Secure and Privacy-Preserving User Authentication Using Biometrics

Nikolaos Theodorakis

Thesis submitted for the degree of Master of Science in Digital Systems Security. *Developed under the Erasmus framework*

Thesis supervisor:
Prof. Konstantinos Lambrinoudakis

Mentors:
Prof. dr. ir. Bart Preneel
Christina-Angeliki Toli
Enrique Argones Rúa

Academic year 2017 – 2018

Preface

First of all, I would like to thank my promotor Prof. Bart Preneel for giving me the opportunity to carry out my Master's thesis at COSIC as an Erasmus student. I would like to express my utmost gratitude to my two daily supervisors, Christina-Angeliki Toli and Enrique Argones Rua for their help in improving my language and academic writing skills, their extensive feedback and proof reading my thesis. Without their continuous guidance and support, this text would have never be completed.

Finally, my sincere gratitude goes to my family and friends for their great support and encouragement during my stay in Leuven.

Nikolaos Theodorakis

Contents

Abstract

Identity management lies in the field of Information Security, presenting numerous attractive research categories. Biometrics have been established as a new approach to mitigate the limitations and weaknesses of traditional access methods of passwords and tokens. However, biometrics introduce new security and privacy risks since they cannot be easily revoked.

Due to the noisy nature of biometrics, traditional cryptography cannot be used to efficiently protect biometric systems. Thus, template protection schemes have been developed in order to encrypt and decrypt biometric data in an error-tolerant way.

In this thesis, we design, implement and evaluate a fuzzy vault template protection scheme, in order to protect fingerprint minutiae points. Fuzzy vault schemes can protect a secret value using a biometric template, and this secret value can be decrypted only if an input template overlaps significantly with the original one. In these designs, the security is based on the hardness of the polynomial reconstruction problem.

During this research, the most challenging issue that has been faced is the issue of alignment for the biometric templates in the encrypted domain. State-of-the art is discussed, being focused on the use of minutiae points and any information extracted from the fingerprint image. By addressing the advantages and disadvantages of the suggested methods, we utilize an alignment technique to handle the problematic area of the minutiae patterns alignment in cryptographic approaches. This method is considered to be ideal for private friendly biometric designs based on stored minutiae points, instead of full fingerprints, rendering irreversible the access at the original images.

After the fuzzy vault implementation, our experiments show that there is significant trade-off between system's security and performance, under different parameter values. Finally, the results of this thesis can provide a useful tool for other researchers, since our findings indicate which of the combinations for the experiments can be utilized for the design of a biometric scheme that is both accurate and secure.

List of Figures and Tables

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Goals and Contribution

In this thesis we design, implement and evaluate a biometric template protection scheme for minutiae-based fingerprint templates, exploiting the advantages that fuzzy vault designs can offer. From a performance perspective, we aim to design a system that is as accurate as possible, minimizing false acceptance rates for impostors and false rejection rates for genuine users. Our scheme also needs to be fast, in terms of computational complexity, able to produce results in a feasible time period. From a security perspective, our system has to provide provable security, preserving user's privacy as well.

A well-known problem found in fingerprint biometric systems is the migration from an unprotected minutiae-based system to a protected one. The reason is that, in practice, fingerprints are stored in a minutiae-based format to avoid feature extraction during every matching. In this work, we are focused only on minutiae-based fingerprint template protection schemes, even though, in principle, our system can be used with any biometric modality. The alignment of two templates in the encrypted domain is the most significant problem that researchers can meet in fuzzy vault schemes for fingerprint biometrics. To mitigate this issue, we use a minutiae-based alignment method by utilizing some of the enrolled template's minutiae as helper data. The goal of this alignment technique is to assist in the alignment of two templates without revealing any information about the enrolled template that could allow to an adversary the access at the genuine user's fingerprint data or the possible authentication of an impostor.

During the last decade, research is usually focused on the use of minutiae as helper data, but without taking into account the security implications of this specific method. Through experiments, it can be easily proved that the use of helper data increases the chances of both genuine and false acceptance rates. Since the purpose of the helper data is to assist in the alignment when the enrolled template is encrypted, this helper data will be either unprotected or its protection will not be part of the template protection scheme. In the worst-case scenario, in our threat model we should consider that the adversary might have access to this helper data. In the case of the fuzzy vault approach, minutiae that are stored inside the vault can actually decrypt it. This fact explains the reason why minutiae stored in the vault should not be used as helper data. The removal of minutiae of the vault or the helper data degrades the overall accuracy of the system, since it reduces the chances of the vault's decryption. Researchers in the past did not remove minutiae in their implementations, designing schemes that produce promising results but not secure enough. Our contribution aims to address this issue by implementing a scheme that does not use the same minutiae in the vault and for the alignment, thus mitigating the risks expressed above. We also investigate experimentally the performance trade-off, using minutiae either in the alignment or the vault, mainly based on their quantity and quality. We present our results where we identify how each of these options affect on the performance of our system and we provide specific parameter values that can maximize the performance.

Finally, we develop a simple alignment evaluation method based on relative distances between minutiae points, which is independent from the fuzzy vault scheme that we will use for our experiments. Since the issue of a precise alignment is necessary for the accuracy of the fuzzy vault, our alignment evaluation can assist in identifying the upper limits of our vault's performance, locating the potential errors. In that way we aim to address the possible areas that improve the overall accuracy through the correction of either the alignment or the fuzzy vault design.

1.2 Outline

Chapter 2 presents the necessary background on the science of biometrics, explaining how an arbitrary biometric system works, providing the standard terminology and performance evaluation techniques used in the field and introducing fingerprint characteristics.

Chapter 3 discusses briefly the cryptographic template protection techniques used to protect biometrics and how they influence security and privacy. We describe in detail the fuzzy vault scheme that we implement in this thesis, along with its security performance, possible attacks and countermeasures. Finally, we highlight the most popular fuzzy vault designs found in the literature.

Chapter 4 introduces our approach to the fuzzy vault scheme. Initially, we propose the alignment technique along with an evaluation method, and secondly our fuzzy vault implementation is presented. We discuss the challenges that have lead to our implementation choices, affecting the performance and accuracy of our system.

