improved, superior results have been reported in the literature [6, 7, 8]. Thus, further work in this area is likely to yield an improvement.

A more serious issue is that there appears to be no way for hashed templates to share a mask. As demonstrated in Chapter 5 this causes a noticeable decrease in accuracy. It remains an open problem in this research. One possible solution could be to employ an alternative iris extraction program which either minimises the reliance on masks or removes it entirely.

Rotational inconsistencies were also found to hamper the system’s accuracy. Recall that an inputted vector will be rotated and hashed multiple times. The minimum distance between its hashes and each stored hash will be used to determine its similarity to them. This process greatly increases the number of false positives. Therefore, an alternative approach to handling rotational inconsistencies could greatly improve the results. If a segmentation process that does not utilise masks was employed then the rotation invariant templates proposed by Damer et al. [9] could potentially be used to address this problem.

6.1.2 Blockchain

As previously noted, the blockchain had several shortcomings. Addressing these would increase the usability and security of the system. The modular design of the system allows another blockchain to be used instead of Ethereum. Hyperledger Fabric was judged to have some advantages over Ethereum. However, it is still not clear if it would fix any of the major issues identified in the system’s blockchain component.

The fact that entities cannot make anonymous transactions weakens the system’s se-
curity. Ideally the trusted entities would be able to upload the iris hashes anonymously to the blockchain. However, since this is not possible in the current system it is much easier for attackers to link a user’s iris hash to their vaccination records. This is because, in most cases, a user’s iris hash and vaccination records will be uploaded by the same trusted entity. Therefore, when an attacker is trying to link vaccination records and iris hashes, they only have to compare ones that were added by the same entity. Fortunately, significant work is currently ongoing to enable anonymous transactions in Ethereum. Anonymisation methods can build off ring signatures and non-interactive zero-knowledge proofs. This is an area which should be further investigated in order to improve the system’s security.

The current system uses Ether and contract permissions to limit the transactions a non-authorised entity can perform. However, as discussed in Chapter 4 this has some limitations. Ideally non signers should be prevented from making any transactions. This would also mean the contract would not have to maintain a separate list of authorised entities. Instead, it could rely solely on the list of signers maintained using the Clique protocol. This would greatly simplify the application and help to improve security.

6.1.3 Further Testing

Perhaps the most important area for future work is to further test the system. Due to time and resource constraints the testing was limited to running different Docker containers on a single machine. A more robust test could be performed by running the different nodes on separate machines. This could be done efficiently using cloud computing. For example, Amazon Web Services (AWS) could be used with each node running on a separate instance. This should give a better indication of the system’s performance as the nodes would not need to share computer resources. Additionally, it should provide an insight into how the system would run when scaled up.

Extra testing around the security of the system is particularly important. The usefulness of the system is entirely dependent on how secure it is. The system cannot be employed in the real world regardless of how efficient it is if it is not secure. The exact information that should be used to generate the unique identifier remains an open question. Running simulated attacks against the system should help to provide an answer and offer a greater insight into the system’s overall security.

Conducting a trial of the system in the field is necessary before it can be employed in the real world. It would offer a look at how usable and practical the system is for ordinary people in their everyday lives. It may provide valuable insights into the system’s viability and help to inform any necessary pivots.
6.1.4 Other Applications

This research has exclusively focused on applying the system proposed by Chaudhari et al. [1] to a blockchain based vaccination passport. However, the work in that paper and this dissertation could instead be applied to other areas.

Naturally the system could be adapted to a more general medical records storage system. This would of course require much stronger security. For example, it may be prudent to encrypt the records rather than just anonymising them. Furthermore, it should provide a guarantee that no attack against the system is feasible. Since medical records are much longer and more complex than vaccination records it would probably be impractical to store them directly on the blockchain. A more efficient solution would be to store them off-chain as discussed in Chapter 4.

