Biometric Matching by Hashing and Applications

Lea Grieder (328216)
Leila Sidjanski (330810)

Computer Science / Communication Systems
EPFL

June 2024

Responsible
Prof. Serge Vaudenay

Abstract

This report explores the application of fuzzy hashing for finger-vein biometric authentication, aiming to enhance security and efficiency in biometric systems. Finger-vein authentication, which leverages internal anatomical features, provides a unique security advantage as these features are less susceptible to replication or theft. The primary focus is on the development and assessment of two algorithms: preHash and postHash. These algorithms transform biometric data into secure hash values that maintain consistency despite slight variations in the input data. The preHash algorithm selects significant bits from the biometric data using a permutation keyed by a secure key, effectively reducing data dimensionality while preserving distinguishing features. The postHash algorithm then compresses these indices into a compact hash. This process is further bolstered by the incorporation of fuzzy extractors, which ensure that even if the stored data is compromised, reconstructing the original biometric data remains computationally infeasible. The experiments conducted involved extensive comparisons, demonstrating the algorithms' efficacy in maintaining a balance between security and performance. Despite some challenges, including the integration of previous work and managing extensive experiment runtimes, the developed system shows promise for secure and efficient biometric authentication.

Keywords: Fuzzy hashing, finger-vein biometric authentication, preHash, postHash, fuzzy extractors, biometric security, data compression, biometric hashing.

Contents

1 Introduction

1.1 Background

Authentication is the process of confirming the validity of a claimed identity seeking access to a system or resource. Over decades, authentication mechanisms have evolved from basic password systems in the 1960s to advanced methods such as multifactor authentication by the late 2010s, driven by a persistent commitment to combat evolving security risks while enhancing user convenience[2]. Various methods such as password-based authentication, certificate-based authentication, one-time passwords, multifactor authentication, and biometric authentication are employed[6].

Biometric authentication, which involves analyzing unique physical characteristics, is often considered more secure than traditional authentication methods due to the difficulty in duplicating biometric traits. This encompasses technologies such as facial recognition, fingerprint recognition, eye recognition, and voice recognition[5]. However, despite the enhanced security of biometric authentication, it is not immune to exploitation. For instance, fingerprints left on surfaces can be copied, or hackers may obtain images of individuals online to deceive authentication systems. Choosing the right authentication mechanism requires careful consideration of various factors including the necessary security level, ease of use for users, cost implications for setup and ongoing upkeep, as well as the unique risks and vulnerabilities pertinent to the system or data in question. Typically, the requisite level of security steers the selection process; for example, platforms managing sensitive personal information might mandate the use of robust authentication methods, such as biometric verification. The inherent challenge in deploying such secure systems lies in achieving a delicate equilibrium between high security measures and user convenience. The goal is to create an authentication process that is both seamless and efficient, ensuring that access is granted swiftly and accurately to the rightful user without necessitating multiple attempts, thus maintaining a user-friendly experience while upholding the highest security standards.

While advanced biometric systems typically rely on externally visible physical attributes, finger-vein authentication focuses on internal anatomical features, adding a unique layer of security, as they are less prone to replication or theft compared to external characteristics. Nevertheless, it is important to note that finger-vein authentication does not completely eliminate challenges. Despite its emphasis on internal features, attackers can exploit inherent structures in finger veins, such as common patterns among individuals and predictability in acquired data, which poses risks to the authentication process.

In light of these considerations, this project is dedicated to authentication using finger-vein features. This involves utilizing a specialized scanner equipped with two infrared cameras to capture finger veins from different angles. The registration process involves capturing an image, termed the model image, while the authentication process involves capturing another image, known as the probe image. These images undergo processing through a pipeline designed to extract and align the finger-vein

patterns. The pipeline outputs a feature vector, which is essentially a bitstring, where 0's represent where there are no finger veins, and 1's show where veins are present. Following this process, the system evaluates whether the feature vector extracted from the probe image sufficiently matches the feature vector of the model image associated with the individual attempting authentication.

1.2 Extraction Pipeline

This project extends the work on optimizing a finger-vein recognition pipeline that has demonstrated the lowest Equal Error Rate (EER) by incorporating a novel hashing step to process the output of the pipeline. The purpose of integrating Hash Functions within this context is multifold, but before delving into hash functions, which are central to our project, it's essential to outline the foundation upon which we have built our advancements. This initial context will provide a clearer understanding of the starting point from which our developments began.

Simon Sommerhalder and Burcu Yildiz have both made significant contributions to the system. Simon introduced an innovative approach to the alignment of freshly captured images (probe images) with those stored in the system (model images), ensuring the hashing process (following the alignment of the finger) is based on the unique finger-vein pattern rather than how the finger is positioned on the scanner.

Simon has developped a pipeline (see Figure 1.1) to align finger-vein images, enhancing security by eliminating the need to compare the model and probe images side by side. He organized the pipeline into six clear steps:

1. **Masking**: The first step of the pipeline isolates the finger area in the image. This involves creating a mask that outlines the finger, ensuring that subsequent processing focuses solely on the relevant part of the image.
2. **Prealignment**: This step involves adjusting the position and orientation of the finger within the image before extracting vein patterns. It's aimed at roughly aligning the image based on the finger's outline, helping to standardize the position of the finger across different scans.
3. **Histogram equalization**: To ensure the images have consistent lighting and contrast, this step adjusts the brightness levels. This makes the vein patterns more distinct and comparable across different images.
4. **Feature extraction**: Here, the actual vein patterns are identified and extracted from the image. The process converts the visual image into a digital format that represents the presence or absence of veins at specific locations.
5. **Postalignment**: After extracting the vein patterns, this step fine-tunes the alignment of the image. It's based on the vein patterns themselves, ensuring that the comparison between model and probe images is as accurate as possible.
6. **Distance Calculation**: The final step involves comparing the feature vector of the probe image with that of the model image. This is done using a specific

metric to quantify the similarity between the two, ultimately determining if they match closely enough for authentication to succeed.

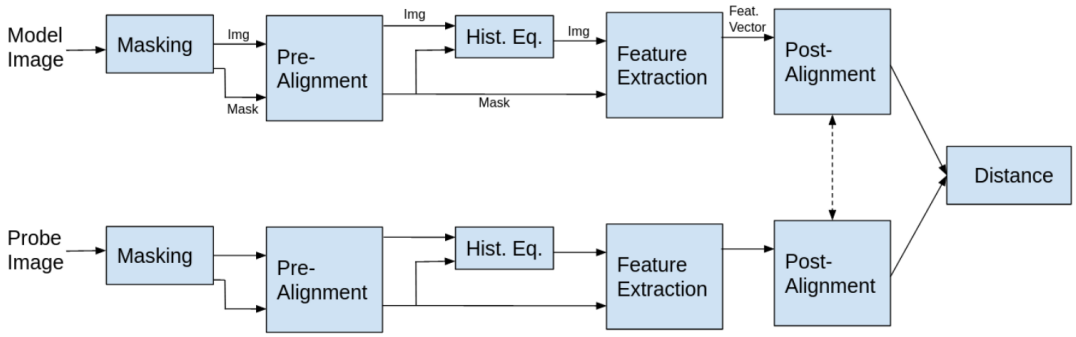


Figure 1.1: Simon's Extraction Pipeline. Compares a single probe image to a model image

Burcu's work on the finger vein authentication project built upon Simon's foundational pipeline, focusing on refining image processing for vein extraction and evaluating different distance calculation methods for authentication. She optimized preprocessing steps, investigated masking and prealignment issues, and tackled reference selection to enhance matching accuracy. A significant part of her contributions involved exploring fuzzy extractors Fuzzy Extractors to bolster security, conducting a thorough analysis of the dataset to identify and address challenges, and proposing solutions to improve the system's reliability and efficiency.

Simon and Burcu explored a variety of function combinations at different stages of the authentication pipeline, aiming to pinpoint the configuration that yields the most favorable outcomes. They utilized the Equal Error Rate (EER) as a benchmark to measure the pipeline's efficacy, focusing on achieving a balance between security and accessibility. This metric, representing the point where the rate of false acceptances (impostor incorrectly granted access) matches the rate of false rejections (legitimate user incorrectly denied), serves as an indicator of the system's reliability and accuracy. The configuration that demonstrated superior performance, leading to the lowest EER and thereby optimizing the process of analyzing and processing images, is showcased in the subsequent figure.

Table 1: EER values for the best pipeline: Edge masking, translation pre-alignment, no pre-processing, no post- processing, Miura matching as the post-alignment

Camera	EER
Camera 1	0.041
Camera 2	0.029
Sum of cameras	0.030

1.3 Project Overview

In the progression of our project, building upon the work of Simon and Burcu, we aim to address the inherent variability in biometric data, particularly with finger vein patterns, by considering hash functions. The final objective of the system is to add a hashing step at the end of our established pipeline, prior to the post-alignment phase. The integration of this step serves to enhance security and overall functionality by allowing us to store a hash of each biometric image X in our database, instead of the raw biometric data. Upon receiving a new image Y, we compute hashes for all potential translations of Y, comparing these with the hash of X. Unfortunately, this process is computationally expensive because it requires computing the hash for every possible translation of the image. The purpose of employing a hashing process in this system is multifold:

- **Security**: Hash values can be stored instead of raw biometric data. In the event of a database breach, attackers would find it significantly more challenging to reconstruct the original biometric information from the hashed values due to the one-way nature of hash functions.
- **Consistency**: By focusing on the unique patterns of the biometric trait (like finger-vein patterns) and standardizing how this data is processed and hashed, the system aims to produce consistent hash values for the same individual across different scanning sessions. This is crucial for reliable authentication, ensuring that minor variations in finger placement do not affect the system's ability to recognize the user.
- **Performance**: Hashing biometric data into a compact, fixed-size format facilitates quicker comparison and verification processes. It's more efficient to compare hash values than to perform complex pattern recognition operations on raw biometric images.

Traditional hash functions, while pivotal in various data security contexts, generate a unique output for each unique input. This one-to-one mapping means even minor variations in the input — common in biometric data due to natural changes in biological traits or differences in scanning conditions — result in completely different hashes. This sensitivity to input variability poses a challenge in biometric authentication systems, where the goal is to accurately recognize and authenticate an individual despite these natural variations.

Fuzzy hashing stands as a sophisticated solution to this challenge. Unlike traditional hash functions, fuzzy hashing is designed to produce consistent cryptographic keys for inputs that are similar, but not identical. This is particularly advantageous in biometric authentication systems, where it's essential to recognize the same biometric trait across different instances, despite slight variations. The "fuzziness" of this approach allows the system to map these similar inputs to the same or closely related hash values, thereby ensuring that legitimate users are not incorrectly denied access due to minor discrepancies in their biometric data. Furthermore, the application of fuzzy hashing in our pipeline is instrumental in protecting user privacy. Since the hashed values, rather than raw biometric data, are stored and used for authentica-

tion, users' biometric information is safeguarded against potential breaches. Even if hashed values were accessed without authorization, the complexity of fuzzy hashing algorithms makes it extremely challenging to reverse-engineer the original biometric data.

The process of our fuzzy hashing algorithm, that we will detail in Section 3 begins with a biometric capture, a finger image, which goes through the already developped pipeline 1.1 to extract a bitstring. This bitstring undergoes a pre-hashing process, where a subset of significant bits (denoted as vein pixels) is selected based on a permutation keyed by a secure key, resulting in a tuple that significantly reduces the data's dimensionality while preserving its distinguishing features.

Upon generating the fuzzy hash, the next step involves further compressing this hash to prepare it for storage. This compression is achieved through a function, postHash, which maps the tuple to a bitstring of a defined length. The output, essentially a compressed fuzzy hash, exhibits nearly uniform distribution when both the input data and the key are random, enhancing security and storage efficiency.

To further bolster the security of the stored biometric data, this system incorporates Fuzzy Extractors. The fuzzy extractor framework ensures that even if the stored data is compromised, reconstructing the original biometric data or compromising individual privacy remains computationally infeasible. This is accomplished by generating a secure, random key from the biometric input using a generation process (Gen) and allowing for the reliable reproduction of this key from an approximation of the original input using a reproduction process (Rep), without directly storing the biometric data itself.

In essence, we aim to store the output of the fuzzy extractor, which includes the reproducible cryptographic key and the helper data, instead of the raw biometric data or its direct hash. This method ensures that the stored biometric data is not only compact and efficiently stored but also securely obfuscated, requiring the correct biometric input for any form of decryption or matching to occur.

2 Biometric Setting

In this section, we explore the representation and processing of biometric data, specifically focusing on finger vein images. These images are converted into bitstrings to enhance processing efficiency. Each bitstring represents the presence or absence of vein pixels, forming the basis for biometric templates. The similarity between two biometric captures is quantified using Hamming weights and distances, enabling the calculation of a similarity score. This score is crucial for authentication and identification processes, ensuring accurate and reliable matching of biometric data.

2.1 Biometric Data and Similarity Scores

We begin by delving into the representation of the biometric data. Finger images, designated as biometric templates and formated as 240×376, result in $n = 90'240$ pixels per image. To enhance processing efficiency, these templates are converted into vectors, departing from their original 2-dimension image structure.

In the biometric context, each finger serves as a biometric subject, with a corresponding biometric capture represented as a bitstring X of length n. This bitstring encapsulates the specific vein pixel information extracted from the finger image, while the biometric template serves as a reference model derived from these images.

Each of the n bits of the biometric capture is designated as $X_1, \ldots, X_n$, with X_i set to 1 if the i-th bit corresponds to a vein and 0 otherwise. We define the random variable $\left(\frac{\mathrm{HW}(X)}{n}\right)$, as the Hamming Weight of a feature vector[1] X divided by the total number of bits in the vector. In the case where i is a uniformly distributed random index and X is randomly chosen, we define the value p[2] as the mean of this random variable.

$$p = \mathbb{E}\left(\frac{\mathrm{HW}(X)}{n}\right) = Pr\left[X_i = 1\right] \tag{2.1}$$

The experimental results for the mean (p), along with the variance ($\sigma_p{}^2$)[3] and standard deviation (σ_p) of the random variable $\left(\frac{\mathrm{HW}(X)}{n}\right)$ across all images will be detailed in Section 2.2, and are quantified as follows:

$$p^{\mathrm{obs}} = \mathbb{E}\left(\frac{\mathrm{HW}(X)}{n}\right) = 3.51\% \tag{2.2}$$

[1]The terms "feature vector" and "bitstring" are used interchangeably throughout this report to refer to the vein pixel information extracted from the finger image.

[2]Throughout this report, experimentally derived variables are denoted with the superscript "obs" (e.g., $\mathrm{var}^{\mathrm{obs}}$), while theoretical variables are presented without any superscript.

[3]The notation, σ_p^{obs} denotes the observed standard deviation, and $(\sigma_p^{\mathrm{obs}})^2$ represents the observed variance of the random variable $\left(\frac{\mathrm{HW}(X)}{n}\right)$. Here p refers to the mean of this random variable. These notations will be used consistently throughout our report.

$$\sigma_{\mathrm{p}}^{\mathrm{obs}} = \sqrt{\mathbb{E}\left[\left(\frac{\mathrm{HW}(X)}{n} - p^{\mathrm{obs}}\right)^2\right]} \approx 5.02 \times 10^{-3} \tag{2.3}$$

$$(\sigma_{\mathrm{p}}^{\mathrm{obs}})^2 = \mathbb{V}\left(\frac{\mathrm{HW}(X)}{n}\right) \approx 2.52 \times 10^{-5} \tag{2.4}$$

In biometric authentication and identification, the uniqueness of each biometric capture is encoded in its bits. The objective revolves around discerning the similarity between two biometric captures to verify or identify two individuals. Therefore, the scoring mechansim plays a crucial role in quantatively determining the similarity between biometric captures. This similarity score holds significance in verifying the identity of an individual (authentication) or identifying potential matches in a database (identification). A higher score indicates a greater ressemblance between captures, while a lower score suggests less similarity. This scoring mechanism is indispensable for ensuring the accuracy and reliability of biometric systems. The score of (X, Y) is computed as

$$\begin{aligned} Score(X,Y) &= \frac{HW(X \wedge Y)}{HW(X) + HW(Y)} \\ &= \frac{1}{2} - \frac{1}{2}\frac{d_H(X,Y)}{HW(X) + HW(Y)} \end{aligned} \tag{2.5}$$

where HW denotes the Hamming Weight and d_H the Hamming Distance.

In conjunction with the scoring mechanism, Miura matching emerges as a specialized technique for comparing biometric samples. This method entails determining an optimal offset translation, denoted as d_X and d_Y, between two biometric samples to align their features for comparison, thereby compensating for differences in positioning or orientation. The alignment process maximizes the similarity score, defined as:

$$Score = \mathrm{Score}(d_X \cdot X, d_Y \cdot Y)$$

Once the optimal offsets, d_X and d_Y, are determined, they are applied to the original samples for alignment. This results in the calculated aligned positions, denoted as $\bar{X}$ and $\bar{Y}$, such that:

$$\begin{aligned} \bar{X} &= d_X \cdot X \\ \bar{Y} &= d_Y \cdot Y \end{aligned}$$

The probability of the i-th pixel of two captures not being the same after applying the optimal offset translations depends on the distribution of $(\bar{X}, \bar{Y})$, and is denoted

as δ. We define the random variable $\left(\frac{d_H(\bar{X},\bar{Y})}{n}\right)$ as the normalized Hamming distance between two feature vectors. The parameter δ is defined as the mean of this random variable, and we additionally define the standard deviation and the variance of this random variable, as shown in the following figures (where i is a uniformly distributed random index).

$$\delta = \mathbb{E}\left(\frac{d_H(\bar{X},\bar{Y})}{n}\right) = Pr[\bar{X}_i \neq \bar{Y}_i] \tag{2.6}$$

$$\sigma_\delta = \sigma\left(\frac{d_H(\bar{X},\bar{Y})}{n}\right) \tag{2.7}$$

$$(\sigma_\delta)^2 = \mathbb{V}\left(\frac{d_H(\bar{X},\bar{Y})}{n}\right) \tag{2.8}$$

Distinguishing between the types of distributions associated with the two captures is crucial. When both captures originate from the same biometric subject, it is referred to as δ_{same}. Conversely, if the captures are from different subjects, it is labeled as δ_{diff}. Additionally, when $\bar{X}$ and $\bar{Y}$ consist of $2n$ independent random bits with an expected value of p and no optimal offset is applied, it is classified as δ_{indep}. In this scenario, $\delta^{\text{obs}}_{\text{indep}} = 2p^{\text{obs}}(1-p^{\text{obs}}) = 6.78\%$. As will be detailed in Section 2.3, the experimental results produce the following figures:

	δ^{obs}	$(\sigma^{\text{obs}}_\delta)^2$	$\sigma^{\text{obs}}_\delta$
Same Biometric Subjects	4.55%	1.06×10^{-4}	1.03×10^{-2}
Different Biometric Subjects	5.99%	4.58×10^{-5}	6.76×10^{-3}

Table 2: Comparison of Distributions: Mean, Variance, and Standard Deviation of the random variable $\left(\frac{d_H(\bar{X},\bar{Y})}{n}\right)$ when both captures originate from the same and from different biometric subjects

Utilizing the formulas for p (2.1) and δ (2.6), the joint distribution for $(\bar{X}_j,\ \bar{Y}_j)$ across j random instances is derived:

	$\bar{Y}_i = 0$	$\bar{Y}_i = 1$
$\bar{X}_i = 0$	$1 - p - \frac{\delta}{2}$	$\frac{\delta}{2}$
$\bar{X}_i = 1$	$\frac{\delta}{2}$	$p - \frac{\delta}{2}$

Table 3: Joint Distribution of $(\bar{X}_j,\ \bar{Y}_j)$ for Random Instances

2.2 Experimental Derivation of the Probability p

In order to derive the probability that a distributed random pixel is a vein (p), a dataset of 20 unique individuals was utilized. Each individual contributed images of both right and left index/middle fingers, with 5 trials per finger, captured using 2 different cameras. This methodology resulted in a comprehensive dataset of 800 images. Following the approach in Section 1.2, Simon's optimal pipeline was implemented to extract the feature vectors from each image in the dataset (refer to Figure 1.1). Subsequently, statistical analysis was performed on these feature vectors to gain insights into vein patterns, building on the preliminary work by Burcu. It's important to note that the calculation of the probability p does not take into account any postalignment method yet.