Chapter 5 describes the performance evaluation for our experiments, by presenting and comparing our results under various parameters values.

Finally, in Chapter 6 we provide a general conclusion on advantages and limitations of our system, as well as potential applications. To conclude, we present our ideas for future research on the security perspectives for fuzzy vault designs.

Chapter 2

Theory and Background

This chapter stands as a general introduction to the field of biometrics. Firstly, all the necessary definitions of biometric systems are provided. Specifically, it is explained what is an arbitrary biometric system and how this works; how biometric characteristics are categorized, and how we can evaluate the performance of such schemes. Accordingly, the fingerprint characteristics and the corresponding approaches for the feature extraction and the matching procedures are presented.

2.1 Introduction

The traditional approaches for user authentication are as follows:

- something the user *knows* (passwords, PIN's)
- something the user *has* (cards, tokens)
- something the user *is* (biometrics)

The first two approaches are still considered as mechanisms that they have not yet provided a secure and usable solution to the authentication related problems. Namely, passwords are considered as easy to be forgotten, get stolen or even cracked if not chosen wisely by the user. Password management can become a really hard task when a user has to remember dozens of different passwords for different services. Entering passwords is a time-consuming process, and it may be inconvenient in public areas. Regarding the token-based authentication procedure, it seems helpful in some of the previously mentioned problems, especially when it is used under the combination of a password in a two-factor authentication design. In this configuration, an attacker will need to steal both the token and the password in order to authenticate the claimed identity. However, this approach introduces new challenges, since tokens can be lost and stolen as well, and they are much more expensive to be implemented in a secure system. Consequently, biometrics have been introduced in order to mitigate such issues, ensuring a reliable authentication environment [10].

2.2 Fingerprints

2.2.1 Introduction

A *fingerprint*, as a representation of the epidermis of a human finger, consists of patterns and combinations of *ridges* and *valleys* [23]. Ridges are the dark lines creating the actual fingerprint, whereas valleys are the white space between them, as these can be shown in Figure 2.1.

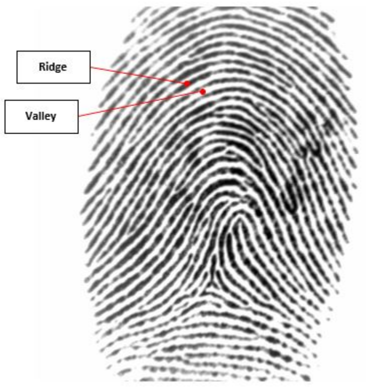
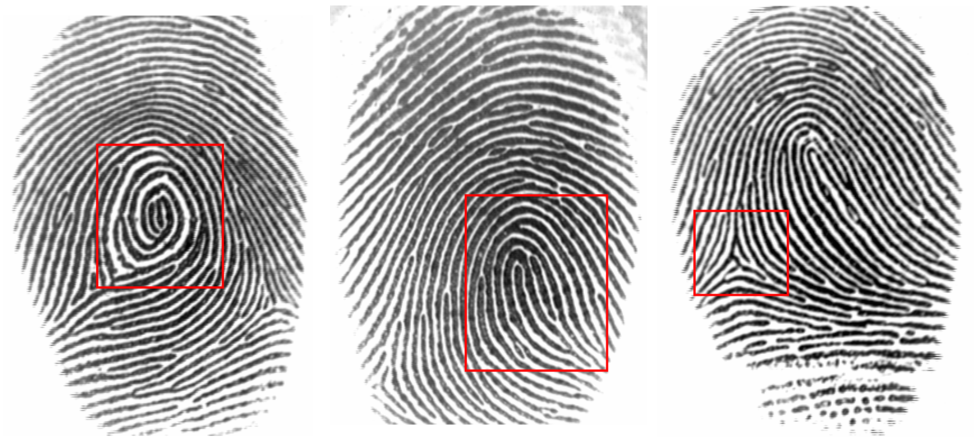


Figure 2.1: Ridges and valleys of a typical fingerprint

These patterns are shaped according to genetic and environmental factors and they are different even among identical twins [12], thus making the fingerprint an ideal biometric modality. Two decades ago, traditionally, the fingerprints were collected by an ink-painted finger against a paper, nowadays electronic sensors (usually called fingerprint scanners) are used to collect a digital image of the fingerprint. The sensor is probably the most important level of a fingerprint biometric system, since the quality of the collected sample greatly influences the performance of the whole system. The quality of the sample can be influenced negatively by a poorly cleaned sensor, skin distortion or wet fingers.

At global level, three main classes of patterns can characterize a fingerprint [22]. These are the *loops*, *whorls* and *deltas*. Loops have the shape of $\cap$, whorls stand as O, while deltas as Δ respectively. Figure 2.2 presents these regions. At local level, the most important characteristics defining a fingerprint are the *minutiae*. A minutia is formed at the discontinuation of a curve (line) on a fingerprint. The two most common minutiae is the *ridge ending* and the *bifurcation*. A ridge ending is formed when a ridge simply stops, whereas a bifurcation is formed when a ridge is split into two others, as shown in Figure 2.3.



(a) whorl (b) loop (c) delta

Figure 2.2: Main types of global regions

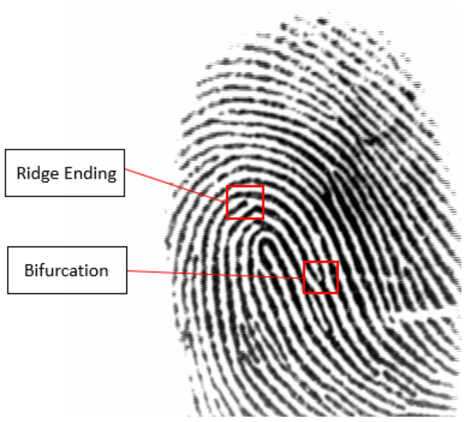


Figure 2.3: Ridge ending and bifurcation

2.3 Terminology

Biometrics are defined as the science of establishing the identity of an individual considering the physical, chemical or behavioural attributes of one person [30]. A biometric design is essentially a system that acquires biometric data from an individual. From this data, useful information is extracted, in a specified format such as features. For the matching process, this information is compared with stored biometric sets in a database. These steps can be described thoroughly in the following modules [13]:

1. **Sensor module**
   Typically, a sensor is a reader or a scanner device that acquires the raw biometric features coming from an individual. For instance, a

camera can acquire a person's face or an iris view of human eye, or a fingerprint scanner that reads a fingerprint image from a user's finger.