Alternatively, a less security critical use for the system would be for event tickets. In this case, a user would provide an iris scan when requesting a ticket. The user would receive a reference number which would be used alongside their iris hash to create their unique identifier. The iris hash and unique identifier would be stored on the blockchain in the manner described in this paper. In order to obtain entry to the event the user would simply have to prove “ownership” of their identifier. This would be done by providing another iris scan and their reference number. The system would follow the same protocol outlined in this dissertation to reconstruct the identifier. This system might be considered overly elaborate for ordinary events. However, it offers better security than a traditional ticket system. Furthermore, it might serve a niche where a guarantee is required that the ticket is used by the same person who it is issued to. There are many examples of this kind of situation. Students enrolled in a module could be issued a ticket to ensure that they sit their own exams. If someone was required to have a medical check-up before attending an event this system could be employed to guarantee the same person attends the check-up as the event. This scenario is especially relevant in current times where many events require people to have been COVID tested prior to attending. The system could also be used to prevent ticket resellers. One obvious drawback of this application is that users could potentially be required to submit iris scans to multiple different entities. However, this should not be an issue if the entities are behaving honestly. Since the iris scans are hashed using an input hiding function no information about the user’s iris would be leaked. Additionally, provided that each event uses a unique set of hashing parameters it would be impossible to link a user’s different iris hashes. Of course, this system could be run on a private blockchain to further improve security at the cost of transparency.

More generally the system could be applied to any application where users have to prove “ownership” of something. Therefore, there is plenty of scope to expand this
research into other areas and to adapt it to other applications.
Bibliography


59


A1 Appendix

A1.1 CASIA-Iris-Syn Parameter Search

<table>
<thead>
<tr>
<th></th>
<th>120</th>
<th>160</th>
<th>200</th>
<th>240</th>
<th>280</th>
<th>320</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1.90391</td>
<td>1.55098</td>
<td>1.59814</td>
<td>1.67039</td>
<td>1.76294</td>
<td>1.82951</td>
</tr>
<tr>
<td>16</td>
<td>1.89265</td>
<td>1.55855</td>
<td>1.60619</td>
<td>1.71401</td>
<td>1.77561</td>
<td>1.84622</td>
</tr>
<tr>
<td>20</td>
<td>1.90591</td>
<td>1.58290</td>
<td>1.62843</td>
<td>1.70636</td>
<td>1.79817</td>
<td>1.85044</td>
</tr>
<tr>
<td>24</td>
<td>1.91138</td>
<td>1.58684</td>
<td>1.61620</td>
<td>1.74181</td>
<td>1.79562</td>
<td>1.86310</td>
</tr>
<tr>
<td>28</td>
<td>1.91777</td>
<td>1.58179</td>
<td>1.63404</td>
<td>1.71641</td>
<td>1.80376</td>
<td>1.84871</td>
</tr>
<tr>
<td>32</td>
<td>1.92626</td>
<td>1.57303</td>
<td>1.64353</td>
<td>1.72315</td>
<td>1.80375</td>
<td>1.84367</td>
</tr>
</tbody>
</table>

Table A1.1: Angular and Radial Resolution Comparison for CASIA-Iris-Syn

![Number of Rotations Comparison for CASIA-Iris-Syn](image)

Figure A1.1: Number of Performed Rotations Comparison for CASIA-Iris-Syn

A1.2 IITD Irises Parameter Search
<table>
<thead>
<tr>
<th></th>
<th>120</th>
<th>160</th>
<th>200</th>
<th>240</th>
<th>280</th>
<th>320</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4.20279</td>
<td>4.60531</td>
<td>4.77657</td>
<td>4.87771</td>
<td>4.85019</td>
<td>4.79500</td>
</tr>
<tr>
<td>20</td>
<td>4.26516</td>
<td>4.59830</td>
<td>4.81983</td>
<td>4.79050</td>
<td>4.84404</td>
<td>4.80709</td>
</tr>
<tr>
<td>32</td>
<td>4.33011</td>
<td>4.65603</td>
<td>4.85725</td>
<td>4.80121</td>
<td>4.83482</td>
<td>4.77046</td>
</tr>
</tbody>
</table>

Table A1.2: Angular and Radial Resolution Comparison for IITD Irises

![Number of Rotations Comparison for IITD Irises](image)

Figure A1.2: Number of Performed Rotations Comparison for IITD Irises