In order to derive the mean of $\left(\frac{HW(X)}{n}\right)$, i.e. p, we have implemented an algorithm which processes the dataset's feature vectors, performing two main functions:

1. For each pixel position, it updates an *veins[i]* array, which accumulates detections of veins across all images for each camera $i = 1, 2$.
2. It maintains a count of the number of images analyzed per camera $i = 1, 2$ in an *image_count[i]* array.

Following the processing of all feature vectors, the algorithm computes the probability of a pixel being part of a vein by dividing the aggregated vein detections by the total number of images analyzed, across both cameras. Visual representations were generated to illustrate the distribution of detected vein pixels for each camera individually and for the combined data from both cameras.

1. **Camera 1 and Camera 2 distribution**: The following histograms showcase the frequency distribution of vein pixels detected in images captured by camera 1 and 2 individually, with a Gaussian fit overlaid to highlight the data's normal distribution trend.

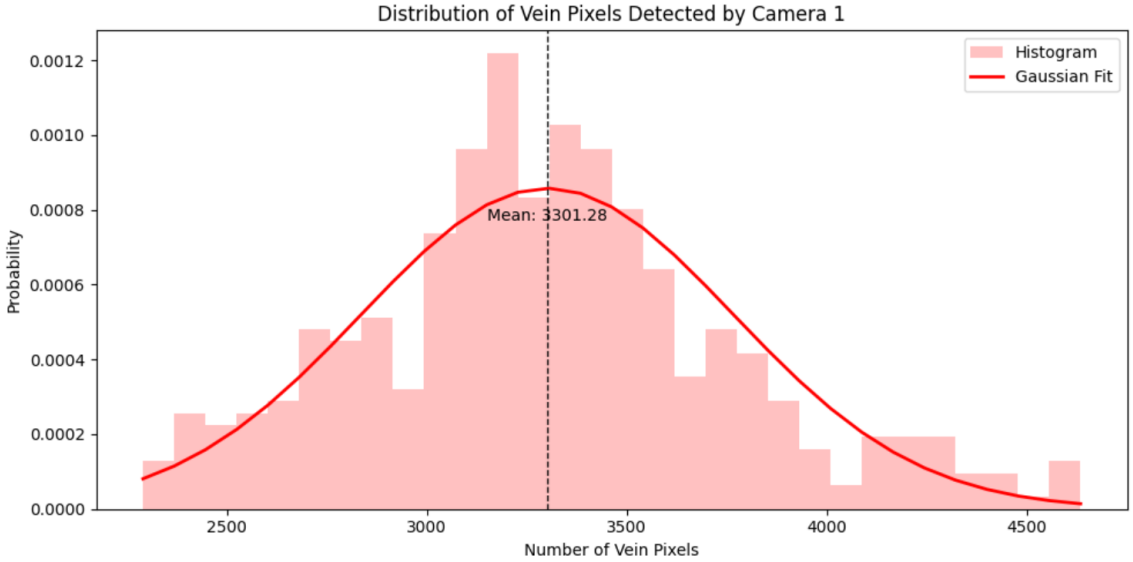


Figure 2.1: Vein Pixel Distribution Analysis Using Camera 1: Histogram Representation with Gaussian Fit Overlay with $X_i \sim \mathcal{N}(3301.28, 216433.17)$

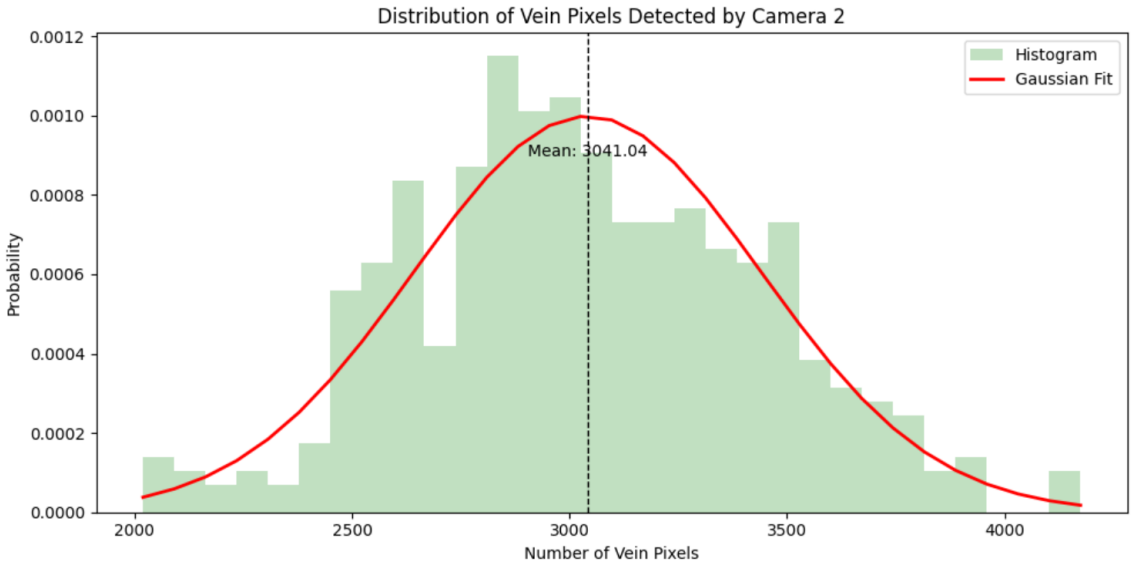


Figure 2.2: Vein Pixel Distribution Analysis Using Camera 2: Histogram Representation with Gaussian Fit Overlay $X_i \sim \mathcal{N}(3041.04, 159577.13)$

2. **Combined Cameras Distribution**: To understand the aggregate behavior of vein detection across both cameras, the data was merged, generating a comprehensive histogram with a Gaussian fit.

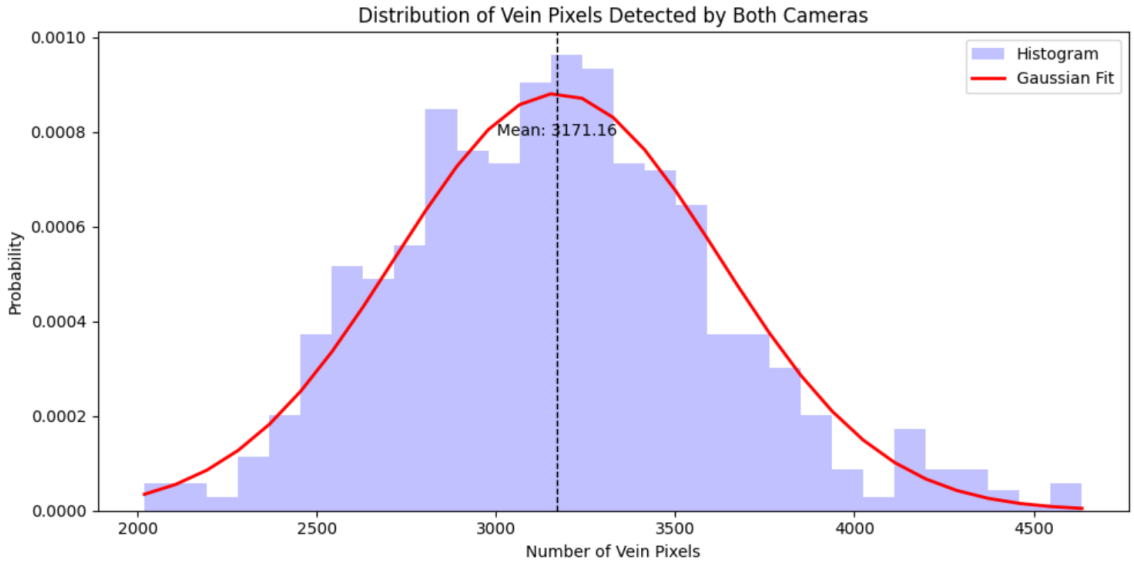


Figure 2.3: Aggregate Vein Pixel Distribution Analysis: Combined Histogram and Gaussian Fit from Both Cameras $X_i \sim \mathcal{N}(3171.16, 204935.79)$

Concluding the analysis of the vein pixel distribution across different camera captures and their aggregate dataset, the average probability p that a given pixel corresponds to a vein was computed. This involved summing up all vein-identifying pixels within the datasets for Camera 1, Camera 2, and the combined dataset. Subsequently, this sum was divided by the total number of pixels processed, multiplying the count of images by the total pixels per image, to ascertain the average likelihood p that any randomly selected pixel is part of a vein pattern.

The findings from this analysis revealed the following probabilities:

1. For camera 1, the probability $p^{\text{obs}} = \mathbb{E}\left(\frac{\text{HW}(X)}{n}\right)$ was calculated to be 0.03658, indicating a 3.66% chance that any given pixel in images from Camera 1 represents a vein.

2. For camera 2, the probability $p^{\text{obs}} = \mathbb{E}\left(\frac{\text{HW}(X)}{n}\right)$ was slightly lower at 0.03369, translating to a 3.37% chance for vein representation in its images.

3. When considering the datasets from both cameras combined, the probability $p^{\text{obs}} = \mathbb{E}\left(\frac{\text{HW}(X)}{n}\right)$ averaged out to 0.03514, suggesting a 3.51% likelihood of a pixel depicting a vein across the entire dataset. Moreover, the variance $((\sigma_p^{\text{obs}})^2)$ and the standard deviation (σ_p^{obs}) of the random variable $\left(\frac{\text{HW}(X)}{n}\right)$ provide insights into the the random variable's spread and consistency. The observed variance is low, at approximately 2.52×10^{-5}, indicating minimal fluctuation in the random variables' values among different feature vectors.

This suggests that the feature extraction method is highly reliable and consistent across different images. Furthermore, the standard deviation, calculated as approximately 5.02×10^{-3}, reinforces the idea of minimal dispersion around the mean probability (p) of detecting a vein.

2.3 Experimental Derivation of the Probabilities $\delta_{\text{same}}, \delta_{\text{diff}}, \delta_{\text{indep}}$

To assess the probability that the i-th pixel of two captures diverges after applying the optimal offset translations d_X and d_Y (Equation 2.6), we undertake additional experimental analysis. This investigation continues to utilize the same dataset comprising 20 distinct individuals. Here, the post-alignment process, specifically Miura Matching, is included in our examination as the calculation formula integrates this step. As detailed in Section 2.1, our objective is to evaluate three distinct delta values: δ_{same}, δ_{diff}, and δ_{indep}.

The computation of δ_{indep} is relatively straightforward, having determined p^{obs}. In order to calculate the mean, the standard deviation, and the variance of the random variable $\left(\frac{d_H(\bar{X},\bar{Y})}{n}\right)$ when X and Y are made up of $2n$ independent random bits of expected value p and without applying the optimal offset to get $\bar{X}$ and $\bar{Y}$, we can proceed as follows:

$$\delta_{\text{indep}}^{\text{obs}} = 2p^{\text{obs}}(1-p^{\text{obs}}) = 2 \times 0.03514 \times (1 - 0.03514) \approx 6.78\%$$

Next, the standard deviation and the variance of the normalized Hamming distance are given as:

$$\sigma_{\delta_{\text{indep}}}^{\text{obs}} = \sqrt{\mathbb{E}\left[\left(\frac{d_H(X,Y)}{n} - \delta_{\text{indep}}\right)^2\right]} = \sqrt{\frac{4p^{\text{obs}}(1-p^{\text{obs}})(1-2p^{\text{obs}}(1-p^{\text{obs}}))}{n}} \approx 1.18{\times}10^{-3}$$

$$(\sigma_{\delta_{\text{indep}}}^{\text{obs}})^2 = \mathbb{V}\left(\frac{d_H(X,Y)}{n}\right) = \frac{4p^{\text{obs}}(1-p^{\text{obs}})(1-2p^{\text{obs}}(1-p^{\text{obs}}))}{n} \approx 1.40 \times 10^{-6}$$

To accurately evaluate δ_{same} and δ_{diff}, a meticulous procedure was executed on our dataset, which involves 20 unique individuals, each having multiple biometric captures. Below is an outline of the methodology applied:

1. **Pairwise Comparison**: For each individual, we compared every biometric capture to every other capture within the dataset, applying the Hamming distance metric to compute the normalized distances.
2. **Statistical Analysis**: We organized the resulting distances into two distinct categories:

- Intra-individual distances, where captures from the same individual (same finger images) were compared, forming the first group.
- Inter-individual distances, comprising comparisons between biometric captures from different individuals, forming the second group.

3. **Probability Computation**:
   - For δ_{same}, we calculated the average normalized distances from intra-individual comparisons. This analysis provided the following results, quantifying the observed differences between captures from the same individual after alignment:

$$\delta_{\text{same}}^{\text{obs}} \approx 4.55\%$$
$$\sigma_{\delta_{\text{same}}}^{\text{obs}} \approx 1.03 \times 10^{-2}$$
$$(\sigma_{\delta_{\text{same}}}^{\text{obs}})^2 \approx 1.06 \times 10^{-4}$$

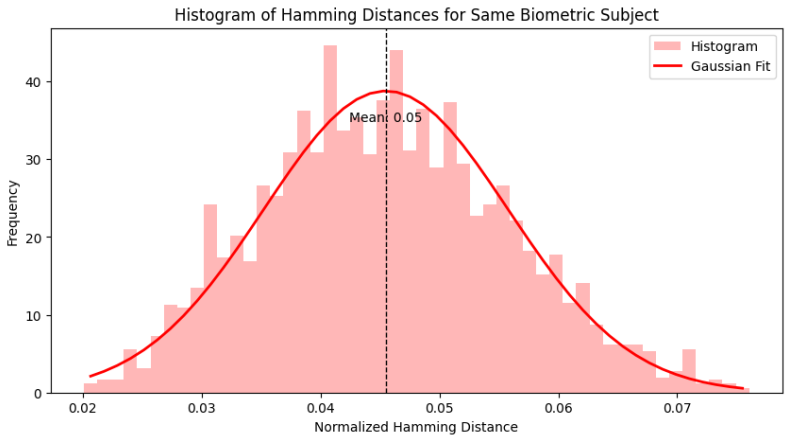


Figure 2.4: Frequency Distribution of Normalized Hamming Distances for Identical Biometric Samples with Alignment $\delta_{\text{same}} \sim \mathcal{N}(0.05, 1.06 \times 10^{-4})$

- For δ_{diff}, we performed a similar averaging of the normalized distances from the inter-individual comparisons, which furnished us with the following results for observing a difference between captures from the different individuals after alignment:

$$\delta_{\text{diff}}^{\text{obs}} \approx 5.99\%$$
$$\sigma_{\delta_{\text{diff}}}^{\text{obs}} \approx 6.76 \times 10^{-3}$$
$$(\sigma_{\delta_{\text{diff}}}^{\text{obs}})^2 \approx 4.58 \times 10^{-5}$$

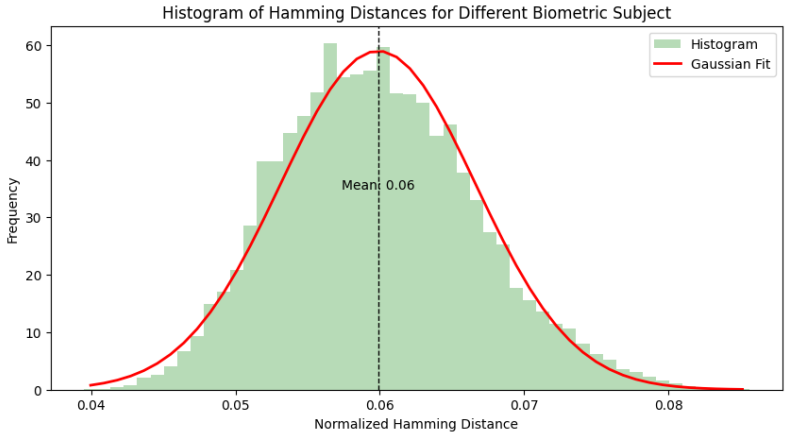


Figure 2.5: Frequency Distribution of Normalized Hamming Distances for Different Biometric Samples with Alignment $\delta_{\text{diff}} \sim \mathcal{N}(0.06, 4.58 \times 10^{-5})$

The joint probability distribution presented in Table 3 represents the relationship between the variables $\bar{X}_i$ and $\bar{Y}_i$, which compare the i-th bit of two biometric samples. The parameter p denotes the probability of a bit being 1, and by complement, $1 - p$ signifies the probability of a bit being 0. The parameter δ encapsulates the probability that the two bits are not identical.