2. **Quality assessment and feature extraction module**
   A quality assessment test is an optional procedure where an algorithm may decide if the acquired raw biometric data is suitable for further processing. If the quality of this data is considered as poor, then the user may be asked to insert the biometrics again. Furthermore, the quality of the acquired data is enhanced using signal enhancement algorithms. During the feature extraction process, specific biometric data is extracted from the raw biometric input in such a form that is suitable for the following procedures of processing and comparison.

3. **Matching and decision-making module**
   During the matching procedure, a system utilizes an algorithm that compares the input biometric data with those that have been stored after the enrollment phase, in order to return the decision for the acceptance of a genuine user or not. Further definitions regarding the matching process are described below.

4. **System database module**
   The database module is named after the stage of the storage of human biometric data of the successfully enrolled (see definition below) users, along with other useful information about each user, such as names or other identity references, usually called as helper data.

As described in [25], a specific trait or characteristic is referred as a biometric *modality*. Examples of such modalities are fingerprints, iris, facial geometry and voice among others. A biometric reference stored in the database in an appropriate format for comparison is called *template*. The process of acquiring, extracting features and storing them in a database is known as *enrollment*, whereas we refer to *matching* as the comparison between a query template and an enrolled template, and the respective decision upon the result as a *matching score*. The enrollment procedure is shown in Figure 2.4.

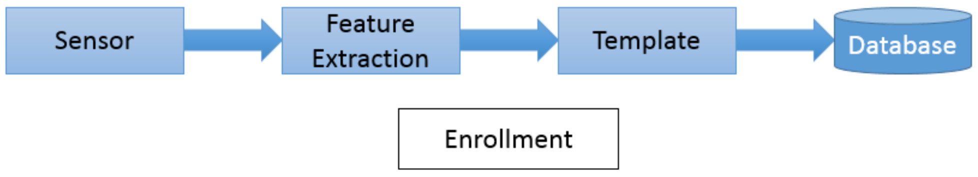


Figure 2.4: Enrollment procedure

The decision upon a *match* (acceptance as a genuine person) and a *non-match* (rejection as an impostor) is based upon a pre-arranged value named *threshold*. The algorithm that decides if a query template is genuine or not is called *matcher*. The matching procedure can be summarized in Algorithm 1.

Algorithm 1 Matching

```
1: procedure Matching
2:     $matchingScore \leftarrow$ Matcher($queryTemplate, enrolledTemplate$)
3:     if $matchingScore >= threshold$ then
4:         return match
5:     else
6:         return non-match
```

A comparison between an input new template and a user's enrolled templates is referred to as *verification*, also known as one-to-one comparison. This is typically used for deciding if the requested input is genuine or not, and if a person who wants to authenticate a claimed identity is the correct person. An *identification* or one-to-many comparison is a process where a user's input template is compared against all templates stored in a database in order to match the user's identity [3]. During the identification process, a user does not need to claim his identity. This serves not only as an identity recognition method, but can prevent users from claiming multiple identities. Figure 2.5 presents the verification procedure, while Figure 2.6 shows the identification procedure.

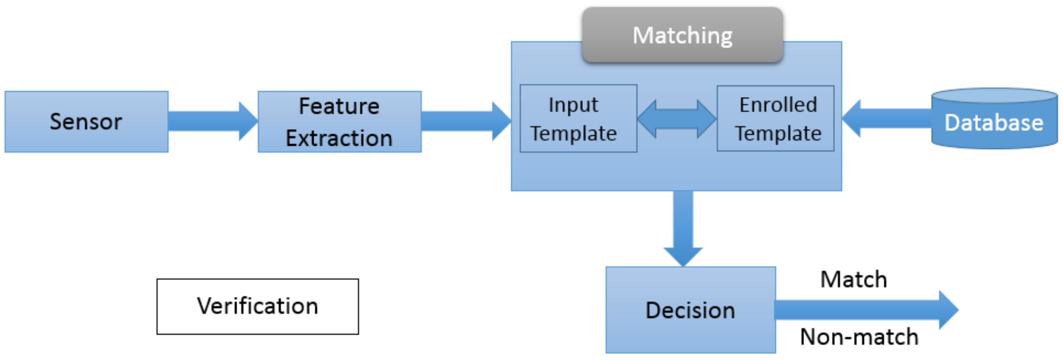


Figure 2.5: Verification procedure

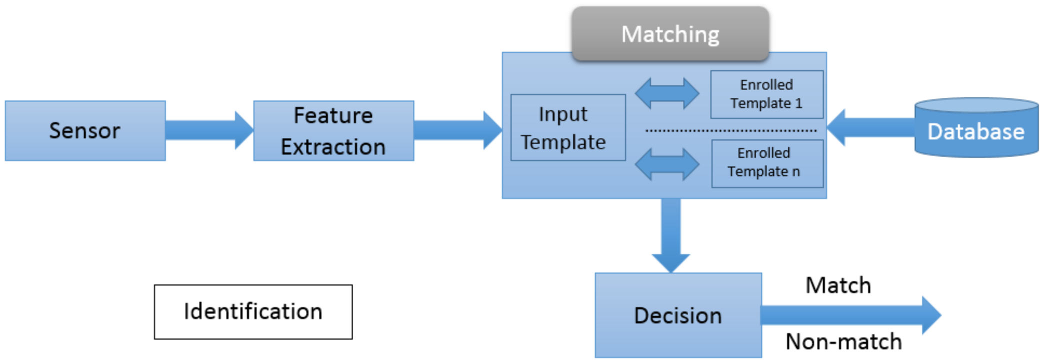


Figure 2.6: Identification procedure

Due to the nature of biometrics, it is considered as almost impossible, for two templates under the same identity to be completely the same [8]. This occurs as a result of the different input conditions (presentation artifacts, as pose), sensor conditions (thermal noise, dirt, sensor ageing), ambient conditions (room light), alterations in the biometric characteristic (ageing, health-related changes, surgery) [23]. This simply means that there will be noticeable changes within templates from the same user (*intra-class variations*), and even larger variations within templates coming from different users (*inter-class variations*). However, it is worthwhile to mention that in case where two templates are matched with perfect similarity, a “red flag”should be raised, indicating a replay attack.

