Fuzzy Hashing

Lea Grieder (328216)
Leila Sidjanski (330810)

February 2024

EPFL

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[1]. Various methods such as password-based authentication, certificate-based authentication, one-time passwords, multifactor authentication, and biometric authentication are employed[2].

Biometric authentication, which involves analyzing unique physical characteristics, are 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[3]. 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

[1] https://cybersecurity.asee.io/blog/history-of-authentication/
[2] https://www.microsoft.com/en-us/security/businesssecurity-101what-is-authentication.
[3] https://www.logintc.com/types-of-authentication/biometric-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 Presentation of the Project

This project extends the work on optimizing a finger-vein recognition pipeline that has demonstrated the lowest Equal Error Rate (EER) 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. This project aimed to find an alignment technique that could work effectively without accessing both the model and probe simultaneously, a significant challenge given the variability in how users may place their finger on the scanner.

Simon has developped a pipeline (see Figure **??**) to align finger-vein images independently, 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 Caclulation**: 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.

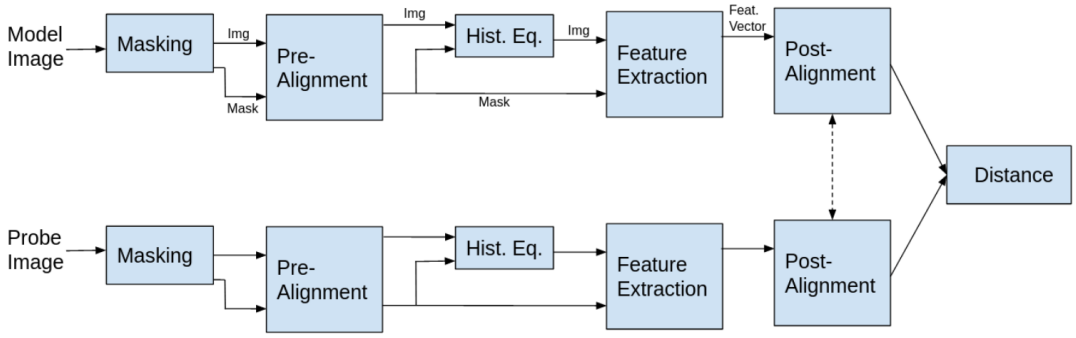

Figure 1.1: Simon's Extraction Pipeline. Compares a single probe image to a model image

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.

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. Our aim is to incorporate a hashing step at the end of the already developed pipeline, subsequent to the post-alignment phase. The integration of this hashing process at the current pipeline's conclusion is aimed

at achieving several key objectives, enhancing the system's overall functionality and security. 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 authentication, 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.

We will implement....

1.3 Structure of the Report

2 Theoretical Framework

2.1 Biometric Setting

Explain section 1 of the document (parameterization and representation of biometric data). Explain what biometric data is and its importance in security applications.Explain the transformation of finger pictures into compressed formats for vein pixel extraction (Simon's pipeline). Explain the probability models of section 1: introduce p and the concept of matching biometric captures with an optimal offset, denoted by sigma. to minimize the mismatch probability (p(Xi!=Yi))

2.2 Concept of Fuzzy Hashing

Explain what fuzzy hashing is and how it is different from standard hashing in security systems. Describe the PreHash functionality on biometric data, using random permutation to select specific indices corresponding to feature points. Explain the formula for comparing two pre-hashed biometric captures using the hamming weight and the bitwise AND operation to assess the match probability. Explain the calculation of the matching score, using the ratio of the hamming weight of the conjunction of two bit strings to their combined hamming weights. Explain what mu is.

2.3 Compressed Fuzzy Hashing

Explain how post-hash functions compress the pre-hash bit strings into a more compacted form, maintaining near uniform distribution when inputs are random. Discuss the implications of tis compression on matching probability and introduce the concept of D=2 puissance d to understand the compression ratio and its effects on hash collision probabilities. Detail the formula calculating the probability of hash matches after compression and the role of mu in determining these probabilities, particularly how it adjusts based on the distribution of biometric captures (same, different, independent)

2.4 Pre-Hash and Post-Hash Implementation and Analysis

Explain the algorithm and how we have implemented it

2.5 Applications in Biometric Macthing

Illustrate the use of fuzzy hashing in biometric matching by employing the hamming distance to compare hashes, reducing the data required for storing biometric templates and enhancing privacy. Explain how a threhold is defined for determining a match and how false negative rates (FNR) and false positive rates (FPR) are calculated using the cumulative distirbution function (CDF) of the normal distribution, emphasizing the statistical approach to balancing match accuracy and security.

2.6 1:N Matching

Discuss the challenges and strategies for matching a single biometric sample against a large database, focusing on the computational and memory complexities involved. Explain how parameters are adjusted to manage trade offs between time complexity, space complexity, and match probability. Provide insight into the statistical models used for evaluating the performance of 1:N natching systems, including the dervia-tion of FNR and the optimization of parameters for efficient and accurate matching across large datasets.

3 Experimental Verifications

3.1 Methodology for Assessing Theoretical Values (p, delta, mu)

3.2 Analyzing FPR and FNR (Disussion of Results)

4 Optimization Strategies

4.1 Data Compression Techniques for m=1, d=4

4.2 Efficiency Improvement in Hashing Process

5 [1:N] Matching and System Evaluation

5.1 Implementation of [1:N] Matching

5.2 System Performance Evaluation

6 Conclusion

6.1 Future Directions and Enhancement

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 Acceptance Rate (FAR) This is the probability of incorrectly accepting an unauthorized user

False rejection Rate (FRR) 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.