In Table 3, we explain each entry:

- By utilizing the previously determined value of δ, we can infer the probabilities $Pr[\bar{X}_i = 1, \bar{Y}_i = 0]$ and $Pr[\bar{X}_i = 0, \bar{Y}_i = 1]$ as follows:

$$\delta = Pr[\bar{X}_i \neq \bar{Y}_i] = Pr[\bar{X}_i = 1, \bar{Y}_i = 0] + Pr[\bar{X}_i = 0, \bar{Y}_i = 1]$$

  Hence we have:

$$Pr[\bar{X}_i = 1, \bar{Y}_i = 0] = Pr[\bar{X}_i = 0, \bar{Y}_i = 1] = \frac{\delta}{2}$$

- To determine the probability$Pr[\bar{X}_i = 0, \bar{Y}_i = 0]$, we consider the following relationships and utilize the law of total probability:

$$Pr[\bar{X}_i = 0] = 1 - p = Pr[\bar{X}_i = 0, \bar{Y}_i = 0] + Pr[\bar{X}_i = 0, \bar{Y}_i = 1]$$

$$Pr[\bar{Y}_i = 0] = 1 - p = Pr[\bar{X}_i = 0, \bar{Y}_i = 0] + Pr[\bar{X}_i = 1, \bar{Y}_i = 0]$$

By solving these equations, we can derive the probability of interest, $Pr[\bar{X}_i = 0, \bar{Y}_i = 0]$, and hence infer the previously obtained values for $Pr[\bar{X}_i = 1, \bar{Y}_i = 0]$ and $Pr[\bar{X}_i = 0, \bar{Y}_i = 1]$:

$$Pr[\bar{X}_i = 0, \bar{Y}_i = 0] = 1 - p - \frac{\delta}{2}$$

- The probability $Pr[\bar{X}_i = 1, \bar{Y}_i = 1]$ is derived by

$$Pr[\bar{X}_i = 1, \bar{Y}_i = 1] = 1 - Pr[\bar{X}_i = 0, \bar{Y}_i = 0] = p - \frac{\delta}{2}$$

Utilizing $\delta^{\text{obs}}_{\text{same}}$, $\delta^{\text{obs}}_{\text{diff}}$ and $\delta^{\text{obs}}_{\text{indep}}$, we can determine the corresponding probabilities presented in Table 3. This gives us the following figures:

	$Y_i = 0$	$Y_i = 1$
$X_i = 0$	$(1-p)^2 = 93.1\%$	$p(1-p) = 3.39\%$
$X_i = 1$	$p(1-p) = 3.39\%$	$p^2 = 0.12\%$

Table 4: Joint Distribution of $(X_j,\ Y_j)$ for X and Y made up of $2n$ independent random bits of expected value p

	$\bar{Y}_i = 0$	$\bar{Y}_i = 1$
$\bar{X}_i = 0$	$1 - p - \frac{\delta^{\text{obs}}_{\text{same}}}{2} = 94.21\%$	$\frac{\delta^{\text{obs}}_{\text{same}}}{2} = 2.27\%$
$\bar{X}_i = 1$	$\frac{\delta^{\text{obs}}_{\text{same}}}{2} = 2.27\%$	$p - \frac{\delta^{\text{obs}}_{\text{same}}}{2} = 1.24\%$

Table 5: Joint Distribution of $(\bar{X}_j,\ \bar{Y}_j)$ for X and Y Originating from the Same Biometric Subject

	$\bar{Y}_i = 0$	$\bar{Y}_i = 1$
$\bar{X}_i = 0$	$1 - p - \frac{\delta^{\text{obs}}_{\text{diff}}}{2} = 93.49\%$	$\frac{\delta^{\text{obs}}_{\text{diff}}}{2} = 2.99\%$
$\bar{X}_i = 1$	$\frac{\delta^{\text{obs}}_{\text{diff}}}{2} = 2.99\%$	$p - \frac{\delta^{\text{obs}}_{\text{diff}}}{2} = 0.52\%$

Table 6: Joint Distribution of $(\bar{X}_j,\ \bar{Y}_j)$ for X and Y Originating from Different Biometric Subjects

3 Fuzzy Hashing

Fuzzy hashing, as opposed to traditional hashing, produces consistent cryptographic keys for similar but not identical inputs, enabling recognition of the same biometric trait across different instances despite slight variations. This approach ensures legitimate users are not incorrectly denied access due to minor discrepancies and protects user privacy by storing and using hashed values instead of raw biometric data, making it difficult to reverse-engineer the original data even if unauthorized access occurs. This section will discuss how we implemented the fuzzy hashing algorithm, its corresponding mathematical aspects and some experiments.

3.1 PreHash Algorithm

The preHash algorithm is the first step in the fuzzy hashing process, designed to manipulate biometric templates extracted from finger vein patterns. It operates on a bitstring X, representing the presence (1) or absence (0) of vein pixels across n pixels, where $n = 90'240$.

Algorithm Inputs and Outputs:

1. **Inputs**: As illustrated in the pseudocode from Algorithm 1, it takes three main inputs:
   - **A parameter m**: the number of indices to find
   - **A bitstring X**: the feature-extracted vein patterns of a biometric capture
   - **A key**: used to initialize a Pseudorandom Number Generator (PRNG)
2. **Output**: The algorithm outputs a tuple $(i_1, ..., i_m)$ consisting of the m smallest indices i_j such that $1 \leq i_1 < ... < i_m$ and the pixel at $PRNG_{\text{key}}(i_j)$ in X is identified as a vein pixel.

Detailed Process of preHash:

- **Initiallization**: Utilizing the provided key, the algorithm initializes a PRNG. This PRNG is based on the Advanced Encryption Standard (AES) in Counter (CTR) mode, ensuring the generation of uniform and independent pseudorandom sequences.
- **Nonce Generation**: A nonce in CTR mode encryption is initialized to zero to maintain simplicity and security. We opted against using a keyed hash function to generate the nonce, as it would tie the nonce to the secret key. Such a dependency would mean that both the nonce and the Pseudo-Random Number Generator (PRNG) would rely on the same key, creating a security risk by concentrating security on a single element. To avoid this, we keep the generation of the nonce separate from the key.
- **Pseudorandom Sequence Generation**: Upon initialization with the specified parameters—key and nonce—the PRNG operates in Counter (CTR) mode

to generate a sequence of pseudo-random numbers. As illustrated in Figure 3.1, the process involves encrypting an incrementing counter combined with the nonce using the specified key. The output from this block cipher encryption is used to generate a unique stream of pseudo-random bits. To optimize the utilization of the cryptographic output and reduce operational overhead, the PRNG is designed to generate a full 128-bit block of pseudo-random data at a time, even though only a portion of these bits are used immediately. When a new block is generated, the algorithm extracts the lowest 17 bits to obtain the pseudo-random number for the current operation. This choice is aligned with the project's requirement for representing image sizes, where a maximum of $90'240$ pixels can be represented, necessitating 17 bits per operation ($\lceil \log_2(90'240) \rceil = 17$). These extracted bits serve as the pseudo-random number. Following extraction, the algorithm discards the used bits from the current block by right-shifting the block by 17 bits. This operation updates the state of the PRNG, ensuring that the consumed bits are no longer considered in subsequent operations. In the case where there are no longer 17 bits available after extraction, the algorithm generates a new pseudo-random number using the PRNG and repeats the process. This mechanism guarantees the efficient utilization of all bits produced by the block cipher, enhancing efficiency and reducing operational overhead. The predictability and reproducibility of the sequence are entirely dictated by the chosen key. Employing the same key across sessions guarantees the generation of an identical sequence of pseudo-random numbers.

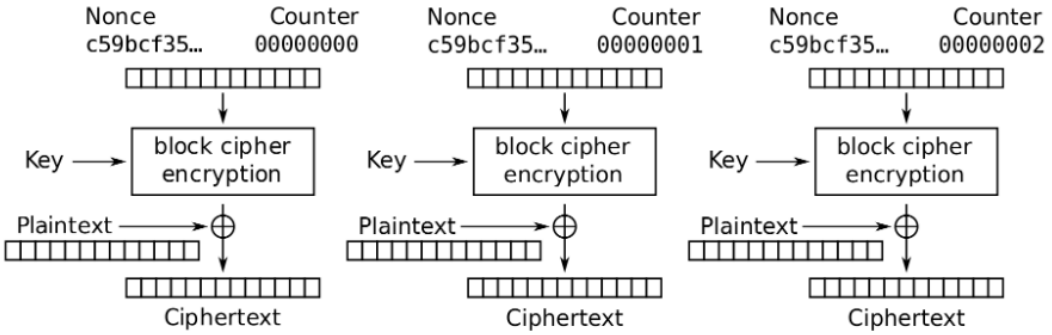


Figure 3.1: Counter (CTR) mode encryption[1]

- **Selection of Indices**: The algorithm iterates through the generated pseudo-random sequence, selecting the first m indices corresponding to vein pixels in the biometric template X. This selection process involves a careful mechanism to ensure the uniqueness and proper ordering of indices.
- **Corner Cases**: In scenarios where there are no veins present in the image, the algorithm incorporates a built-in mechanism to address such situations. Prior to initiating the preHash process, it assesses whether at least one vein is present in the image. If no veins are detected, the algorithm suspends further execution, preventing an infinite loop.

Algorithm 1 preHash Algorithm

function $\text{preHash}^{m}_{\text{KEY}}(X)$
    ▷ $L \leftarrow \lceil \log_2(n) \rceil$
    Initialize PRNG using key for L bits
    $i \leftarrow 0$
    for $j \leftarrow 1$ to m **do**
        repeat
            repeat
                $k \leftarrow$ PRNG
            until $1 \leq k \leq n$
            $i \leftarrow i + 1$
        until X_k is a vein pixel
        $i_j \leftarrow k$
    end for
    return $i_1, \ldots, i_m$
end function

The algorithm, preHash, illustrated in Figure 1 generates m smallest indices, denoted by i_j, such that $j \in [1, m]$ and $1 \leq i_1 < ... < i_m$, where each i_j correponds to an index such that $X_{PRNG_{\text{key}}(i_j)} = 1$. It achieves this by rigorously verifying that the numbers produced by PRNG stay within the specified bounds, i.e. $\text{PRNG}[i] \in (0, n]$ for all $i \in [0, m]$.

3.2 Assessing Similarity of Biometric Inputs After PreHash Application

After the finger images are processed through the pipeline described in Figure 1.1 to extract their feature vectors, and the preHash algorithm is applied, the outcome is a set of indices that fall within the inclusive range $\text{PRNG}[i] \in (0, n]$ for $i \in [0, m]$, effectively mapping each selected feature to a unique index within the feature vector's length.

In the context of a simplified scenario where the hash length parameter (m) is set to 1, implying the generation of a single-index hash, and assuming a randomly chosen key for the preHash algorithm, along with k representing a uniformly distributed random index, the probability that the preHash operation yields the same index for two different fixed inputs X and Y can be mathematically delineated as follows:

$$
\begin{aligned}
Pr[preHash^1_{\text{key}}(X) = preHash^1_{\text{key}}(Y)] &= \sum_{i>0} Pr[preHash^1_{\text{key}}(X) = preHash^1_{\text{key}}(Y) = i] \\
&= \sum_{i>0} Pr[X_k = Y_k = 0]^{i-1} Pr[X_k = Y_k = 1] \\
&= \frac{Pr[X_k = Y_k = 1]}{1 - Pr[X_k = Y_k = 0]} \\
&= \frac{HW(X \wedge Y)}{HW(X) + HW(Y) - HW(X \wedge Y)} \\
&= \frac{1}{\frac{1}{Score(X,Y)} - 1}
\end{aligned}
\tag{3.1}
$$

This equation encapsulates the likelihood of two images, X and Y, having their singular hash index coincide, based on the presence of matching features identified by the algorithm. The final form of the equation relates the probability to the scoring function between X and Y, inversely proportional to the score minus one.

It is noticed that there is a direct link with the Miura matching score that is of interest. The direct link between the preHash algorithm's outcomes and the Miura matching score lies in their shared foundation of evaluating biometric similarities. Specifically, both methodologies utilize Hamming weight and bitwise operations to assess the overlap between biometric samples, such as finger vein patterns. The preHash algorithm, through its probabilistic formula, quantifies the likelihood of matching indices based on feature presence, closely paralleling the Miura score's approach of comparing binary patterns to derive a similarity score. The above computation can also be expressed as follows:

$$
\begin{aligned}
Pr[preHash^1_{\text{key}}(\bar{X}) = preHash^1_{\text{key}}(\bar{Y})] &= \frac{Pr[\bar{X}_k = \bar{Y}_k = 1]}{1 - Pr[\bar{X}_k = \bar{Y}_k = 0]} \\
&= \frac{\frac{Pr[X_k=1]+Pr[Y_k=1]}{2} - \frac{1}{2}Pr[\bar{X}_k \neq \bar{Y}_k]}{\frac{Pr[X_k=1]+Pr[Y_k=1]}{2} + \frac{1}{2}Pr[\bar{X}_k \neq \bar{Y}_k]}
\end{aligned}
\tag{3.2}
$$

The following approximations are made, inspired by equations p (2.1) and δ (2.6):

$$
\mathbb{E}\left(\frac{\frac{Pr[X_k=1]+Pr[Y_k=1]}{2} - \frac{1}{2}Pr[\bar{X}_k \neq \bar{Y}_k]}{\frac{Pr[X_k=1]+Pr[Y_k=1]}{2} + \frac{1}{2}Pr[\bar{X}_k \neq \bar{Y}_k]}\right) \approx \frac{p - \frac{\delta}{2}}{p + \frac{\delta}{2}} \tag{3.3}
$$

The core of this approximation revolves around the expectation formula, which integrates probabilities of feature presence $Pr[X_k = 1] + Pr[Y_k = 1]$ and the likelihood of discrepancies between $\bar{X}$ and $\bar{Y}$, $Pr[\bar{X}_k \neq \bar{Y}_k]$. This formula essentially aims to quantify the similarity between two biometric samples by considering both the concurrence of features and the instances where they diverge.

To achieve a more accurate approximation of the expected value of Equation 3.2, we define the following parameters and employ a Taylor series expansion of the function $f(A,B,C) = \frac{A+B-C}{A+B+C}$ around the point (p,p,δ). This method allows us to incorporate not only the means but also the variances and covariances of the variables involved.

$$A = \frac{HW(\bar{X})}{n}, \quad B = \frac{HW(\bar{Y})}{n}, \quad C = \frac{d_H(\bar{X},\bar{Y})}{n}$$

where $HW(\cdot)$ denotes the Hamming weight and $\frac{d_H(\cdot,\cdot)}{n}$ represents the normalized Hamming distance. Given these definitions, we know the expected values and variances from Section 2 :

$$\mathbb{E}(A) = \mathbb{E}(B) = p^{\text{obs}}, \quad \mathbb{E}(C) = \delta^{\text{obs}}, \quad \mathbb{V}(A) = \mathbb{V}(B) = (\sigma_{\text{p}}^{\text{obs}})^2, \quad \mathbb{V}(C) = (\sigma_{\delta}^{\text{obs}})^2$$

Next, we perform a Taylor series expansion of $f(A,B,C)$ up to the second order around the point (p,p,δ). The first-order partial derivatives are calculated as follows:

$$f'_A = \frac{2C}{(A+B+C)^2}, \quad f'_B = \frac{2C}{(A+B+C)^2}, \quad f'_C = \frac{-2(A+B)}{(A+B+C)^2}$$

The second-order partial derivatives are:

$$f''_{AA} = f''_{BB} = f''_{AB} = \frac{-4C}{(A+B+C)^3}, \quad f''_{CC} = \frac{4(A+B)}{(A+B+C)^3}, \quad f''_{AC} = f''_{BC} = \frac{-2(A+B-C)}{(A+B+C)^3}$$

By substituting $A = p$, $B = p$, and $C = \delta$ into these second order derivatives, we obtain:

$$f''_{AA} = f''_{BB} = f''_{AB} = \frac{-4\delta}{(2p+\delta)^3}, \quad f''_{CC} = \frac{8p}{(2p+\delta)^3}, \quad f''_{AC} = f''_{BC} = \frac{2(2p-\delta)}{(2p+\delta)^3}$$

Finally, the second order Taylor expansion of $f(A,B,C)$ around (p,p,δ) is given by:

$$f(A,B,C) \approx f(p,p,\delta) + f'_A(p,p,\delta)(A-p) + f'_B(p,p,\delta)(B-p) + f'_C(p,p,\delta)(C-\delta)$$

$$+\frac{1}{2}\left[f''_{AA}(p,p,\delta)(A-p)^2 + f''_{BB}(p,p,\delta)(B-p)^2 + f''_{CC}(p,p,\delta)(C-\delta)^2\right.$$

$$+2f''_{AB}(p,p,\delta)(A-p)(B-p)+2f''_{AC}(p,p,\delta)(A-p)(C-\delta)+2f''_{BC}(p,p,\delta)(B-p)(C-\delta)$$

To approximate the expected value $E(f(A,B,C))$, we use the following relations:

$$\mathbb{E}(A-p) = 0, \quad \mathbb{E}(B-p) = 0, \quad \mathbb{E}(C-\delta) = 0$$

$$\mathbb{V}(A) = \mathbb{V}(B) = \mathbb{E}((A-p)^2), \quad \mathbb{V}(C) = \mathbb{E}((C-\delta)^2)$$

$$\text{Cov}(A,B) = \mathbb{E}((A-p)(B-p)), \quad \text{Cov}(B,C) = \mathbb{E}((B-p)(C-\delta))$$

$$\mathrm{Cov}(A, C) = \mathbb{E}((A - p)(C - \delta))$$

And we hence have:

$$\mathbb{E}(f(A, B, C)) \approx \frac{p - \frac{\delta}{2}}{p + \frac{\delta}{2}} + \frac{1}{2}\left[f''_{AA}(p, p, \delta)\mathbb{V}(A) + f''_{BB}(p, p, \delta)\mathbb{V}(B) + f''_{CC}(p, p, \delta)\mathbb{V}(C)\right.$$

$$\left. + 2f''_{AB}(p, p, \delta)\mathrm{Cov}(A, B) + 2f''_{AC}(p, p, \delta)\mathrm{Cov}(A, C) + 2f''_{BC}(p, p, \delta)\mathrm{Cov}(B, C)\right]$$

Simplifying with our values and notations we get:

$$\begin{aligned}\mathbb{E}(f(A, B, C)) \approx \frac{p - \frac{\delta}{2}}{p + \frac{\delta}{2}} + \frac{1}{2}\Bigg[&\frac{-4\delta}{(2p + \delta)^3} \cdot (\sigma_p^{\mathrm{obs}})^2 + \frac{-4\delta}{(2p + \delta)^3} \cdot (\sigma_p^{\mathrm{obs}})^2 \\ &+ \frac{8p}{(2p + \delta)^3} \cdot (\sigma_\delta^{\mathrm{obs}})^2 + 2 \cdot \frac{-4\delta}{(2p + \delta)^3} \cdot \mathrm{Cov}(A, B) \\ &+ 2 \cdot \frac{2(2p - \delta)}{(2p + \delta)^3} \cdot \mathrm{Cov}(A, C) + 2 \cdot \frac{2(2p - \delta)}{(2p + \delta)^3} \cdot \mathrm{Cov}(B, C)\Bigg]\end{aligned} \tag{3.4}$$

Hence for (X, Y) random,

$$Pr[preHash^1_{\mathrm{key}}(offset_X * X) = preHash^1_{\mathrm{key}}(offset_Y * Y)] \leq \mathbb{E}(f(A, B, C)) \tag{3.5}$$

where equality is reached for the optimal offset translations.

Depending on the distribution of (X, Y), it is denoted

$$\mu = \mathbb{E}(f(A, B, C)) \tag{3.6}$$

In order to assess the theoretical approximations of μ_{same}, μ_{diff}, and the value of μ_{indep}, we can easily replace the different values in Equation 3.4 with the corresponding values calculated in the previous sections. We demonstrate how this is done for $\mu^{\mathrm{obs}}_{\mathrm{indep}}$. Using p^{obs}, $\delta^{\mathrm{obs}}_{\mathrm{indep}}$, and their respective standard deviations, we aim to calculate $\mathbb{E}(f(A, B, C))$. This calculation is based on X and Y comprising $2n$ independent random bits with an expected value of p, without applying the optimal offset to obtain $\bar{X}$ and $\bar{Y}$. Equation 3.4 is simplified because the covariance between A and B is zero (because of independence).

For clarity, note that in the following equations:

- p corresponds to p^{obs},
- δ corresponds to $\delta^{\mathrm{obs}}_{\mathrm{indep}}$,
- σ_{p} corresponds to $\sigma^{\mathrm{obs}}_{\mathrm{p}}$, and
- σ_δ corresponds to $\sigma^{\mathrm{obs}}_{\delta_{\mathrm{indep}}}$.