For the evaluation of the performance of biometrics, or more specifically of biometric matchers, the following metrics have been introduced. There are usually two types of errors that a biometric matcher can make, *false match* and *false non-match*. A false match occurs when two templates from different users are incorrectly considered as samples from the same user and the system results in a match (also known as *false positive*). A false non-match occurs when two templates from the same user are incorrectly considered samples from different users, which results in a non-match (also known as *false negative*) [14].

The accuracy of the biometric system is measured by the comparisons between false acceptance and true acceptance attempts with the assistance of the following performance indicators [25]:

- False Acceptance Rate (FAR) or False Matching Rate (FMR): The percentage of false acceptance attempts
- False Rejection Rate (FRR) or False Non-Matching Rate ($FNMR$): The percentage of false rejection attempts.
- Genuine Acceptance Rate (GAR): The percentage of true acceptance attempts that can also be calculated as $1 - FRR$
- Equal Error Rate (EER): The point where FAR and FRR are equal. A lower EER value usually indicates a system with better performance.

- Receiver Operating Characteristic (ROC) curve: The trade-off between FRR and FAR rates for various threshold values, presented in a graph. A ROC curve with normal-deviation scaling can be seen in Figure 2.7. This specific ROC representation is called Detection Error Trade-Off (DET). DET curves are used for convenience, since the curves are more linear than ROC curves and the information is shown more clearly.
- Failure to Capture Rate ($FTCR$) or Failure to Acquire Rate($FTAR$): It indicates the percentage of the failure attempts, usually due to the insufficient available features after feature extraction
- Failure to Quantize Rate($FTQR$): A new metric that we introduce in this thesis and it indicates the percentage of failure attempts due to insufficient features, after a quantization of a biometric template

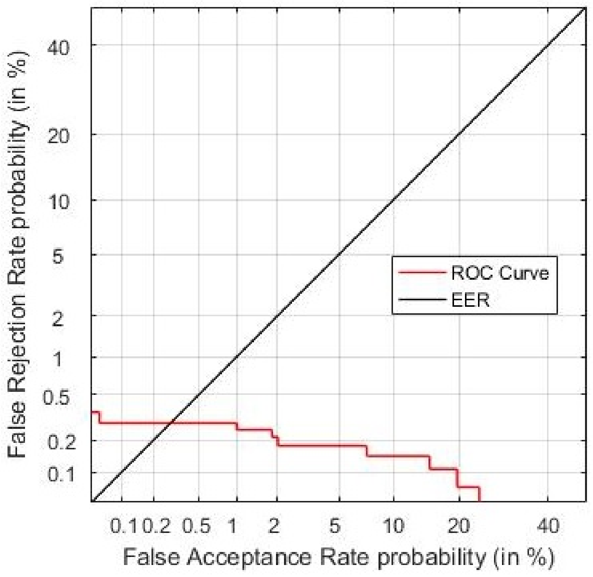


Figure 2.7: ROC curve for trade-offs between FAR and FRR

2.4 Fingerprint Matching

It is underlined that the feature extraction process is considered to be outside the scope of this thesis, and due to that the most popular method is described very briefly. Initially, when the sample is collected in gray-scale image mode, it is enhanced and binarized to improve the quality of the image. The image is thinned in order to reduce the size of the ridge to one pixel. Following the procedure where the image is thinned, minutiae can be detected easily, by simply comparing each pixel with its neighbours.

There are several different methods to match two fingerprint templates and they can be summarized into three main classes: i) correlation-based matching, ii) ridge feature-based matching, iii) minutiae-based matching. These methods are described in [22], where in correlation-based matching, the two templates are compared in an image-focused mode. The input template is rotated and translated in order to align with the enrolled template, and the similarity between the two images is calculated by cross-correlation. This method introduces some problematic issues, since distortion influences heavily the two images at global level and is computationally expensive when compared to minutiae-based matching algorithms. Ridge feature-based matching is another image processing technique, where the matching is based on the location and direction of ridge lines, especially around the core point of the fingerprint. The most widely used technique is the minutiae-based matching, and it is this one that we will utilize in this research.

On a minutiae-based matching, minutiae that are extracted from both templates, they are presented as points into a two-dimensional coordinate system. Along with the two coordinates $\{x, y\}$, the minutiae angle or orientation ϑ and the minutiae type (ridge ending or bifurcation) t can be used. Usually the minutia type is omitted, and the minutia pattern consists of $m = \{x, y, \vartheta\}$. The minutiae-based matching can be further classified into two different methods, the *global structure matching* and the *local structure matching*. In global structure matching, before performing the comparison between minutiae position and rotation, the two templates must be aligned appropriately (rotated and translated). In local structure matching, minutiae can be described without transformation, by comparing them with other minutiae in their local neighbourhood. Further analysis and experimentation between these two methods can be found in [17].

Chapter 3

Template Protection

In this chapter, we present a literature review of the security and privacy perspectives for biometrics, including the template protection techniques, properties, requirements and challenges. Accordingly, we perform a deeper analysis on the fuzzy vault schemes that are useful for the research performed during this thesis.

3.1 Introduction

Biometric deployments have become a natural fit for many applications. For that reason, the security of biometrics is of utter importance. Biometric systems, as identity management and authentication mechanisms, need to ensure the security of their modules and in the same time protect the privacy of their users. This is a crucial step for the public acceptance of biometric designs. The security of biometric systems usually revolves around the ensurance that a genuine user will be able to authenticate, while an impostor will not, while as the stored templates in the database will be protected from illegal access. The last statement also introduces the privacy requirements on biometrics, due to the particular nature of biometrics carrying much information about the person that cannot be easily revoked when compromised. It is important to emphasize that security and privacy requirements are not the same, and they need a different approach. An accurate system must minimize both False Acceptance Rate and False Rejection Rate, thus achieving the best performance possible.