We have the following equalities:

$$\frac{p-\frac{\delta}{2}}{p+\frac{\delta}{2}} \approx 1.78 \times 10^{-2}, \quad \frac{-4\delta}{(2p+\delta)^3} \cdot (\sigma_{\mathrm{p}}^{\mathrm{obs}})^2 \approx -2.6 \times 10^{-3}$$

$$\frac{8p}{(2p+\delta)^3} \cdot (\sigma_{\delta}^{\mathrm{obs}})^2 \approx 1.5 \times 10^{-4}, \quad 2 \cdot \frac{2(2p-\delta)}{(2p+\delta)^3} \cdot \mathrm{Cov}(A,C) = 2 \cdot \frac{2(2p-\delta)}{(2p+\delta)^3} \cdot \mathrm{Cov}(B,C) \approx 1.31 \times 10^{-6}$$

This allows us to express $\mu_{\mathrm{indep}}^{obs}$ as follows:

$$\begin{aligned}
\mu_{\mathrm{indep}}^{\mathrm{obs}} = \mathbb{E}(f(A,B,C)) \approx \quad & \frac{p-\frac{\delta}{2}}{p+\frac{\delta}{2}} + \frac{1}{2}\left[\frac{-4\delta}{(2p+\delta)^3} \cdot (\sigma_{\mathrm{p}}^{\mathrm{obs}})^2 + \frac{-4\delta}{(2p+\delta)^3} \cdot (\sigma_{\mathrm{p}}^{\mathrm{obs}})^2\right. \\
& + \frac{8p}{(2p+\delta)^3} \cdot (\sigma_{\delta}^{\mathrm{obs}})^2 + 2 \cdot \frac{2(2p-\delta)}{(2p+\delta)^3} \cdot \mathrm{Cov}(A,C) \\
& \left. + 2 \cdot \frac{2(2p-\delta)}{(2p+\delta)^3} \cdot \mathrm{Cov}(B,C)\right] \\
& = 1.78 \times 10^{-2} + \frac{1}{2}\left[-2 \cdot 2.6 \times 10^{-3} + 1.5 \times 10^{-4}\right. \\
& \left. + 2 \cdot 1.31 \times 10^{-6}\right] \\
& \approx 0.01536 = 1.54\%
\end{aligned} \tag{3.7}$$

We proceed in a similar way to assess the theoretical values of μ_{same} and μ_{diff}, except in this case the covariances are non-zero and have been calculated by observing the correlation between the different random variables (A, B, C) in the experimental results obtained in Section 2. That is, we calculated the covariance matrix between the following random variables:

$$A = \frac{HW(\bar{X})}{n}, \quad B = \frac{HW(\bar{Y})}{n}, \quad C = \frac{d_H(\bar{X},\bar{Y})}{n}$$

We obtained the following covariance matrices from the experiments held to find $\delta_{\mathrm{diff}}^{\mathrm{obs}}$ and $\delta_{\mathrm{same}}^{\mathrm{obs}}$:

	A	B	C
A	0.000025	0.000022	0.000036
B	0.000022	0.000025	0.000036
C	0.000036	0.000036	0.000106

Covariance Matrix for Same Distribution

	A	B	C
A	0.000025	0.000006	0.000026
B	0.000006	0.000026	0.000027
C	0.000026	0.000027	0.000046

Covariance Matrix for Different Distribution

We have the following equations for μ_{same} and μ_{diff}. Depending on the distribution (same or different), the variables p, δ, σ_{p}, and σ_δ correspond to their observed counterparts as indicated:

- $p \to p^{\text{obs}}$
- $\delta \to \delta^{\text{obs}}_{\text{same/diff}}$
- $\sigma_p \to \sigma^{\text{obs}}_{\text{p}}$
- $\sigma_\delta \to \sigma^{\text{obs}}_{\delta_{\text{same/diff}}}$

Moreover, we have the following equalities for $\mu^{\text{obs}}_{\text{same}}$:

$$\frac{p - \frac{\delta}{2}}{p + \frac{\delta}{2}} \approx 2.14 \times 10^{-1}, \quad \frac{-4\delta}{(2p+\delta)^3} \cdot (\sigma^{\text{obs}}_{\text{p}})^2 \approx -2.9 \times 10^{-3}$$

$$\frac{8p}{(2p+\delta)^3} \cdot (\sigma^{\text{obs}}_{\delta})^2 \approx 1.9 \times 10^{-2}, \quad 2 \cdot \frac{-4\delta}{(2p+\delta)^3} \cdot \text{Cov}(A, B) \approx -5.1 \times 10^{-3}$$

$$2 \cdot \frac{2(2p-\delta)}{(2p+\delta)^3} \cdot \text{Cov}(A, C) = 2 \cdot \frac{2(2p-\delta)}{(2p+\delta)^3} \cdot \text{Cov}(B, C) \approx 2.3 \times 10^{-3}$$

This allows us to express $\mu^{\text{obs}}_{\text{same}}$ as follows:

$$\begin{aligned}\mu^{\text{obs}}_{\text{same}} = \mathbb{E}(f(A,B,C)) \approx \quad & \frac{p-\frac{\delta}{2}}{p+\frac{\delta}{2}} + \frac{1}{2}\left[\frac{-4\delta}{(2p+\delta)^3}\cdot(\sigma^{\text{obs}}_{\text{p}})^2 + \frac{-4\delta}{(2p+\delta)^3}\cdot(\sigma^{\text{obs}}_{\text{p}})^2\right. \\ & + \frac{8p}{(2p+\delta)^3}\cdot(\sigma^{\text{obs}}_{\delta})^2 + 2\cdot\frac{-4\delta}{(2p+\delta)^3}\cdot\text{Cov}(A,B) \\ & \left. + 2\cdot\frac{2(2p-\delta)}{(2p+\delta)^3}\cdot\text{Cov}(A,C) + 2\cdot\frac{2(2p-\delta)}{(2p+\delta)^3}\cdot\text{Cov}(B,C)\right] \\ & = 2.14\times10^{-1} + \frac{1}{2}\left[-2\cdot2.9\times10^{-3} + 1.9\times10^{-2}\right. \\ & \left. -5.1\times10^{-3} + 2\cdot2.3\times10^{-3}\right] \\ & \approx 0.22048 = 22.05\% \end{aligned} \tag{3.8}$$

We have the following equalities for μ^{obs}_{diff} :

$$\frac{p-\frac{\delta}{2}}{p+\frac{\delta}{2}} \approx 7.99\times10^{-2}, \quad \frac{-4\delta}{(2p+\delta)^3}\cdot(\sigma^{\text{obs}}_{\text{p}})^2 \approx -2.7\times10^{-3}$$

$$\frac{8p}{(2p+\delta)^3}\cdot(\sigma^{\text{obs}}_{\delta})^2 \approx 5.8\times10^{-3}, \quad 2\cdot\frac{-4\delta}{(2p+\delta)^3}\cdot\text{Cov}(A,B) \approx -1.3\times10^{-3}$$

$$2\cdot\frac{2(2p-\delta)}{(2p+\delta)^3}\cdot\text{Cov}(A,C) \approx 4.9\times10^{-4}, \quad 2\cdot\frac{2(2p-\delta)}{(2p+\delta)^3}\cdot\text{Cov}(B,C) \approx 2.3\times10^{-3}$$

This allows us to express $\mu^{\text{obs}}_{\text{diff}}$ as follows:

$$\begin{aligned}\mu_{\text{diff}}^{\text{obs}} = \mathbb{E}(f(A,B,C)) \approx \quad & \frac{p-\frac{\delta}{2}}{p+\frac{\delta}{2}} + \frac{1}{2}\left[\frac{-4\delta}{(2p+\delta)^3}\cdot(\sigma_{\text{p}}^{\text{obs}})^2 + \frac{-4\delta}{(2p+\delta)^3}\cdot(\sigma_{\text{p}}^{\text{obs}})^2\right. \\ & + \frac{8p}{(2p+\delta)^3}\cdot(\sigma_{\delta}^{\text{obs}})^2 + 2\cdot\frac{-4\delta}{(2p+\delta)^3}\cdot\text{Cov}(A,B) \\ & \left. + 2\cdot\frac{2(2p-\delta)}{(2p+\delta)^3}\cdot\text{Cov}(A,C) + 2\cdot\frac{2(2p-\delta)}{(2p+\delta)^3}\cdot\text{Cov}(B,C)\right] \\ & = 7.99\times10^{-2} + \frac{1}{2}\left[-2\cdot2.7\times10^{-3} + 5.8\times10^{-3}\right. \\ & \left. -1.3\times10^{-3} + 4.9\times10^{-4} + 5\times10^{-4}\right] \\ & \approx 0.07993 = 7.99\% \end{aligned} \tag{3.9}$$

Finally, the following figures are provided:

μ_{same}	μ_{diff}	μ_{indep}
21.41%	7.99%	1.78%

Table 7: Theoretical Approximations without Taylor Series Expansion: μ_{same}, μ_{diff}, and μ_{indep}

μ_{same}	μ_{diff}	μ_{indep}
22.05%	7.99%	1.54%

Table 8: Theoretical Approximations with Taylor Series Expansion: μ_{same}, μ_{diff}, and μ_{indep}

We can note that there is not a major difference between approximations from Table 8 and Table 7. Yet, the approximation from Table 8 yields extremely close values to the experimental results from subsection 3.3.

Finally, it is observed that:

$$Pr[preHash_{\text{key}}^{\text{m}}(offset_X * X) = preHash_{\text{key}}^{\text{m}}(offset_Y * Y)] \leq \mu^{\text{m}} \tag{3.10}$$

where equality is reached for the optimal offset translations.

3.3 Experimental Derivation of the Probabilities $\mu^{\mathbf{obs}}_{\mathbf{same}}, \mu^{\mathbf{obs}}_{\mathbf{diff}}, \mu^{\mathbf{obs}}_{\mathbf{indep}}$

In section 3.2, we have laid out the mathematical framework that defines the upper bound μ. This probability is crucial in our fuzzy hashing approach, as it quantifies the matching likelihood after applying the optimal offset translations to the biometric captures. The application of these optimal offsets is an important preprocessing step before hashing the dataset, ensuring that the biometric features are accurately aligned for comparison.

We will proceed to evaluate these theoretical approximations (Table 8) experimentally, leveraging the same comprehensive database that has been utilized in our prior research, as explained in Section 2.3. This experimental assessment will enable us to substantiate the theoretical predictions of μ_{same} and μ_{diff}.

In the experimental setup, $\mu^{\text{obs}}_{\text{same}}$ refers to the probability that a hash function, when applied to two biometric captures of the same finger, will yield the same index. This is the measure of success in correctly identifying matches within the same individual's biometric captures. Conversely, $\mu^{\text{obs}}_{\text{diff}}$ denotes the probability that the same hash function will produce different indices for biometric captures from different individuals, representing the ability to distinguish between different people's biometric data. Lastly, $\mu^{\text{obs}}_{\text{indep}}$ indicates the probability of a match in the case where the biometric captures are completely independent, serving as a baseline for random chance, which has already been calculated (see Equation 3.7)

For $\mu^{\text{obs}}_{\text{same}}$ and $\mu^{\text{obs}}_{\text{diff}}$, our experimental protocol adhered to the established optimal pipeline (refer to Figure 1.1) to process the images in our dataset. Following the alignment of images through the computation of optimal offsets (referred to as Miura Matching), our fuzzy hashing function was applied to the extraced features. In alignment with the premise that μ is computed under the condition where the hash function's output is a single index ($m = 1$).

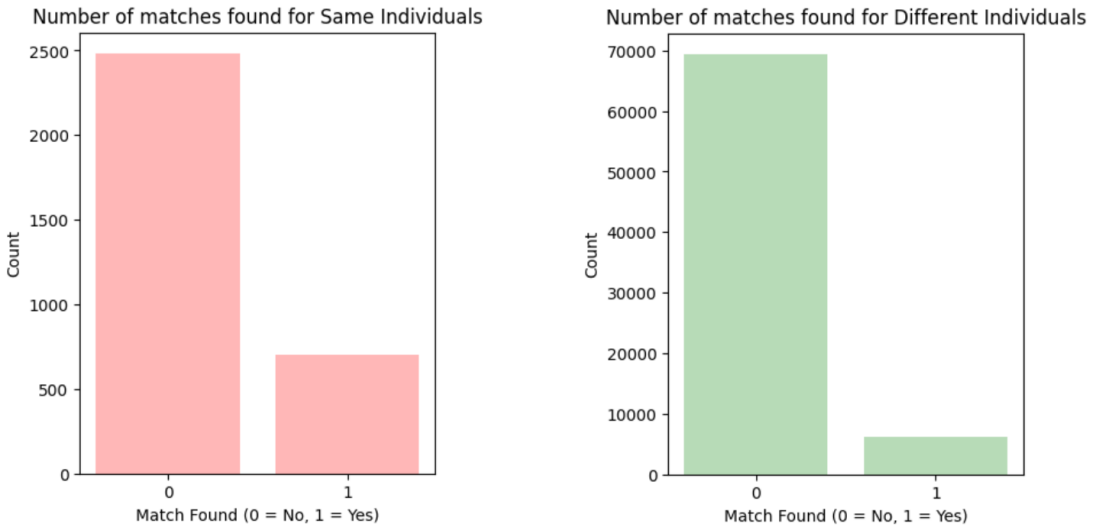
1. **Pairwise Comparison**: We performed a pairwise comparison for each image pair within the dataset after applying the optimal offsets to ensure proper alignment. The fuzzy hashing function, tailored to output a single index per image, was utilized to hash the images.
2. **Hamming Distance Computation**: We calculated the Hamming distance between the indices obtained from the fuzzy hashing. The distances were binary:
    - A Hamming distance of 0 indicates the indices are identical, suggesting a match.
    - A Hamming distance of 1 indicates differing indices, suggesting a non-match.
3. **Statistical Analysis and Probability Computation**:
    - $\mu^{\mathbf{obs}}_{\mathbf{same}}$: We computed $\mu^{\text{obs}}_{\text{same}}$ by averaging the number of Hamming distances that resulted in a match (Hamming distances of 0) from the distribution of same biometric subject comparisons, which furnished us with

the following probability:

$$\mu_{\text{same}}^{obs} \approx 0.22047 = 22.05\%$$

- $\mu_{\textbf{diff}}^{\textbf{obs}}$: Similarly, $\mu_{\text{diff}}^{\text{obs}}$ was determined by averaging the number of Hamming distances that resulted in a match (Hamming distances of 0) from the distribution of different biometric subject comparisons, which furnished us with the following probability:

$$\mu_{\text{diff}}^{\text{obs}} \approx 0.08249 = 8.25\%$$



(a) Count of the number of matches for Same, Aligned Biometric Samples with Single Index preHashing

(b) Count of the number of matches for Different, Aligned Biometric Samples with Single Index preHashing

Figure 3.2: Comparison of match counts for Same and Different Aligned Biometric Samples

In our analysis of the dataset at hand featuring biometric data from 20 unique individuals, we have quantified the performance of the preHash function in distinguishing between identical and different biometric subjects. Specifically, when evaluating samples from the same individual, the probability that the preHash function will produce the same index is approximately 22.05%. Conversely, when the function is applied to samples from different individuals, the probability of a match in the indices drops to roughly 8.25%. This indicates that the likelihood of mistakenly identifying different individuals as the same is about three times lower than correctly matching samples from the same individual, demonstrating the preHash function's relative efficacy in biometric differentiation within this specific dataset. Moreover, we can observe that the experimental results that we got are very close to the theoretical values predicted from Table 8. This shows that our theoretical framework and the Taylor series approximations accurately model the behavior of

μ_{diff} and μ_{same}. The minor differences between the theoretical model and the experiments can be explained by the fact that real-world biometric data can include noise and measurement errors that are not accounted for in the theoretical model.

3.3.1 Experimental Derivation of the Probabilities μ_{same} and μ_{diff} Excluding the Post-Alignment Step

We conducted an experimental investigation to determine the impact of omitting the Miura Marching step, which optimizes offsets before comparing images, on the coincidence probabilities of the indices. The Miura Matching step serves as a post-alignment phase in the current pipeline. According to Equation 3.5, excluding the Miura Matching phase should result in the probabilities of matching indices being less than or equal to the previous values determined for μ_{same} and μ_{diff}, respectively. Following the setup and conducting the experiment as described in the previous section, we obtained the following statistical analysis and probability computations:

- $\mu_{\text{same}}^{\text{obs}}$:

$$\mu_{\text{same}}^{\text{obs}} \approx 0.11871 = 11.87\%$$

- $\mu_{\text{diff}}^{\text{obs}}$:

$$\mu_{\text{diff}}^{\text{obs}} \approx 0.05651 = 5.65\%$$

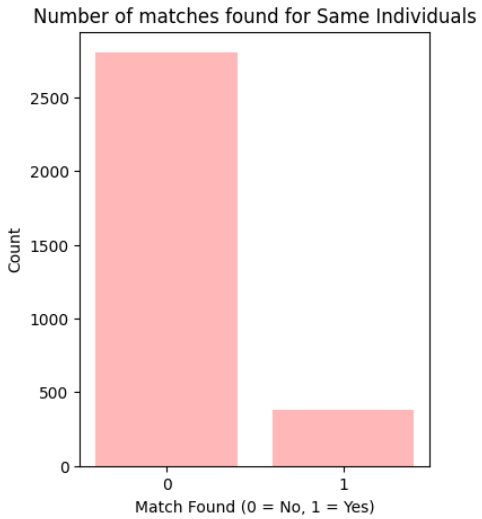


Figure 3.3: Count of the number of matches for Same, Non-Aligned Biometric Samples with Single Index preHashing

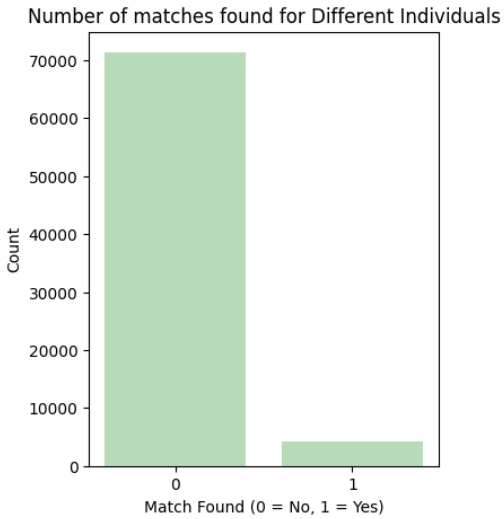


Figure 3.4: Count of the number of matches for Different, Non-Aligned Biometric Samples with Single Index pre-Hashing

Figure 3.5: Comparison of match counts for Same and Different Non-Aligned Biometric Samples

As expected, omitting the Miura Matching step resulted in $\mu_{\text{same}} = 11.87\%$ and $\mu_{\text{diff}} = 5.65\%$, which are indeed less than the previously determined values of $\mu_{\text{same}} = 22.05\%$ and $\mu_{\text{diff}} = 8.25\%$.