3.2 Requirements

In order to secure biometric systems and protect users' privacy we need to analyse and model the requirements for the system's protection. These requirements should be satisfied in order to consider a system as secure and privacy-aware. In this section, we will summarize the security and privacy requirements, as described by Breebaart et al. in [4].

3.2.1 Security Requirements

The security of a biometric system usually revolves three main fundamental requirements: *confidentiality, integrity, availability.* Other requirements such as *renewability* and *revocability* have been introduced particularly for privacy-friendly biometric deployments [4].

- *Confidentiality* ensures that no information will be revealed about the stored and transmitted data of the system. For a biometric system, this means that stored templates and transmission between modules will be protected from unauthorized access, listening and modification. This can be achieved through cryptographic operations and network security techniques.
- *Integrity* ensures the accuracy of stored and transmitted data. This implies that no-one will be able to modify the data and can be achieved by utilizing cryptographic techniques.
- Availability requires that every part of the system will be functioning properly and will be available when this is necessary. Security mechanisms and controls should be established in order to protect the system from physical and network attacks or accidental failures.
- *Renewability and Revocability* require that a biometric template can be revoked and substituted with a new one in case of compromise. This cannot be achieved with an unprotected template, since biometric traits cannot be changed and this creates a vulnerability for a lifetime. Hence, an enrolled template should be transformed.

3.2.2 Privacy Requirements

Privacy refers to the ability of a user to control by whom and how his personal information is collected and used. The acquirement of stored templates in a biometric database poses a great privacy threat to the user. A biometric reference can contain personal information, in many cases sensitive, that can reveal a lot of information about the person's identity or body condition, such as health related information. Acquired templates from different databases can be cross-linked to reveal information about the user's preferences and behaviour. We present three types of privacy requirements, *Identity Privacy*, *Irreversibility* and *Unlinkability*, as described in the literature[4].

- *Identity Privacy* ensures that additional information related to the identity of the user enrolled and stored along with biometric templates are protected effectively, so it doesn't reveal any information about the user's identity and it cannot be used to link the user to other applications.
- *Irreversibility* requires a stored template to be transformed in a way that it will be easy to transform a template one-way, but

computationally infeasible to reverse it to its original state, so if an adversary acquires a stored template, he will not be able to recover the original one.

- *Unlinkability* implies that two biometric samples acquired from different sources cannot be linked, even if they belong to the same person.

3.3 Template Protection Schemes

In order to satisfy the above requirements, protection methods need to be devised in such a way that biometric templates are protected effectively against possible attacks. The biggest challenge during the protection of biometrics is that, as already mentioned, due to intra-class variations, two biometric references can never be completely alike. This means that traditional cryptographic schemes are not enough, as a small variation to the input data will result in huge variations in the ciphertext (a property known as *diffusion*). Hence, new cryptographic methods need to be introduced that can encrypt and decrypt biometric data in an error-tolerant way [21]. These cryptographic methods are widely known as *template protection schemes*. A classification of template protection schemes is illustrated in Figure 3.1 as presented by Jain et al. in [11]). Secondly, we describe briefly each category and template protection schemes, as those can be found in the literature [29].

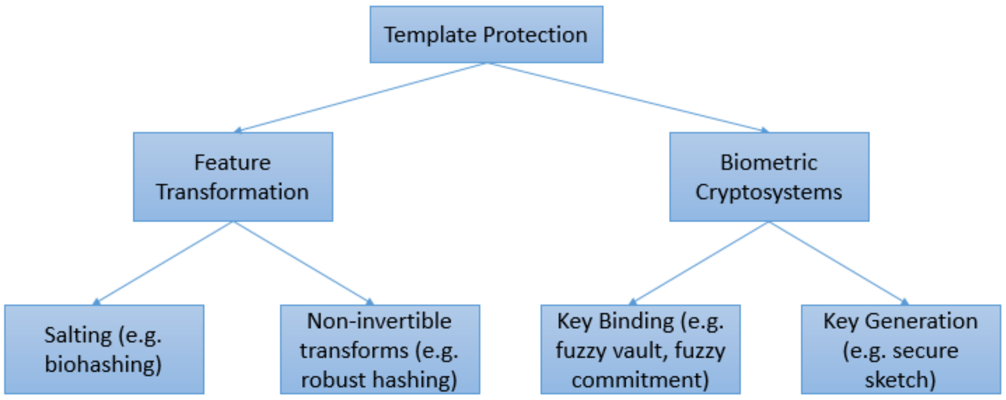

Template Protection
Feature Transformation
Biometric Cryptosystems
Salting (e.g. biohashing)
Non-invertible transforms (e.g. robust hashing)
Key Binding (e.g. fuzzy vault, fuzzy commitment)
Key Generation (e.g. secure sketch)

Figure 3.1: Classification of template protection schemes

Feature transformation schemes, also known as *cancelable biometrics* use a function to transform the enrolled templates, usually with the assistance of a key, which is used as a parameter. During matching, the input template is transformed as well with the same function and parameters, and the matching is performed in the transformed domain. *Salting* technique uses a function which is invertible if the key is known, hence the security of the scheme lies in the protection of the parameters. *Non-invertible transforms* use one-way functions that are hard to invert in terms of computational complexity, even when the parameters are known [6].

Biometric cryptosystems consist of schemes developed to either secure a digital secret, in practice a cryptographic key, by encrypting it with biometric features (*key-binding cryptosystems*), or generating a digital secret directly from biometric features (*key-generation cryptosystems*) [37]. Biometric cryptosystems use additional information known as *helper data* to validate the secret or assist in the alignment of the two templates [35]. This additional information should not reveal anything but as few information as possible regarding the stored template or the protected secret. Cryptosystems rely heavily on the use of error-correcting codes to manage intra-user variations [34].