4 Compressed Fuzzy Hashing

Compressed fuzzy hashing generates concise hashes of data, capturing key features while allowing for slight variations - a concept referred to as "fuzziness". Unlike conventional cryptographic hashing, which generates fixed-length outputs and is highly sensitive to even minimal changes in input, compressed fuzzy hashing maintains a robust link to the original content despite alterations. This section delves into the mechanics of the compressed fuzzy hashing algorithm, called the postHash procedure. We will explore the mathematical concepts that enable its effectiveness and discuss experimental results that demonstrate its capabilities and performance on our dataset.

4.1 PostHashing Algorithm

The postHash algorithm, constituting the second and final step in the fuzzy hashing process, generates compact hashes based on the indices derived from the preHash algorithm (see Section 3). This algorithm is designed to transform a tuple of indices into a concise hash format, $h_1||\ldots||h_m$, where each h_i, for $i \in [1, m]$, is an integer within the range $[0, \ldots, D-1]$.The parameter D, defined as 2^d, denotes the total number of possible values each h_i can assume, with d representing the bit depth used per index.

Algorithm Inputs and Outputs:

1. **Inputs**: As illustrated in figure 2, the algorithm takes as input:
   - **A tuple of indices**: $(i_1, \ldots, i_m)$, which is the output of the preHash algorithm
2. **Output**: Converted indices, forming a compact hash $h_1||\ldots||h_m$, where h_i $i \in [1, m]$, is an integer in $[0, \ldots, D-1]$

Detailed process of postHash, for the case when $d < 3$ (refer to figures 2 and 3):

1. **Subroutine**: For each index provided by the preHash, subroutine T checks if the index is within the acceptable range. If an index is out of range, T returns 0, otherwise, it proceeds with a table lookup. The subroutine maps each valid index i to an integer nearly uniformly distributed between $[0, D-1]$, ensuring that all potential hash values are equally probable. This mapping is important for the security and effectiveness of the fuzzy hashing system.

   The subroutine function, T, works as follows:

   - **Inputs**:
     - **Shifted Index**: Each index from the preHash output tuple $(i_1, \ldots, i_m)$ is adjusted before conversion into a hash component. Specifically, the index i is shifted by subtracting a previously determined index i', resulting in a transformed index $i - i'$. This shifting process normalizes the indices, maintaining a consistent relative positioning within

the dataset, which helps in achieving a more uniform distribution of hash values across the range $[0, D-1]$.

- **d**: Represents the bit length used for each hash index h_i. This value determines $D = 2^d$, the total number of distinct values each h_i can take, effectively setting the hash resolution.

- **Output**: h_i, the mapped integer

For the case when $d = 3$ and $d = 4$, a different subroutine called *srcambled_T* is used. This subroutine corresponds to a partition problem, which is detailed at the end of Section 4.3. Although the mapping of integers differs in this subroutine, it similarly takes an index from the preHash tuple as input and maps it to an integer, just as in the process for $d < 3$.

Algorithm 2 postHash Algorithm

```
function (postHash ∘ preHash^m_KEY)(X)
    hash = []
    i' ← 0
    for i ∈ indices do
        h_i ← Subroutine(i − i', d)
        hash.append(h_i)
        i' ← i
    end for
    return h_1, ..., h_m
end function
```

Algorithm 3 *Subroutine* Algorithm

```
function Subroutine(i, d)
    p = 0.03514
    h_i = ⌊2^d (1 − p)^i⌋
    return h_i
end function
```

4.2 Assessing Similarity of Biometric Inputs After PostHash Application

After processing finger images through the pipeline referenced in Pipeline 1.1 to extract their feature vectors, the resulting data undergo the preHash and subsequently, the postHash algorithms. The final output from postHash consists of a set of integers h_i, each bounded within the inclusive range $[0, D-1]$. This process effectively assigns each feature vector index to an integer within this range.

In our methodology, we model these integers h_i as following a geometric distribution when X and key are random. This assumption is based on the output relationship

established by $Hash^m_{\text{key}}(X) = postHash(preHash^m_{\text{key}}(X))$. The probability of the Hash operation yielding the same index for two different aligned inputs $d_X \cdot X$ and $d_Y \cdot Y$ can be mathematically characterized as follows:

$$Pr[Hash^m_{\text{key}}(d_X \cdot X) = Hash^m_{\text{key}}(d_Y \cdot Y)] \leq \mu^m(1 - \frac{1}{D}) + \frac{1}{D}$$

where equality is reached for the optimal offset translations.

The overall probability q of a matching hash index under random inputs X and Y, reflecting their distribution, is denoted by:

$$q = \mu^m \left(1 - \frac{1}{D}\right) + \frac{1}{D} \tag{4.1}$$

Based on the range of possible values for d, we have computed several specific instances of q (theoretical values given by the above equation):

	q_{same}	q_{diff}	q_{indep}
$m = 1$ $d = 1$	61.03%	54.12%	50.73%
$m = 1$ $d = 2$	41.54%	31.19%%	26.09%
$m = 1$ $d = 3$	31.79%	19.72%	13.77%
$m = 1$ $d = 4$	26.92%	13.98%	7.61%

Table 9: Comparison of Distributions: q_{same}, q_{diff}, and q_{indep}

4.3 Experimental Derivation of the Probabilities $q^{\text{obs}}_{\text{same}}$, $q^{\text{obs}}_{\text{diff}}$, and $q^{\text{obs}}_{\text{indep}}$ for Different m, d Parameter Combinations

In the subsequent section, we explore enhancements to data compression techniques within our fuzzy hashing framework, emphasizing the use of various m and d parameter combinations. We initiate our analysis by conducting experiments to determine $q^{\text{obs}}_{\text{same}}$, $q^{\text{obs}}_{\text{diff}}$, and $q^{\text{obs}}_{\text{indep}}$.

Building on the foundational calculations for q_{indep}, which we derived using Formula 4.1 with $\mu^{\text{obs}}_{\text{indep}}$ previously identified in Section 3, our exploration extended to $q^{\text{obs}}_{\text{same}}$ and $q^{\text{obs}}_{\text{diff}}$. To evaluate these parameters, we initiated a series of experiments. Integral to this implementation and the following ones, is the creation of the lookup table, to convert indices to their compressed counterparts. The generation of this table and all subsequent ones, is governed by a geometrically-derived formula that accurately encapsulates the transformation of index values. The specific formula is presented as:

$$\text{postHash}(i) = \left\lfloor 2^d \left(1 - p\right)^i \right\rfloor \tag{4.2}$$

Leveraging the above formula, we have developed lookup tables that convert a single index ($m = 1$) into various bit representations ($d = 1$, $d = 2$, $d = 3$, and $d = 4$). Each entry in these tables represents an integer, denoted by i, and includes a corresponding hash output and the probability of that output occurring.

Binary Hash Output Distribution Table ($d = 1$)

$$\text{postHash}(i) = \begin{cases} 1 & \text{if } 1 \leq i \leq 19 \quad (\Pr = 49.32\%) \\ 0 & \text{if } 20 \leq i \quad (\Pr = 50.68\%) \end{cases}$$

Quaternary Hash Output Distribution Table ($d = 2$):

$$\text{PostHash}(i) = \begin{cases} 3 & \text{if } 1 \leq i \leq 8 \quad (\Pr = 24.89\%) \\ 2 & \text{if } 9 \leq i \leq 19 \quad (\Pr = 24.43\%) \\ 1 & \text{if } 20 \leq i \leq 38 \quad (\Pr = 25\%) \\ 0 & \text{if } 39 \leq i \quad (\Pr = 25.68\%) \end{cases}$$

Octal Hash Output Distribution Table ($d = 3$):

$$\text{postHash}(i) = \begin{cases} 7 & \text{if } 1 \leq i \leq 3 \quad (\Pr = 10.18\%) \\ 6 & \text{if } 4 \leq i \leq 8 \quad (\Pr = 14.71\%) \\ 5 & \text{if } 9 \leq i \leq 13 \quad (\Pr = 12.3\%) \\ 4 & \text{if } 14 \leq i \leq 19 \quad (\Pr = 12.13\%) \\ 3 & \text{if } 20 \leq i \leq 27 \quad (\Pr = 12.61\%) \\ 2 & \text{if } 28 \leq i \leq 38 \quad (\Pr = 12.38\%) \\ 1 & \text{if } 39 \leq i \leq 58 \quad (\Pr = 13.12\%) \\ 0 & \text{if } 59 \leq i \quad (\Pr = 12.59\%) \end{cases}$$

Hexadecimal Hash Output Distribution Table ($d = 4$):

$$\text{postHash}(i) = \begin{cases} 15 & \text{if } i = 1 & (\Pr = 3.51\%), \\ 14 & \text{if } 2 \leq i \leq 3 & (\Pr = 6.66\%) \\ 13 & \text{if } 4 \leq i \leq 5 & (\Pr = 6.2\%) \\ 12 & \text{if } 6 \leq i \leq 8 & (\Pr = 8.51\%) \\ 11 & \text{if } 9 \leq i \leq 10 & (\Pr = 5.19\%) \\ 10 & \text{if } 11 \leq i \leq 13 & (\Pr = 7.12\%) \\ 9 & \text{if } 14 \leq i \leq 16 & (\Pr = 6.39\%) \\ 8 & \text{if } 17 \leq i \leq 19 & (\Pr = 5.74\%) \\ 7 & \text{if } 20 \leq i \leq 23 & (\Pr = 6.76\%) \\ 6 & \text{if } 24 \leq i \leq 27 & (\Pr = 5.86\%) \\ 5 & \text{if } 28 \leq i \leq 32 & (\Pr = 6.23\%) \\ 4 & \text{if } 33 \leq i \leq 38 & (\Pr = 6.15\%) \\ 3 & \text{if } 39 \leq i \leq 46 & (\Pr = 6.39\%) \\ 2 & \text{if } 47 \leq i \leq 58 & (\Pr = 6.73\%) \\ 1 & \text{if } 59 \leq i \leq 77 & (\Pr = 6.19\%) \\ 0 & \text{if } 78 \leq i \leq n & (\Pr = 6.36\%) \end{cases}$$

In our current implementation of postHash output tables, each partition, notably partitions for $d = 3$ and $d = 4$, does not distribute probabilities as equally as we would desire among the bins. This uneven distribution, observed from binary to hexadecimal outputs, poses some challenges, particularly in our system, where hash functions are employed to ensure the confidentiality of biometric data. The non-uniform distribution can lead to predictable patterns, potentially compromising the security by making the system more vulnerable to attacks such as hash collisions. These predictable patterns can be exploited by attackers to reverse-engineer or guess hash values, thereby breaching the confidentiality of biometric information and undermining the integrity of our security measures.

In the context of compressed fuzzy hashing, domain scrambling aims to reorder the indices in a way that the resulting hash values are "nearly" uniformly distributed. This task shares similarities with a well known problem in computer science called the Partition Problem[3]. This problem involves deciding whether a given multiset of positive integers can be partitioned into two or more subsets such that the sum of the numbers in each subset is equal. This problem is a specific instance of the subset sum problem, where the target sum is half of the total sum of all elements in the set. The partition problem is classified as NP-complete, a categorization in computational complexity theory used to describe decision problems. A problem is in NP (Non-deterministic Polynomial time) if a solution for the problem can be verified in polynomial time given a candidate solution. moreover, a problem is NP-complete if it is as hard as the hardest problems in NP and if every problem in NP can be reduced to it in polynomial time. This implies two things for NP-complete problems:

1. Verification is Feasible: Given a solution, we can verify whether it is correct in polynomial time.
2. There is no known algorithm that can find an optimal solution to NP-complete problems in polynomial time for all general cases.

Given the NP-completeness of the partition problem, programming an optimal solution for domain scrambling — which is akin to a continuous partitioning challenge — is impractical. Theoretically, while it is feasible to verify if a given distribution of indices across hash buckets is optimal, finding that distribution algorithmically cannot be guaranteed to be efficient for all possible cases. However, we did try to implement this algorithm and observe the results for the cases where the hash depth is $d = 3$ and $d = 4$, resulting in $D = 9$ and $D = 16$ possible hash values respectively. The program written is designed to distribute a sequence of integers (1 to 90′240) across predefined buckets, (0 to 8) and (0 to 15) respectively, to achieve a target probability distribution based on our geometric decay model 4.2. This process begins by mapping each integer to a bucket using a calculated index derived from the geometric formula, adjusted by a scaling factor. Initial bucket assignments aim to approximate a uniform distribution of probabilities. The program then iteratively adjusts these assignments: integers are moved between buckets to better align with the target probabilities, ensuring that the sum of probabilities within each bucket remains close to the desired uniform distribution. This adjustment process continues until the distribution of probabilities across all buckets falls within an acceptable error margin, or until no further beneficial adjustments can be made. The following lookup tables illustrate the distribution of indices across each bucket, including the count of indices falling within each bucket and the range of indices encompassed by it.

$$\text{Bucket Assignments (d = 3)} = \begin{cases} 0 & \text{if } i \in [1, 90240] \quad (\text{Count} = 84435, \Pr = 12.548\%), \\ 1 & \text{if } i \in [2, 5812] \quad (\text{Count} = 1944, \Pr = 12.530\%), \\ 2 & \text{if } i \in [3, 3875] \quad (\text{Count} = 1142, \Pr = 12.540\%), \\ 3 & \text{if } i \in [4, 2741] \quad (\text{Count} = 812, \Pr = 12.486\%), \\ 4 & \text{if } i \in [5, 1937] \quad (\text{Count} = 633, \Pr = 12.535\%), \\ 5 & \text{if } i \in [6, 1313] \quad (\text{Count} = 519, \Pr = 12.546\%), \\ 6 & \text{if } i \in [7, 804] \quad (\text{Count} = 442, \Pr = 12.312\%), \\ 7 & \text{if } i \in [57, 373] \quad (\text{Count} = 313, \Pr = 12.502\%), \end{cases}$$

$$\text{Bucket Assignments (d = 4)} = \begin{cases} 0 & \text{if } i \in [1, 90240] \quad (\text{Count} = 82493, \Pr = 6.297\%), \\ 1 & \text{if } i \in [2, 7750] \quad (\text{Count} = 1941, \Pr = 6.276\%), \\ 2 & \text{if } i \in [3, 5812] \quad (\text{Count} = 1136, \Pr = 6.264\%), \\ 3 & \text{if } i \in [4, 4679] \quad (\text{Count} = 807, \Pr = 6.204\%), \\ 4 & \text{if } i \in [5, 3875] \quad (\text{Count} = 628, \Pr = 6.294\%), \\ 5 & \text{if } i \in [6, 3251] \quad (\text{Count} = 514, \Pr = 6.284\%), \\ 6 & \text{if } i \in [7, 2741] \quad (\text{Count} = 435, \Pr = 6.237\%), \\ 7 & \text{if } i \in [8, 2310] \quad (\text{Count} = 378, \Pr = 6.295\%), \\ 8 & \text{if } i \in [9, 1936] \quad (\text{Count} = 334, \Pr = 6.298\%), \\ 9 & \text{if } i \in [10, 1608] \quad (\text{Count} = 301, \Pr = 6.292\%), \\ 10 & \text{if } i \in [11, 1313] \quad (\text{Count} = 272, \Pr = 6.214\%), \\ 11 & \text{if } i \in [12, 1047] \quad (\text{Count} = 250, \Pr = 6.293\%), \\ 12 & \text{if } i \in [13, 804] \quad (\text{Count} = 232, \Pr = 6.298\%), \\ 13 & \text{if } i \in [14, 580] \quad (\text{Count} = 215, \Pr = 6.127\%), \\ 14 & \text{if } i \in [15, 373] \quad (\text{Count} = 201, \Pr = 6.061\%), \\ 15 & \text{if } i \in [76, 180] \quad (\text{Count} = 103, \Pr = 6.267\%) \end{cases}$$

We can see that the distribution of probabilities and counts across buckets demonstrates a good effort towards achieving uniformity.

Utilizing these derived lookup tables, alongside our established preHash function 1 and the newly implemented postHash function 2, we are equipped to conduct experiments on our dataset to assess q_{same} and q_{diff} for the 4 different combinations of the parameters d and m. The modification in our pipeline involves passing the output from the preHash function directly into the postHash function. Subsequently, we calculate the Hamming distance between each image pair at the conclusion of both the preHash and postHash stages. This method allows us to measure the similarity of the biometric inputs after the complete application of our hashing process.

Note that for $d = 3$ and $d = 4$, we used the lookup tables with redistributed indices defined just above. The initial bucket assignments, calculated using formula 4.2, did not achieve a balanced enough distribution of probabilities across the buckets, necessitating this redistribution.

1. $d = 1$ and $m = 1$: Single-index outputs ($m = 1$) and binary postHash indices ($d = 1$).
   - $q_{\text{same}}^{\text{obs}} = 61.53\%$
   - $q_{\text{diff}}^{\text{obs}} = 52.69\%$

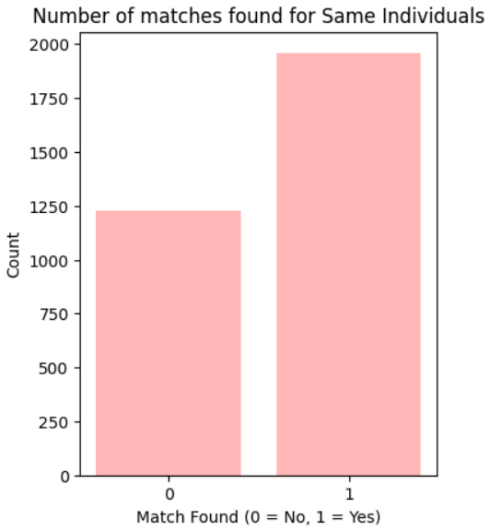


Figure 4.1: Count of the number of matches for Same, Aligned Biometric Samples with Single Index Hashing and Binary PostHash Indices

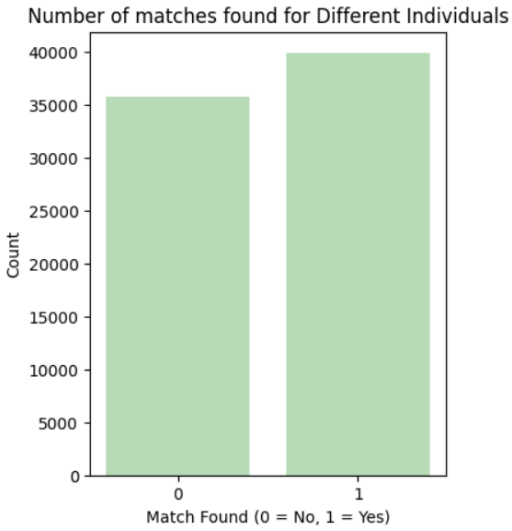


Figure 4.2: Count of the number of matches for Different, Aligned Biometric Samples with Single Index Hashing and Binary PostHash Indices

2. $d = 2$ and $m = 1$: Single-index outputs ($m = 1$) and quaternary postHash indices ($d = 2$).
   - $q_{\text{same}}^{\text{obs}} = 42.62\%$
   - $q_{\text{diff}}^{\text{obs}} = 29.84\%$

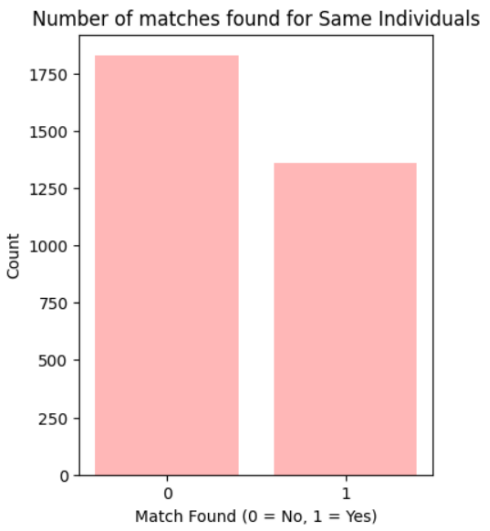


Figure 4.3: Count of the number of matches for Same, Aligned Biometric Samples with Single Index Hashing and Quaternary PostHash Indices

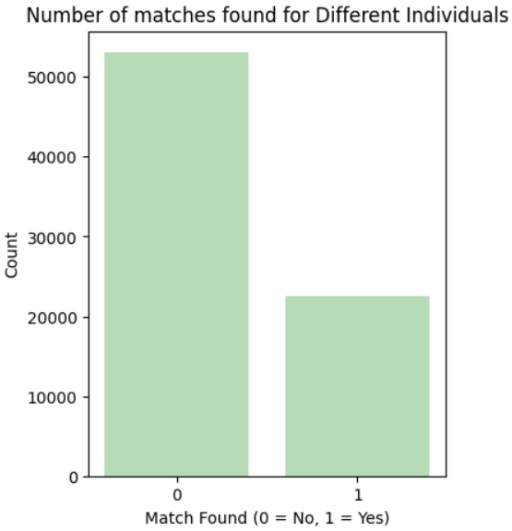


Figure 4.4: Count of the number of matches for Different, Aligned Biometric Samples with Single Index Hashing and Quaternary PostHash Indices

3. $d = 3$ and $m = 1$: Single-index outputs ($m = 1$) and octal postHash indices

($d = 3$).

- $q_{\text{same}}^{\text{obs}} = 32.57\%$
- $q_{\text{diff}}^{\text{obs}} = 19.04\%$

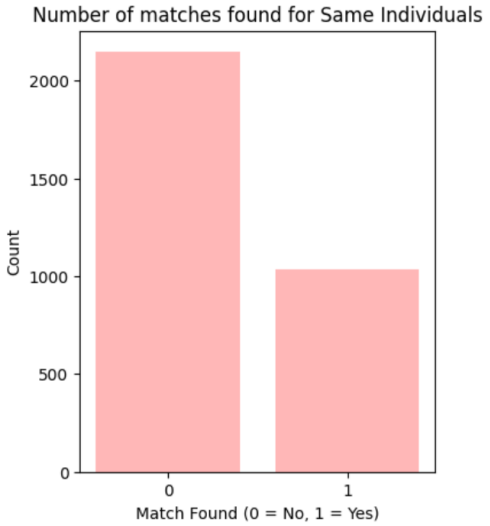


Figure 4.5: Count of the number of matches for Same, Aligned Biometric Samples with Single Index Hashing and Octal PostHash Indices

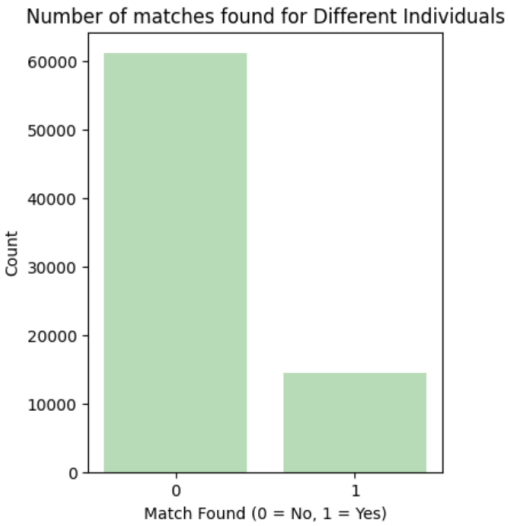


Figure 4.6: Count of the number of matches for Different, Aligned Biometric Samples with Single Index Hashing and Octal PostHash Indices

4. $d = 4$ and $m = 1$: Single-index outputs ($m = 1$) and hexa-decimal postHash indices ($d = 4$).
   - $q_{\text{same}}^{\text{obs}} = 26.92\%$
   - $q_{\text{diff}}^{\text{obs}} = 13.13\%$

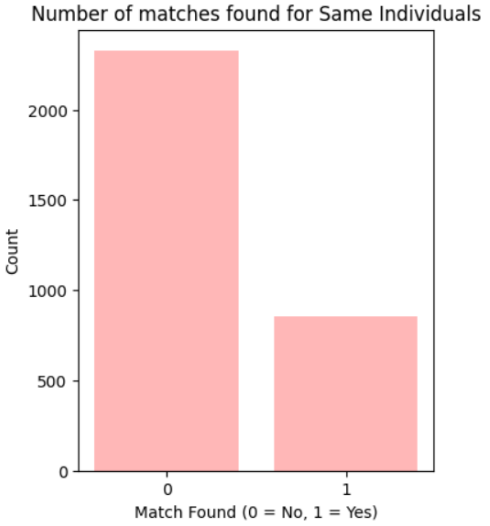


Figure 4.7: Count of the number of matches for Same, Aligned Biometric Samples with Single Index Hashing and Hexa-Decimal PostHash Indices

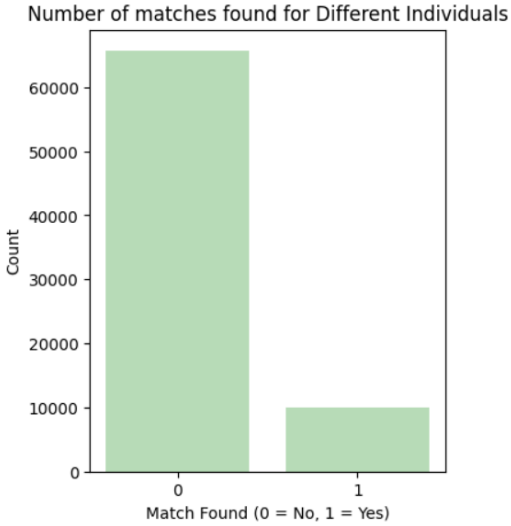


Figure 4.8: Count of the number of matches for Different, Aligned Biometric Samples with Single Index Hashing and Hexa-Decimal PostHash Indices

The theoretical values for q_{same} presented in Table 9 are accurate approximations, as the experimental results closely match these values. The greater discrepancy between theoretical and experimental values for q_{diff} compared to q_{same} can be attributed to the following factor: different biometric samples exhibit higher variability, leading to a broader range of feature differences, which can cause deviations from theoretical assumptions. Outliers and noise further exacerbate the discrepancies for q_{diff}. Moreover, the theoretical values for the parameter combination $m = 1$ and $d = 4$ (last row of Table 9) align almost perfectly with the experimental results, with q_{same} values being identical and q_{diff} values differing by only approximately 0.8%. The lookup table used for this experiment was generated using Equation 4.2, followed by a redistribution of the indices across each bucket to achieve a more uniform distribution. This redistribution process was also applied to the experiment with $d = 3$, achieving experimental results much closer to the theoretical predictions. However, this redistribution was not applied to the first three experiments, which could explain why their results show greater discrepancies.

5 Application: Private and Compact Biometric Matching

This section delves into the practical application of fuzzy hashing within the realm of biometric matching. Employing the Hamming distance for biometric matching offers a systematic approach by iteratively generating l iterations of the preHash function, defined as:

$$\begin{aligned} Hash^{\mathrm{m}}_{\mathrm{key}} &= Hash^{\mathrm{m}}_{\mathrm{key}_1,\ldots,\mathrm{key}_l}(\bar{X}) \\ &= (preHash^{\mathrm{m}}_{\mathrm{key}_1}(\bar{X}), \ldots, preHash^{\mathrm{m}}_{\mathrm{key}_l}(\bar{X})) \end{aligned} \tag{5.1}$$

Subsequently, the Hamming distance between the resulting hash values of two biometric samples X and Y is calculated as:

$$d_H(Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{X}), Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{Y})) = \#\{i : preHash^{\mathrm{m}}_{\mathrm{key}_i}(\bar{X}) \neq preHash^{\mathrm{m}}_{\mathrm{key}_i}(\bar{Y})\} \tag{5.2}$$

This expression quantifies the instances "i" where the outputs of the preHash function differ between samples $\bar{X}$ and $\bar{Y}$.

One notable advantage of this approach is the reduction in size of the stored biometric template. Rather than storing n pixels, ml integers are stored. Additionally, the key renders the reference preHash less privacy-sensitive compared to a biometric template. Specifically, if the key is known, each integer in the hash discloses about $\frac{1}{p}$ pixels, revealing $\frac{ml}{p}$ pixels at worst. This disclosure occurs because, with the keys, the hash values can potentially be reverse-engineered to reveal characteristics of the original biometric pattern. The term p reflects the information entropy associated with each bit being a vein (1) and thus quantifies the average information content each disclosed pixel conveys when the hash is decoded.

Similarly, when employing the Hamming distance for biometric matching through the iterative generation of l iterations of the postHash function, analogous advantages arise. Here, the stored biometric template is condensed to mld integers instead of n pixels. Furthermore, the key diminishes the sensitivity of the reference postHash in terms of privacy, exposing $\frac{mld}{p}$ pixels at most if the keys are known. Additionally, postHash contributes to leakage reduction.

5.1 Theoretical Foundations of FPR and FNR within Fuzzy Hashing Systems

Transforming biometric data into a hash, using methods such as preHash or its more compact version, postHash, plays a key role in enhancing the system's efficiency and security. This process helps protect privacy, which in turn influences important aspects of the system's performance, such as the False Negative Rate (FNR) and the False Positive Rate (FPR). By converting detailed biometric data

into a simpler, hashed format, the system not only uses storage space more efficiently but also reduces the chances of unauthorized access to sensitive information. This transformation is crucial for maintaining the integrity of the data and provides a strong line of defense against potential security threats. Additionally, choosing the right techniques for generating these hashes can greatly improve the system's ability to distinguish between authorized and unauthorized users. This means fewer mistakes in the form of false rejections or acceptances, leading to a more accurate and dependable biometric verification process.

We establish a threshold t to evaluate the match between two biometric samples, $\bar{X}$ and $\bar{Y}$, by analyzing the Hamming distance between their hash values. We define that a match is confirmed if the difference between l (the total number of iterations) and the Hamming distance is equal to or exceeds the threshold t, expressed as:

$$l - d_H(Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{X}), Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{Y})) \geq t$$

In contrast, we define there being no match if the following equation holds:

$$l - d_H(Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{X}), Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{Y})) < t$$

To further refine our understanding, we use statistical methods to approximate this comparison to a normal distribution, allowing us to more accurately calculate the False Negative Rate (FNR).

First, let us define the random variable H as $H = d_H(Hash^{\mathrm{m}}_{key}(\bar{X}), Hash^{\mathrm{m}}_{key}(\bar{Y}))$, representing the number of positions where the preHash values of two samples differ, capturing the number of keys with different hashes. Supposing that the hash keys are generated such that each preHash comparison is independent, the Hamming distance H can be modeled as a binomial random variable, $H \sim Binomial(l, p_H)$, where l is the total number of iterations and $p_H = Pr[Hash^{\mathrm{m}}_{\mathrm{key}_i}(\bar{X}) \neq Hash^{\mathrm{m}}_{\mathrm{key}_i}(\bar{Y})] = 1 - \mu^{\mathrm{m}}_{\mathrm{same}}$ is the probability that the output of the preHash function for a certain key mismatches. This probability uses μ_{same} because we are approximating the FNR, which, by definition, is when the samples should match but don't.
For a large l, the binomial distribution can be approximated by a normal distribution using the Central Limit Theorem. The mean and variance of the binomial distribution are given by

$$\mu_{\mathrm{H,\ same}} = \mathbb{E}(H) = l \cdot p_H = l \cdot (1 - \mu^{\mathrm{m}}_{\mathrm{same}})$$

$$(\sigma_{\mathrm{H,\ same}})^2 = \mathbb{V}(H) = l \cdot p_H \cdot (1 - p_H) = l \cdot (1 - \mu^{\mathrm{m}}_{\mathrm{same}}) \cdot \mu^{\mathrm{m}}_{\mathrm{same}}$$

Therefore, H can be approximated by

$$H \sim \mathcal{N}(\mu_{\mathrm{H,\ same}}, \sigma^2_{\mathrm{H,\ same}})$$

The FNR is the probability that the Hamming distance H exceeds $l - t$ in cases

where there should be a match. This is given by

$$\begin{aligned} FNR &= Pr[l - H < t] = Pr[H > l - t] \\ &= Pr\left[\frac{H - \mu_{\text{H, same}}}{\sigma_{\text{H, same}}} > \frac{l - t - \mu_{\text{H, same}}}{\sigma_{\text{H, same}}}\right] \\ &= \Phi\left(-\frac{l - t - \mu_{\text{H, same}}}{\sigma_{\text{H, same}}}\right) \\ &= \Phi\left(\frac{t - l\mu_{\text{same}}}{\sqrt{l \cdot (1 - \mu^{\text{m}}_{\text{same}}) \cdot \mu^{\text{m}}_{\text{same}}}}\right) \\ &\approx \Phi\left(\frac{t - l\mu_{\text{same}}}{\sqrt{l\mu^{\text{m}}_{\text{same}}}}\right) \end{aligned} \tag{5.3}$$

where the last equality is obtained because $l \cdot (\mu^{\text{m}}_{\text{same}})^2$ is negligeable.

Here, Φ denotes the Cumulative Distribution Function (CDF) of the standard normal distribution, denoted as $\mathcal{N}(0, 1)$.

Using the same deduction as the previous paragraph and considering that for the False Positive Rate (FPR) we use μ_{diff} instead of μ_{same}, we can derive the FPR as follows. The FPR is the probability that non-matching biometric samples are incorrectly identified as matches by the system.

Let H represent the Hamming distance as before. When dealing with non-matching samples, the probability of a mismatch for any given preHash iteration is denoted by μ_{diff}. Thus, the probability of a mismatch is $p_H = 1 - \mu^{\text{m}}_{\text{diff}}$.

For a large l, the Hamming distance H can again be approximated by a normal distribution. The mean and variance for the non-matching case are given by:

$$\begin{aligned} \mu_{\text{H, diff}} &= l \cdot (1 - \mu^{\text{m}}_{\text{diff}}) \\ (\sigma_{\text{H, diff}})^2 &= l \cdot (1 - \mu^{m}_{\text{diff}}) \cdot \mu^{\text{m}}_{\text{diff}} \end{aligned}$$

Therefore, H can be approximated by:

$$H \sim \mathcal{N}(\mu_{H,\text{diff}}, \sigma^2_{H,\text{diff}})$$

The FPR is the probability that the Hamming distance H is less than the threshold $l - t$ in cases where the samples should not match. This is given by:

$$\begin{aligned} \text{FPR} &= Pr[H \leq l - t] \\ &= Pr\left[\frac{H - \mu_{H,\text{diff}}}{\sigma_{H,\text{diff}}} \leq \frac{l - t - \mu_{H,\text{diff}}}{\sigma_{H,\text{diff}}}\right] \\ &= \Phi\left(\frac{l - t - \mu_{H,\text{diff}}}{\sigma_{H,\text{diff}}}\right) \\ &= \Phi\left(-\frac{t - l\mu^{\text{m}}_{\text{diff}}}{\sqrt{l \cdot (1 - \mu^{\text{m}}_{\text{diff}}) \cdot \mu^{\text{m}}_{\text{diff}}}}\right) \\ &\approx \Phi\left(-\frac{t - l\mu^{\text{m}}_{\text{diff}}}{\sqrt{l \cdot \mu^{\text{m}}_{\text{diff}}}}\right) \end{aligned} \tag{5.4}$$

These formulations allow for the evaluation of false match rates based on the standard deviation and mean of the distributions for same and different samples, respectively.

For instance, employing $\Phi(-2.33) \approx 1\%$ as a benchmark, we calculate the threshold (t) from parameters m and l to achieve an FNR of $\approx 1\%$. The theoretical resulting set of parameters is as follows:

m	l	t	$l\mu^{\mathrm{m}}_{\mathrm{same}}$	$l\mu^{\mathrm{m}}_{\mathrm{diff}}$	FNR	FPR
1	217	32	47.85	17.9	$\leq$1%	2^{-10}
1	637	113	140.46	52.55	1.0%	2^{-54}
2	961	31	46.72	6.54	1.0%	2^{-69}
3	2569	15	27.54	1.44	1.0%	2^{-101}

Table 10: Theoretical Parameterization Results for FNR and FPR Calculation, using l Iterations of PreHash

Using compression techniques implemented through postHash, specifically compressing to $D = 16$ ($d = 4$) and $D = 2$ ($d = 1$), we calculate the threshold (t) from parameters $m = 1$ and l to achieve an FNR of $\approx 1\%$. The introduction of compression slightly modifies the equations for FNR (5.3) and FPR (5.4), necessitating the use of q in place of μ^{m}. The theoretical resulting set of parameters is as follows:

m	d	l	t	$lq^{\mathrm{m}}_{\mathrm{same}}$	$lq^{\mathrm{m}}_{\mathrm{diff}}$	FNR	FPR
1	4	1107	258	298	145.3	1.0%	2^{-67}
1	4	347	71	93.4	45.6	$\leq$1.0%	2^{-13}
1	1	3597	2104	2213.2	1895.3	$\leq$1.0%	2^{-20}

Table 11: Theoretical Parameterization Results for FNR and FPR Calculation, using l Iterations of PostHash

5.2 Experimental Derivation of the FNR and FPR for Different (m, l) Parameter Configurations

To experimentally determine the False Negative Rate (FNR) and the False Positive Rate (FPR) for various (m, l) parameter configurations, the initial step is to compute the Hamming distance between $Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{X})$ and $Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{Y})$. The $Hash^{\mathrm{m}}_{\mathrm{key}}$ of an extracted feature vector is generated using l iterations of preHash. We designed an algorithm that accepts a list of l keys (since each preHash iteration requires a new key) and the parameter m, which indicates the number of indices each call to preHash should generate per key. For testing purposes, the l keys are simply integers from 0 to $l-1$, converted into a two-byte binary format. We then calculate the Hamming distance between the resulting hash values for pairs of images. The Hamming distance between $Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{X})$ and $Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{Y})$ is defined as the number of positions at which the corresponding iterations of preHash are different, as defined in Equation 5.2. In this formula, the expression $\#\{i : preHash^{\mathrm{m}}_{\mathrm{key}_i}(\bar{X}) \neq preHash^{\mathrm{m}}_{\mathrm{key}_i}(\bar{Y})\}$ counts the number of iterations i where the pre-hash values of $\bar{X}$ and $\bar{Y}$ differ. Each comparison results in a 1 if the iterations differ and a 0 if they are the same. Summing these comparison results gives the total number of differing positions, thereby computing the Hamming distance between the hash values of the iterations.