3.4 Fuzzy Vault

3.4.1 Definition

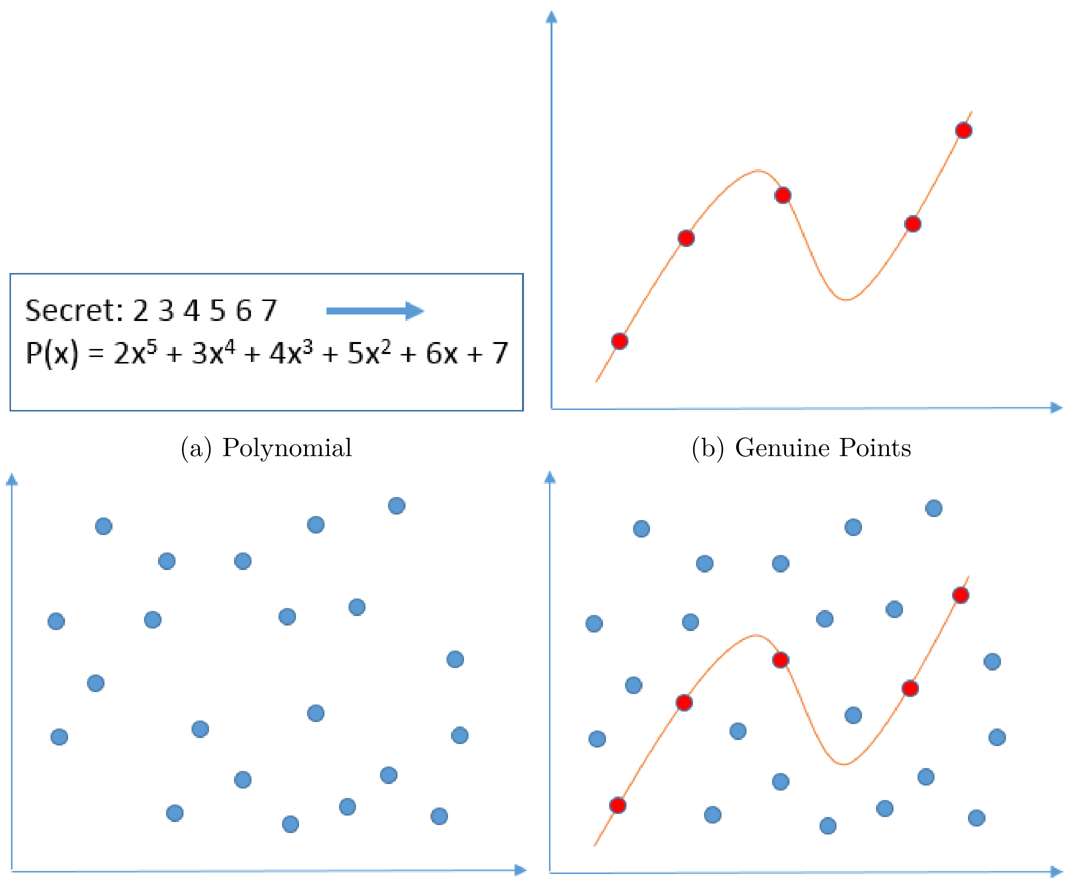
In this section the fuzzy vault key-binding template protection scheme is analysed, as this have been introduced by Juels and Sudan in [18]. This paper was largely based on the previous work of Juels and Wattenberg[19] on another key-binding template protection scheme called fuzzy commitment. Fuzzy commitment protects a secret value k under a key x. In order to unlock the secret k, a user will require any key x' which is close to x under some metric, using an error-correcting code. Fuzzy commitment can be described briefly in the following steps. Let F be a finite field and C a set of codewords for an error-correcting code that lie in F. The user selects randomly a codeword c and computes $\delta = c - x \in F^n$. Let $y = h(c)$ for a one-way function h. The pair (δ,y) consists the commitment pair. In order to decommit, the user computes $\delta + x'$ and decodes to the nearest codeword c'. If $h(c') = y$ the decommitment is successful.

A drawback of this scheme is that it does not support order invariance, thus making it not suitable for modalities such as minutiae-based fingerprints. Fuzzy vault is able to mitigate this issue. Assuming that we have to protect a secret k under a set A. We select a finite field $F(2^m)$ and encode the secret k to a polynomial p as its coefficients of m-bit size. We project every element of set A on to points lying on the polynomial p. Then, we create a number of random points that do not lie on to the polynomial as noise, named chaff points. The purpose of these chaff points is to protect the set of points A from an adversary who cannot distinguish genuine points from chaffs. Hence, this random noise provides the security of the fuzzy vault scheme. The pairs of $\{x, p(x)\}$ for both genuine and chaff points constitute a fuzzy vault and it is illustrated in Figure 3.2.

If another user wishes to unlock the vault, he would have to provide a set of points B that overlaps substantially with set A. If this is true, he will be able to distinguish some of the genuine points of the vault from the chaff points with great probability. In order to recover the secret k that constitutes the coefficients of the polynomial p, the user will need to recover p from a set of genuine elements and projections of these elements on p. This is considered to be a problem known as polynomial reconstruction or as polynomial interpolation, a process of finding a polynomial, given a dataset which belongs to. The original polynomial can be reconstructed successfully if given at least $d + 1$ correct points, where d is its degree, by methods such as Lagrange interpolation or simply solving a system of linear equations with the coefficients as unknown variables. For that reason, it is proposed a special use of Reed-Solomon error-correcting code to reconstruct the polynomial. Noting that any linear error-correcting code could be used for that purpose, but without actually implementing it.

Since Juels and Sudan work does not deal with the application of fuzzy vaults to fingerprints, it does not propose any specific method for solving the alignment problem in this specific biometric modality.

Further details about the use of Reed-Solomon error correction, along with locking and unlocking algorithms can be found in [18].



(a) Polynomial (b) Genuine Points

(c) Chaff Points (d) Fuzzy Vault

Figure 3.2: The creation of a fuzzy vault. A secret is encoded into a polynomial as its coefficients. Genuine minutiae are projected onto the polynomial. Random chaff points are created and mixed with genuine in order to create the vault.

3.4.2 Security Analysis

We investigate the security performance of the fuzzy vault, under a threat model where an attacker has gained access to the database, and hence to the actual stored fuzzy vaults. A successful attack to the vault, in order to unlock the secret, requires the adversary to locate $d+1$ genuine points, where d is the degree of the polynomial out of size n of the vault. A full brute-force attack, in a worst-case complexity, without any information about the points, for instance random guessing, will require $\binom{n}{d+1}$ combinations, namely all combinations inside the vault. Assuming that g is the number of genuine points inside the vault, then only $\binom{g}{d+1}$ combinations can unlock the vault. In order for the adversary to find a correct combination of points, an average time of $\binom{n}{d+1}/\binom{g}{d+1}$ will be required, where a correct combination can be guessed with probability $\binom{g}{d+1}/\binom{n}{d+1}$.

3.4.3 Attacks

From security and privacy perspective, the database is probably the most appealing module for attacking, since it is the place where all biometric templates are stored. In case the templates are stored unprotected, the original features will be exposed to the attacker. A template protection scheme can protect the template to some extent and satisfy some of the security and privacy requirements, but still the database is “open”to new kinds of attacks. Since the fuzzy vault operations are performed between the database and the matching procedures, vault-oriented attack vectors are introduced between these two modules. Below, we summarize the three main classes of attacks found in [32], regarding specifically fuzzy vaults:

1. **Record Multiplicity Attacks**
   Assuming that a user’s biometric reference is stored in multiple databases using a fuzzy vault scheme with the same secret key encoded in it. Then the genuine points will be evaluated under the same polynomial, and thus, they will have the same x, y values in every database. If an adversary acquires stored fuzzy vaults from at least two different databases, he can correlate the vaults and distinguish the common pairs as genuine points and discard the rest as chaff points, and subsequently unlock the vault. This poses a privacy threat, since it does not satisfy the unlinkability requirement, and to mitigate this, fuzzy vault templates should not encode the same key with the same parameters into different databases.

2. **Key-Inversion Attacks**
   A fuzzy vault scheme protects both the biometric template and a secret key. Most attacks are focusing on retrieving the secret key by trying to identify genuine points inside the vault, but there might be occasions where an adversary has acquired the secret key by other means, and wants to retrieve the biometric template. If the adversary possesses the key, this indicates that he knows the correct

polynomial that has been used to encode the vault, and thus, he can identify easily the genuine points by checking which vault points lie on the polynomial and which are not.

3. **Blended Substitution Attacks**
   In blended substitution attacks, an adversary injects his own biometric features into a template belonging to someone else user's reference. If the adversary does not know the secret key by which the vault was encoded, he will have to use a polynomial key of his own. This implies that he will not be able to unlock the original key when he provides his own biometric reference to the sensor. However, he could trick the matcher into authenticating him if there is no other way to validate the recovered key. He could perform a denial of service attack, where the genuine user will not be able to authenticate, if he substitutes the original template with his own. This denial of service attack can expose the modified template quite easily. If the adversary possesses the original secret key, he can encode his features with the same polynomial, and thus both users will be able to decrypt the same key while the adversary remains undetected. This attack violates the requirement of *integrity* for a biometric system and in order to mitigate this, digital signatures can be used to sign the biometric templates.

3.4.4 Related Work

The authors of the fuzzy vault scheme described in [18] do not provide many details about a series of issues regarding the implementation of the scheme, such as the use of error-correcting codes and alignment. In this section, we describe the design choices and the results of some of the most popular implementations found in the literature. Uludag et al. in [36] introduce one of the first implementations of a fuzzy vault, without using error-correcting codes claiming that "it is difficult to achieve error-correction with biometric data"and "the polynomial reconstruction via error-correction has not been demonstrated in the state-of-the-art". Instead, they use a larger set of points than the one needed for polynomial reconstruction, trying all combinations between them using all possible subsets and specifically, subsets of 9 out of 12 points. A Cyclic Redundancy Code *(CRC)* is used to check the correct retrieved secret. They quantize their minutiae points into 16-bit strings by using only x, y coordinates and not minutiae orientation. The polynomial reconstruction is accomplished by utilizing Lagrange interpolation. They achieve in securing a 128-bit secret with 21% *FRR* and 0% *FAR*.

The work of Nandakumar et al. in [27] is largely based on the same implementation choices of [36]. In their design, they include minutiae orientation into their quantized points, minutiae selection based on quality, and an alignment technique based on flow curves. In their experiments, the authors also use more than one template for encoding and decoding (multiple enrollment and query templates). Nandakumar and Jain in [26] utilize a fuzzy vault scheme to encode multiple biometric templates consisting of different modalities into a single entity. Their results show that a biometric fusion of fingerprint minutiae and iriscode features inside a vault provide higher security along with better performance. In their experiments, they achieve a GAR of 98.2% and a FAR of 0.01%. Additionally, the work by Nagar et al. in [24] focuses on the use of minutiae descriptors for alignment. Minutiae descriptors consist of information regarding the positions (ridges orientation and frequency) around a minutiae upon a fingerprint image. Their experiments show that the proposed method reduces the FAR without influencing the genuine acceptance. These works use information about minutiae based on patterns upon the fingerprint image and not actual minutiae points for alignment. This implies that access to the original fingerprint image is required, since these helper data is extracted from the image, by applying image processing techniques.

Additionally, Yang and Verbauwhede in [38] introduce a minutiae-based alignment method using only minutiae points. Their method is based upon the selection of a reliable reference point and the points stored in the vault consist of minutiae information related to the reference point. Specifically, they store distances, position angles and direction differences between all minutiae and the reference point. Their reference point selection is based upon a similarity level between multiple templates. The best quality point based on the similarity level serves as the reference point. They achieved a GAR of 83% for a small database of 10 fingers, without mentioning the FAR. However, this work fails to address any security implications regarding the minutiae-based helper data usage.