Each experiment we conducted uses a different row from Table 10. In each experiment, we generate l preHash iterations for all image pairs and then determine if two images, $\bar{X}$ and $\bar{Y}$, are a match, defined as $l - d_H(Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{X}), Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{Y})) \geq t$. We perform two types of comparisons on our data:

1. **Same Person, Same Finger Comparisons**: This involves comparing different trials of images from the same person and the same finger. These comparisons should theoretically result in a match. If two such images do not match (i.e., $l - d_H(Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{X}), Hash^{\mathrm{m}}_{\mathrm{key}}(\bar{Y})) < t$), this contributes to the FNR.
2. **Different People or Fingers Comparisons**: This involves comparing images from different people and/or different fingers. These comparisons should theoretically not result in a match. If two such images do match, this contributes to the FPR.

To calculate the False Negative Rate (FNR), we count the instances where images that should match do not, and then compute the mean of these instances. Similarly, for the False Positive Rate (FPR), we count the instances where images that should not match do, and then compute the mean of these instances.

For the experiments from Table 10, we obtain the following results:

1. **m = 1, l = 217 and t = 32**
   - $FNR = 18.91\%$
   - $FPR = 2^{-9}$

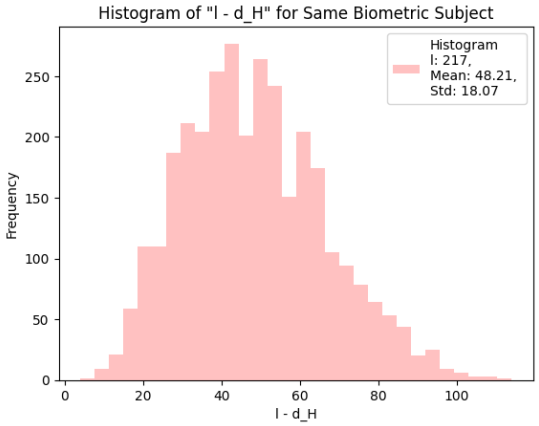


Figure 5.1: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Same, Aligned Biometric Samples with Single Index PreHashing and 217 Total Number of Iterations

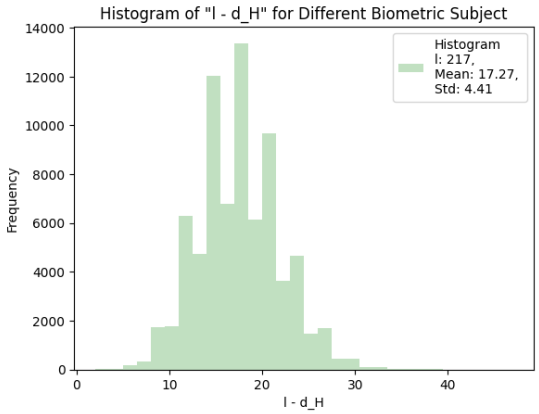


Figure 5.2: Distribution of the Difference between the Total Number of Iterations and the the Hamming Distance between Pairs of Different, Aligned Biometric Samples with Single Index PreHashing and 217 Total Number of Iterations

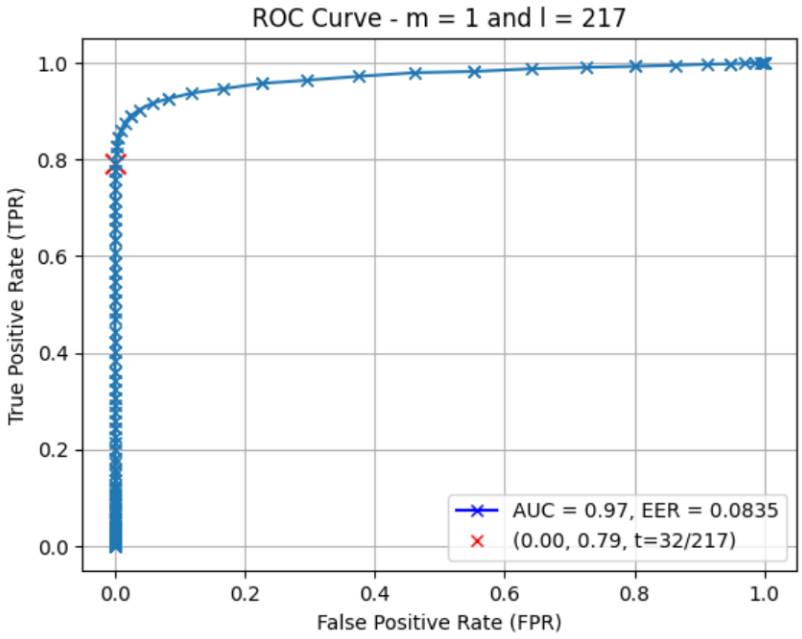


Figure 5.3: ROC for (m = 1, l = 217)

2. **m = 1, l = 637 and t = 113**
   - $FNR = 31.75\%$
   - $FPR = 2^{-14}$

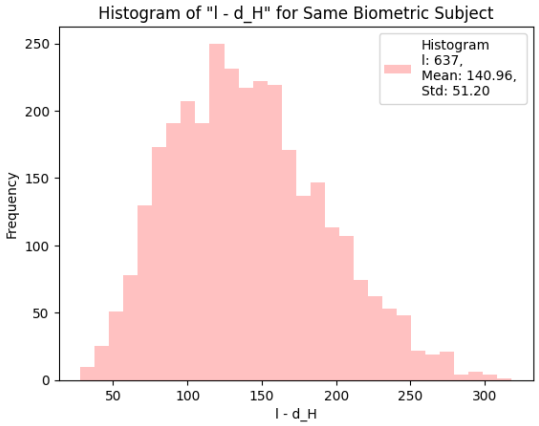


Figure 5.4: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Same, Aligned Biometric Samples with Single Index PreHashing and 637 Total Number of Iterations

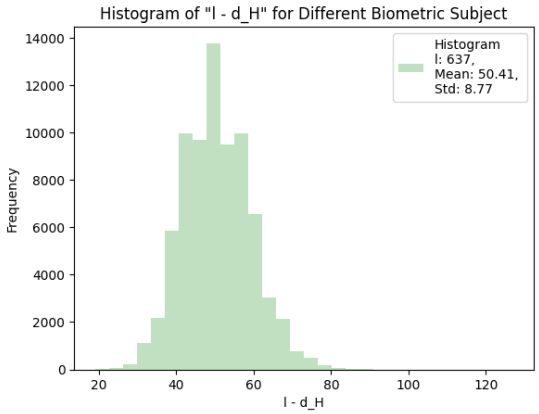


Figure 5.5: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Different, Aligned Biometric Samples with Single Index PreHashing and 637 Total Number of Iterations

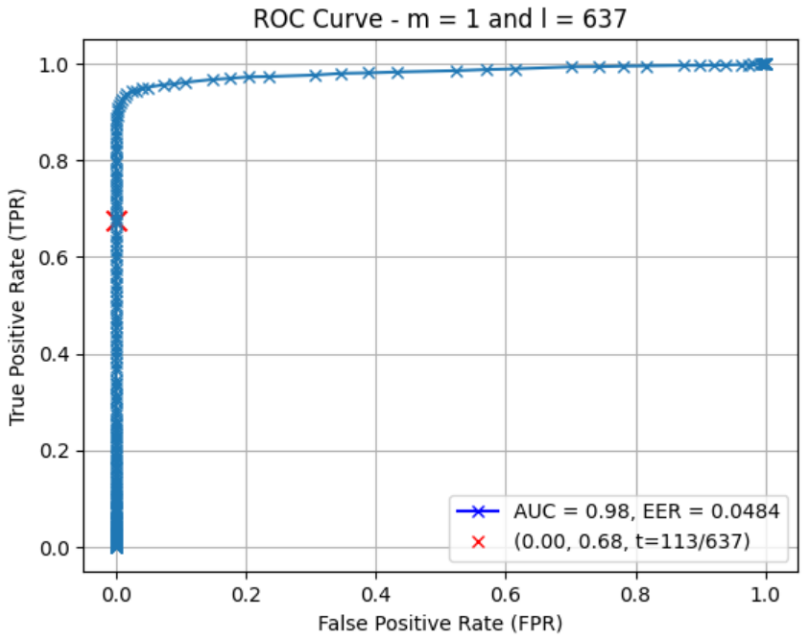


Figure 5.6: ROC for (m = 1, l = 637)

3. **m = 2, l = 961 and t = 31**
   - $FNR = 33.32\%$
   - $FPR = 2^{-14}$

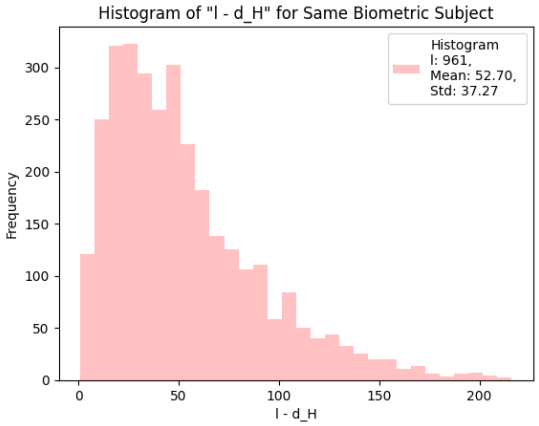


Figure 5.7: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Same, Aligned Biometric Samples with Double Index PreHashing and 961 Total Number of Iterations

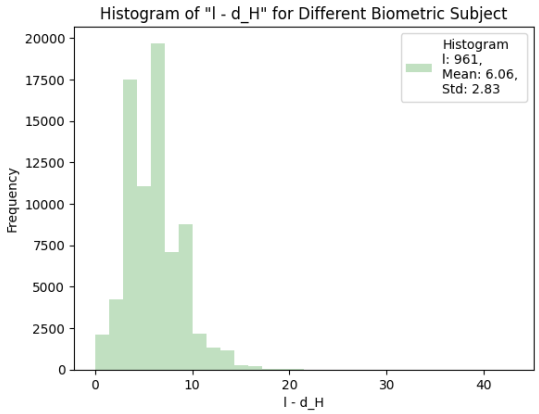


Figure 5.8: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Different, Aligned Biometric Samples with Double Index PreHashing and 961 Total Number of Iterations

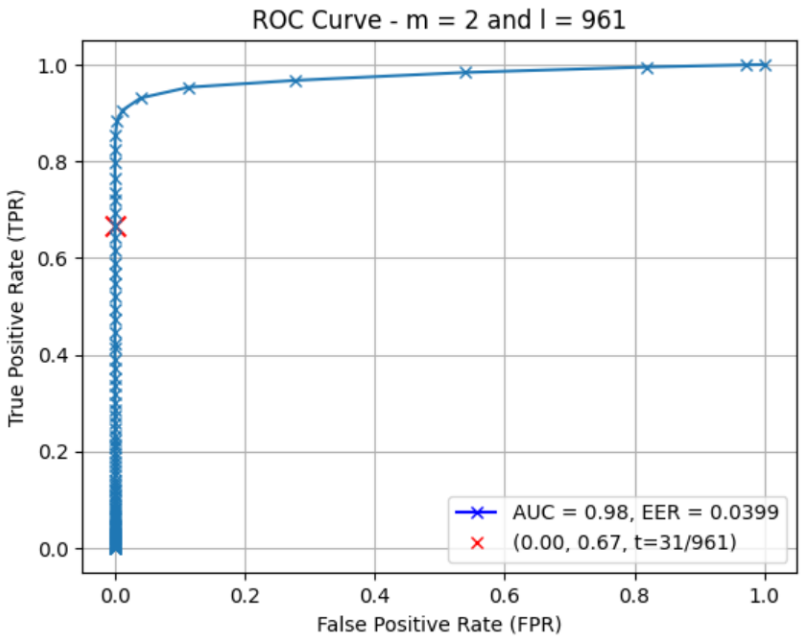


Figure 5.9: ROC for (m = 2, l = 961)

4. **m = 3, l = 2569 and t = 15**
   - $FNR = 33.45\%$
   - $FPR = 2^{-13}$

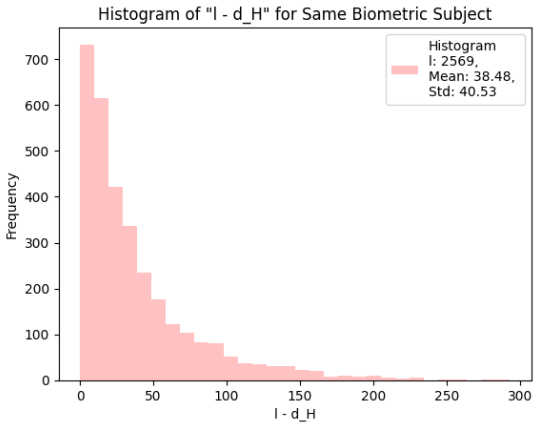


Figure 5.10: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Same, Aligned Biometric Samples with Triple Index PreHashing and 2569 Total Number of Iterations

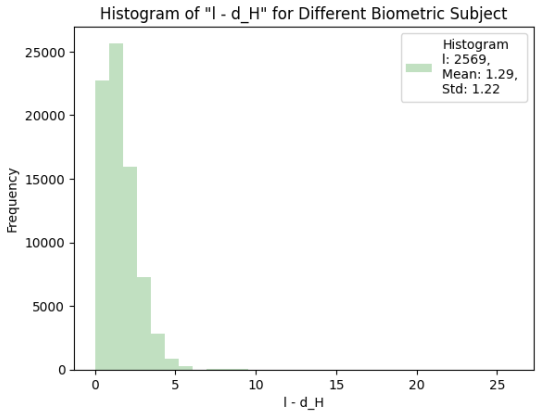


Figure 5.11: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Different, Aligned Biometric Samples with Triple Index PreHashing and 2569 Total Number of Iterations

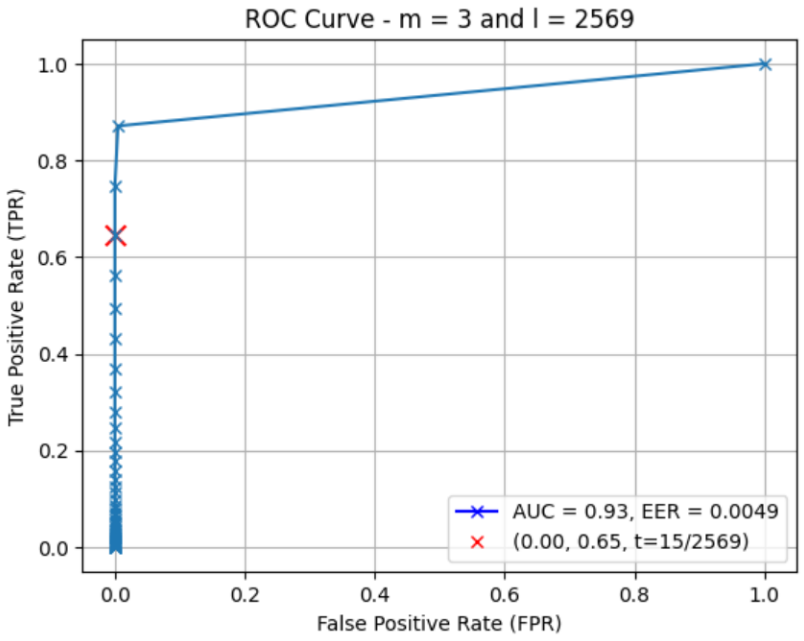


Figure 5.12: ROC for (m = 3, l = 2569)

From all figures in this subsection, we observe that the experimental means of the distributions for same and different biometric samples closely align with the theoretical means predicted ($l \cdot \mu^{\mathrm{m}}_{\mathrm{same/diff}}$) as shown in Table 10. Additionally, the experimental standard deviations of the distributions for different biometric samples are relatively close to the theoretically deduced values, defined as $\sqrt{l \cdot \mu^{\mathrm{m}}_{\mathrm{diff}}}$. However, the standard deviations of the distributions for same biometric samples are

significantly larger than the predicted values, which explains why the histograms of these distributions deviate from the predicted Normal Distribution. This discrepancy could be due to the incorrect assumption that the random variable H follows a Binomial distribution.

It is possible that the assumption of independence among the l hash comparisons generated with their respective keys was incorrect. This dependence might be explained by the consistent identification of areas in the feature vector where vein concentration is higher or lower across all images. Such non-uniformity in the vein patterns could lead to correlations between hash comparisons that violate the independence assumption. Furthermore, we suppose that for two keys to define independent pairs of hashes, the two sequences of indices they define should not have overlapping elements located too far forward. If they do, the probability that $Hash^{\mathrm{m}}_{\mathrm{key}_{i+1}}(\bar{X}) = Hash^{\mathrm{m}}_{\mathrm{key}_{i+1}}(\bar{Y})$ given $Hash^{\mathrm{m}}_{\mathrm{key}_{i}}(\bar{X}) = Hash^{\mathrm{m}}_{\mathrm{key}_{i}}(\bar{Y})$ is higher than the probability that $Hash^{\mathrm{m}}_{\mathrm{key}_{i+1}}(\bar{X}) = Hash^{\mathrm{m}}_{\mathrm{key}_{i+1}}(\bar{Y})$ alone. This might explain the observed dependencies and larger standard deviations.

Further analysis of the variance of the Hamming distances normalized by l (i.e., $\mathbb{V}(H)/l$) across different configurations suggests that this ratio does not behave as a constant. For example, in our experiments, this ratio varied significantly for both same and different biometric samples. Specifically, for same biometric samples, the ratio decreased from 4.1152 to 0.6394 as l increased from 217 to 2569. Similarly, for different biometric samples, the ratio decreased from 0.1209 to 0.0006 over the same range of l. This indicates a complex relationship between l and the variance of Hamming distances, as observed in Figure 5.13. It is important to note that the number of experiments conducted to plot this figure is insufficient to accurately determine the exact nature of this variance, and thus these results should be interpreted with caution.

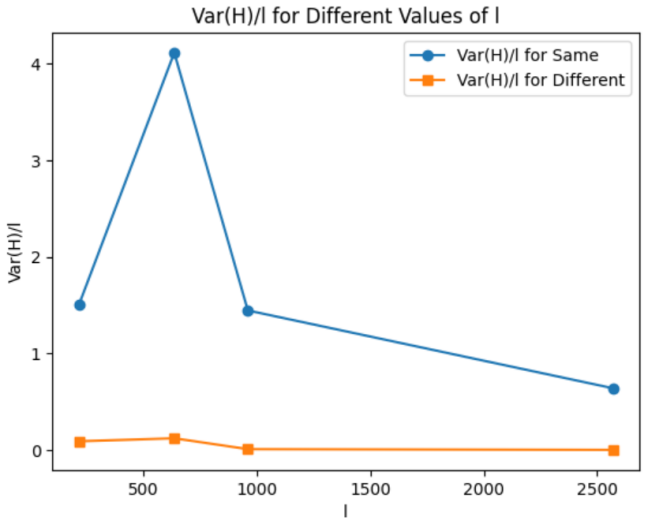


Figure 5.13: Ratio $\frac{\mathbb{V}(H)}{l}$ for both 'Same' and 'Different' biometric samples

Furthermore, the ROC curves plotted in this section provide valuable insights into

the system's performance across different parameter configurations. These curves illustrate the trade-off between the False Positive Rate (FPR) and True Positive Rate (TPR) at various threshold settings. Although the area under the curve (AUC) values of our ROC curves are close to 1, indicating a generally strong performance, the system's overall effectiveness is compromised by high False Negative Rate (FNR) values. This is evident from the plots, where achieving a low FPR does not correspond with a TPR close to 1, resulting in an undesirably high FNR (FNR = 1 - TPR). Evaluating whether the system is optimal depends on the specific real-world application and its tolerance for false positives and false negatives. For applications requiring high security, a lower FPR might be prioritized, even at the expense of a higher FNR. Conversely, for applications where user convenience is critical, a higher TPR might be favored, accepting a slightly higher FPR. This underscores the necessity of tuning the system parameters to strike an appropriate balance based on the intended use case.

Interestingly, we observe that increasing m and, more specifically, l, does not necessarily lead to a better FPR. In some configurations, more iterations do not improve the system's ability to correctly identify different biometric samples, indicating that simply increasing the number of hash comparisons may not enhance performance as expected. In our experiments, while some configurations with higher l showed ROC curves closer to the top-left corner, indicating better performance with lower FPRs and higher TPR's, others did not. This variability suggests that there is a complex relationship between the number of iterations and the system's performance metrics.

5.3 Experimental Derivation of the FNR and FPR for Different (m, l, d) Parameter Configurations with Compression

In this subsection, we will experimentally evaluate the False Negative Rate (FNR) and False Positive Rate (FPR) with compression. We designed an algorithm that uses the previously defined method to generate l iterations of preHash and then compresses each hash value generated depending on the value of $D = 2^d$. For an image, the algorithm generates a list of the l iterations of preHash and then applies the postHash algorithm to each preHash output.

After both images pass through this process, each iteration of each image is compared, resulting in a 1 if the iterations differ and a 0 if they are the same. Summing these comparison results gives the total number of ones, effectively computing the Hamming distance between the compressed and hashed images.

1. **d = 4, m = 1, l = 1107 and t = 258**
   - $FNR = \%$
   - $FPR =$

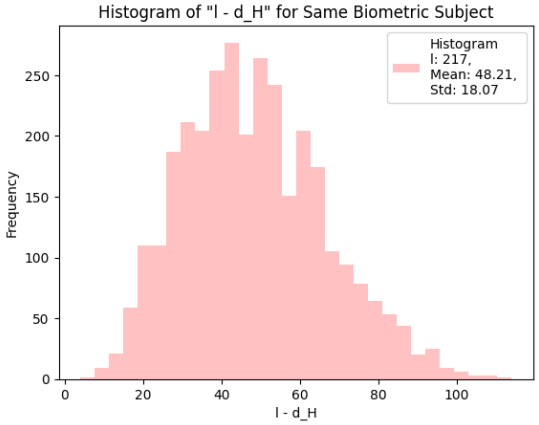


Figure 5.14: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Same, Aligned Biometric Samples with Single Index PreHashing, Hexadecimal PostHash Compression and 1107 Total Number of Iterations

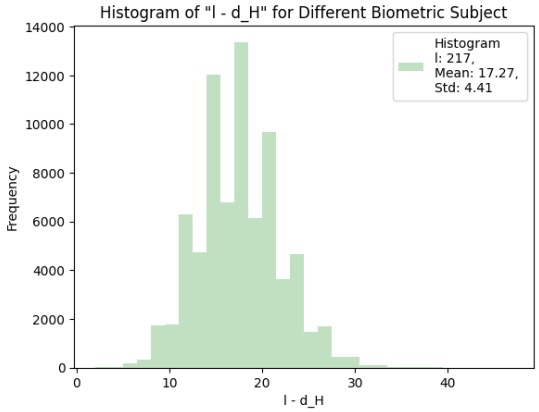


Figure 5.15: Distribution of the Difference between the Total Number of Iterations and the the Hamming Distance between Pairs of Different, Aligned Biometric Samples with Single Index PreHashing, Hexadecimal PostHash Compression and 1107 Total Number of Iterations

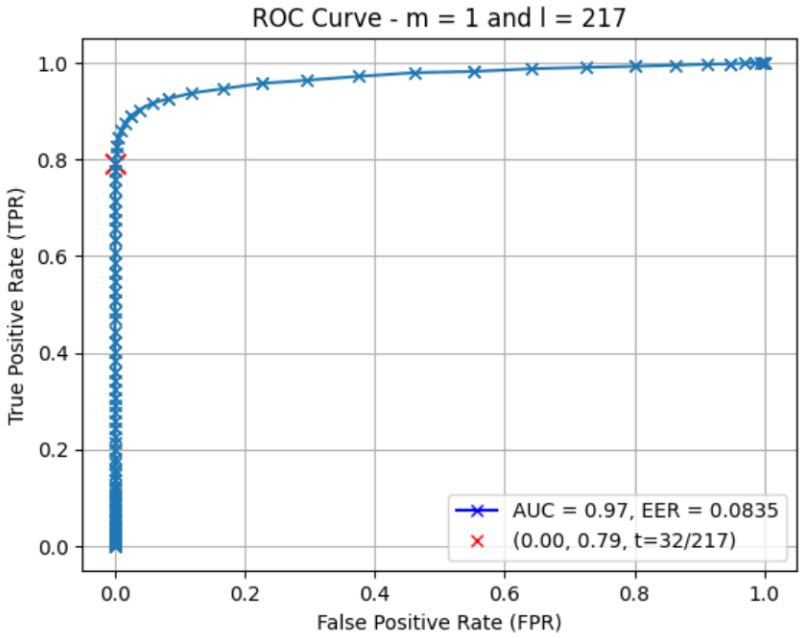


Figure 5.16: ROC for (d = 4, m = 1, l = 1107)

2. **d = 4, m = 1, l = 347 and t = 71**
   - $FNR = 25.69\%$
   - $FPR = 2^{-11}$

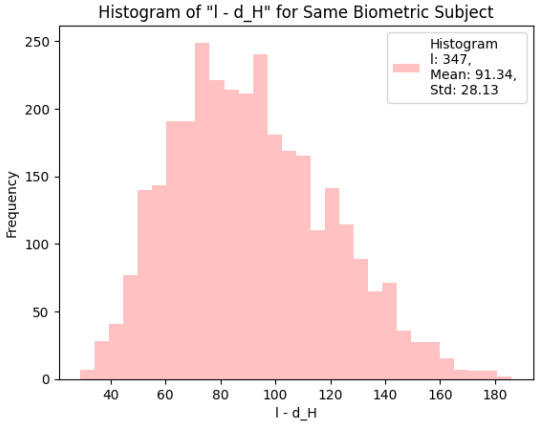


Figure 5.17: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Same, Aligned Biometric Samples with Single Index PreHashing, Hexadecimal PostHash Compression and 347 Total Number of Iterations

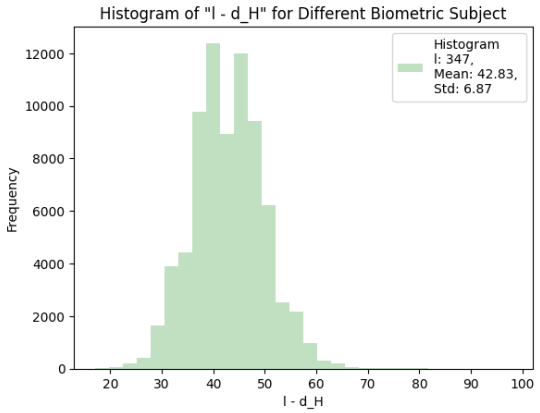


Figure 5.18: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Different, Aligned Biometric Samples with Single Index PreHashing, Hexadecimal PostHash Compression and 347 Total Number of Iterations

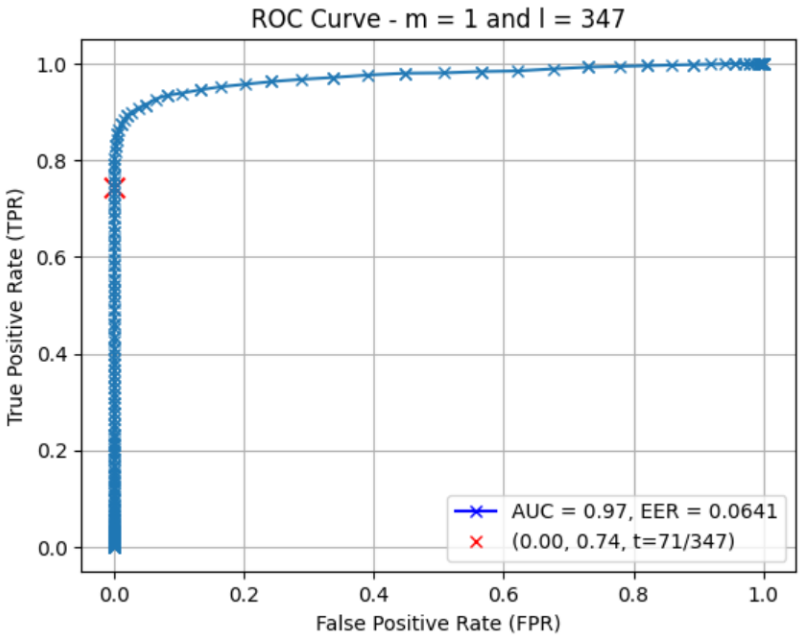


Figure 5.19: ROC for (d = 4, m = 1, l = 347)

3. **d = 1, m = 1, l = 3'597 and t = 2'104**
   - $FNR = \%$
   - $FPR =$

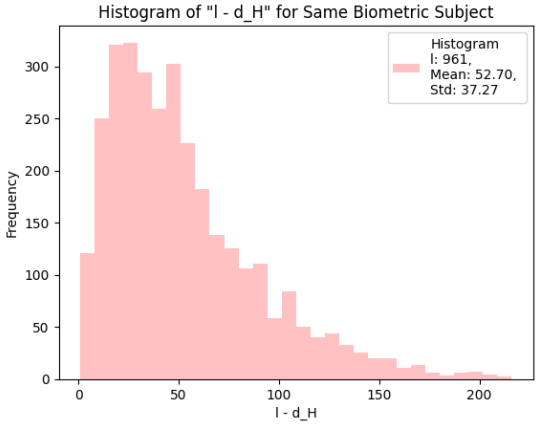


Figure 5.20: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Same, Aligned Biometric Samples with Double Index PreHashing, Binary PostHash Compression and 3′597 Total Number of Iterations

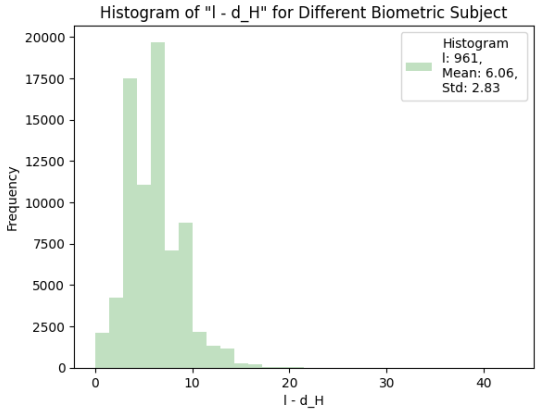


Figure 5.21: Distribution of the Difference between the Total Number of Iterations and the Hamming Distance between Pairs of Different, Aligned Biometric Samples with Double Index PreHashing, Binary PostHash Compression and 3′597 Total Number of Iterations

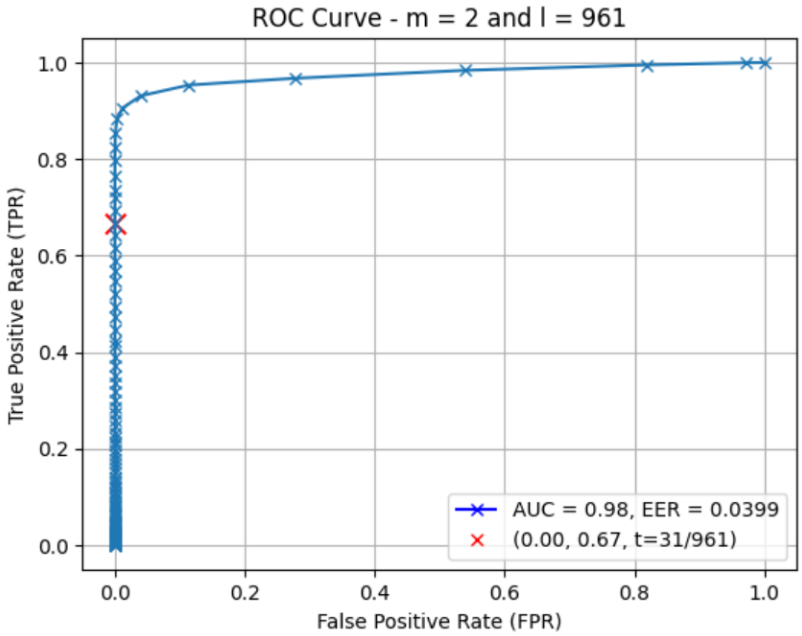


Figure 5.22: ROC for (d = 1, m = 1, l = 3597)

These experimental results are similar to those obtained without compression, where the results did not align as expected with the theoretical predictions. The above figures illustrate these outcomes, but the underlying reasons for the discrepancies remain consistent with those discussed in the previous section.

6 Conclusion

In this report, we have explored the application of fuzzy hashing to finger-vein biometric authentication, aiming to enhance both security and efficiency. Our investigation included the development and assessment of both the preHash and postHash algorithms, which transform biometric data into secure hash values that maintain consistency despite slight variations in the input data. The results of the experiments conducted throughout our project generally aligned well with the predicted values. However, as detailed in Section 5, some discrepancies arose because certain assumptions of independence may not have been accurate.

6.1 Challenges and Future Directions

Throughout our work on this project, we encountered numerous challenges. The first significant difficulty was integrating and building upon the work of previous students. Understanding previously written code can be extremely challenging, highlighting the crucial importance of thorough documentation. We wrote a substantial amount of code for our project, ranging from algorithms such as preHash and postHash to various experimental procedures. Given the initial struggle to comprehend the existing codebase, we prioritized documenting our contributions meticulously. This included adding detailed comments and enhancing the README file to ensure clarity and ease of understanding for future developers. We believe that the comprehensive documentation we have provided will significantly benefit anyone who continues to work on this project, ensuring that our code is accessible and easy to understand.

Another significant challenge was running the experiments. With a dataset of 800 images, each experiment generated four populations of results:

- Population I: Comparisons between same biometric subjects taken from camera 1 ($1'591$ comparisons).
- Population III: Comparisons between same biometric subjects taken from camera 2 ($1'591$ comparisons).
- Population II: Comparisons between different biometric subjects taken from camera 1 ($37'809$ comparisons).
- Population IV: Comparisons between different biometric subjects taken from camera 2 ($37'809$ comparisons).

This amounted to a total of 78,800 comparisons per experiment, making the runtime extremely long and we spent alot of time waiting for results. We should have addressed this issue earlier but only realized the severity of the runtime problem when conducting experiments for Section 5. The extended runtime for multiple iterations of preHash and postHash combinations for each image highlighted the inefficiency. To tackle this, we revisited the previously developed pipeline code and implemented parallel processing. Additionally, we gained access to more powerful ressources (SCITAS[4]) to run our code more efficiently. These changes drastically reduced the runtime: experiments that previously took over 24 hours to complete were reduced to just 1 hour.

However, it's important to note that despite the improved runtime, the experiments still consume a lot of resources. Some experiments, such as the last few rows of Table 10, still take between 1 and 4 days to run. This issue can only be further addressed by refining the actual pipeline, specifically the extraction of feature vectors, and by converting the code, which is currently written in Python, to a lower-level language. This transition would likely result in more efficient code execution and reduced resource consumption. The current system requires this especially because future work will be working on a concept called "1:N matching", which adds another layer of complexity to the system, as this process involves comparing a single biometirc input against a database of multiple potential matches to identify the closest match or matches accurately.

7 Definitions

Equal Error Rate (EER) Metric used to evaluate the performance of a system. It represents the point at which the system's false acceptance rate (FAR) equals its false rejection rate (FRR). A lower EER indicates a more accurate and reliable system as it signifies a balanced trade-off between security (minimizing FAR) and usability (minimizing FRR)

False Positive Rate (FPR) This is the probability of incorrectly accepting an unauthorized user

False Negative Rate (FNR) This is the the probability of incorrectly rejecting an authorized user

Hash Function A hash function is an algorithm that converts input data of any size to a smaller fixed-size string of characters, which typically acts as a data fingerprint. The output, known as a hash, is unique for different inputs in ideal cases, making hash functions crucial for cryptography, data integrity, and indexing in databases.

Fuzzy Extractors Fuzzy extractors are cryptographic tools designed to reliably and securely generate a consistent, reproducible cryptographic key from biometric data or other noisy inputs that are inherently inconsistent. They enable the extraction of a stable key from an input that may vary slightly over different measurements, ensuring that even with minor variations, the same key can be reliably regenerated. This process typically involves two main components: a generator that produces a stable key and some public data from an initial input, and a reproducer that can regenerate the original key from a similar but not identical input using the public data.

Uniform Distribution A uniform distribution signifies that each outcome within a set has an equal chance of occurring. In the context of finger vein patterns, it means any bit i in the biometric capture, representing either the presence or absence of a vein, is equally likely to be selected for analysis.

Hamming Distance The number of positions at which two biometric strings of equal length differ. It measures the similarity between the strings, with a lower distance indicating higher similarity.

Hamming Weight The number of 1's in a biometric string, indicating the presence of veins.

AES in CTR mode AES in Counter (CTR) mode is an encryption method that transforms a block cipher into a stream cipher. It achieves this by encrypting sequential counter values using AES. Each counter value is unique for each block of data, typically starting from an initial nonce (number used once) and incremented for subsequent blocks. This encryption method is traditionally used to encrypt data by generating a sequence of encrypted counters, which are then XORed with plaintext to produce ciphertext, allowing for encryption of arbitrary-sized data without padding. However, in the context of fuzzy

hashing for biometric matching, AES-CTR is repurposed to generate a pseudorandom sequence, not for encrypting data but for selecting specific indices from a biometric template. This application leverages AES-CTR's cryptographic strength to ensure unpredictability and determinism in the selection process.

SHA-256 Hash SHA-256 is a cryptographic hash function in the SHA-2 family that produces a fixed-size 256-bit (32-byte) hash value from an input of arbitrary length. It has three fundamental properties: pre-image resistance, making it computationally infeasible to reverse the hash to find the original input; second pre-image resistance, preventing the discovery of a different input producing the same hash as a given input; and collision resistance, making it highly unlikely to find two different inputs that yield the same hash output. These properties ensure data integrity and security across various digital applications.

References

[1] Bernardo de Araujo. *Counter (CTR) mode encryption*. URL: https://bernardoamc.com/ctr-mode-introduction/.

[2] asee. *History of Authentication: From Zero to Hero*. 2022. URL: https://cybersecurity.asee.io/blog/history-of-authentication/.

[3] Scholarly Community Encyclopedia. *Partition Problem*. 2023. URL: https://encyclopedia.pub/entry/28608.

[4] EPFL. *SCITAS User Documentation*. 2024. URL: https://scitas-doc.epfl.ch/.

[5] LoginTC. *Biometric Authentication*. 2024. URL: https://www.logintc.com/types-of-authentication/biometric-authentication.

[6] Microsoft. *What is authentication?* 2024. URL: https://www.microsoft.com/en-us/security/business/security-101/what-is-authentication.