Jeffers and Arakala in [15, 16] propose three methods for minutiae-based alignment, based on local structures of minutiae points. These methods include *Five Nearest Neighbour Based Structures*, *Voronoi Neighbours* and *Triangle Based Structures*. Five Nearest Neighbour Based Structures use Euclidean distances and angle differences between a minutiae point and its five nearest neighbours. Voronoi Neighbours work in the same way, except for using regions on Voronoi diagrams as nearest neighbours instead of points. The last method of Triangle Based Structures creates local triangles between three minutiae points, using their Euclidean distances and angle orientations. In all these suggestions, an algorithm is used to locate similar structures in an input template under some thresholds, and transform the input template accordingly, by using translation and rotation, before trying to unlock the vault. Although these methods provide satisfying results, they do not take into account the security implications either. The most significant issue on minutiae-based alignment methods is the fact that genuine minutiae can not only assist in the alignment, but can actually unlock the vault. Since the Triangle Based Structures method is the alignment method that we use in our implementation of the fuzzy vault, we provide a thorough analysis on the security implications of our proposed solution.

Finally, an interesting approach on creating non-random chaff points has been introduced by Nguyen et al. in [28]. Most implementations generate chaff points randomly, which is not always the best case in practice, because minutiae are not distributed uniformly on a fingerprint. Hence, an attacker could use statistical analysis to filter out some of the chaff points and reduce the size of a brute force attack.

Chapter 4

Design and Implementation

The implementations of secure biometric systems introduce trade-offs between performance and security [5]. In this thesis, we decide to implement and evaluate a secure biometric system based on the fuzzy vault template protection scheme using only minutiae information, addressing the design choices of the alignment and fuzzy vault and taking into account how these trade-offs influence performance, security and accuracy. The reason to implement a system using only minutiae information is so it would be possible to use the system with an unprotected fingerprint database containing minutiae only. This makes the system compatible with the international standard ISO/IEC 19794-2 which defines fingerprints stored in a "Finger Minutiae Record Format" [9].

The secure biometric system, described in this paper, is based on two distinct subsystems performing independently of each other, an alignment method and a template protection scheme. This chapter is divided in two main sections. The first part of this chapter focuses on our alignment proposal, based on triangle structures comprised of minutiae points as helper data. Limitations and security issues are discussed. Additionally, we present our fuzzy vault template protection scheme, addressing and analysing the design challenges and lessons learned during the procedures of implementation methods. These involve the polynomial reconstruction, the usage of larger subsets of points and the involved parameters, such as the number of genuine and chaff points, finite fields and various thresholds.

4.1 Alignment

The alignment of two templates is one of the most important aspects of a biometric scheme. In practice, when a user presents his fingerprints on the biometric sensor, the scanned template might be distorted and misaligned. Depending on the matching algorithm, the biometric system should ensure that the two templates are properly aligned before the feature extraction process, otherwise the matching will probably fail. The alignment can also be performed after the feature extraction procedure, as long as the input template is transformed accordingly, utilizing any additional information regarding the enrolled template. Nowadays, alignment is still considered to be a hard task to be implemented correctly during the matching of two unprotected templates. It is even more difficult to be performed under a cryptographically protected template scheme. This is mainly happening since the enrolled template is not available and the full matching procedure is performed in the protected domain.

Regarding this part, when the matching is performed in the protected domain under a template protection scheme, the use of additional information related to the protected template is necessary in order to align the query template. It is inevitable that this extra information will reveal some form of data about the protected template. Therefore, the alignment technique that is used should be in such a form that ensures that this additional information reveals as little information as possible. Furthermore, the security weaknesses of such a technique should be taken into account during the security analysis and the overall performance of the system. From now on, we refer to this additional information of the protected template as “helper data”.

Different approaches have been proposed in the literature, as discussed in Chapter 3.4.4, mostly between the use of actual minutiae points and other data being derived from the fingerprint image, without revealing information about the position of minutiae. The use of minutiae for alignment is easier in terms of implementation. However, it introduces security and privacy risks, since genuine minutiae can unlock the vault or reveal information about the user’s fingerprint. The use of other fingerprint image information does not reveal directly the positions of minutiae, but it cannot be easily applied in older biometric systems which do not have access to the original fingerprint images, and thus creates a backwards compatibility issue.

4.1.1 Problem Statement

The alignment method for our biometric cryptosystem is a minutiae-based alignment technique. Our proposed method uses minutiae coordinates and orientations as points, in order to create triangle structures upon a two-dimensional Cartesian coordinate system, as described in [15, 16]. We follow this method since it provides a simple and effective way of aligning fingerprint templates by using only minutiae information, without any other information about the actual fingerprint. Thus, it is not necessary for the system to use the corresponding fingerprint images at all, and avoid more complicated image processing techniques. Our proposed technique can be implemented in current biometric deployments. This is considered to be its greatest advantage, since in many schemes (such as those relying on the fingerprint template representation specified in the ISO/IEC 19794-2 standard [9]), the features are extracted and stored in a minutiae-based format, and the access to the fingerprint images is not available anymore. It is noted that this is extremely useful when a system has too many enrolled users and re-enrollment proves to be infeasible. The security limitations of this method have not been discussed by the authors [15, 16], hence we will try to approach this method from a security and privacy perspective.

4.1.2 Geometric Transformation

For the purposes of the alignment, we utilize two types of transformation: translation and rotation. Due to the fact that our fingerprint templates are collected by the same sensor device, their image size remains the same. For that reason, there is no need for scaling our templates under our studied scenario.

In terms of translation in geometry, we refer to the process of moving every point of a figure to the same distance and in the same direction. Let dx be the translation by dx units along the X axis, and dy the translation of dy units along the Y axis. For given dx and dy, we compute the translation of x and y coordinates of a point using Algorithm 2.

Algorithm 2 Translation

```
Input:$inputPoint\{x, y\}, dx, dy$
Output:$translatedPoint\{x, y\}$
1: $translatedPoint_x \leftarrow inputPoint_x$ - $dx$
2: $translatedPoint_y \leftarrow inputPoint_y$ - $dy$
```

Rotation refers to the motion of every point of a figure by a specified angle around a fixed centre, while the centre and its distance to every other point remain the same. Let $referencePoint_X$ and $referencePoint_Y$ be the coordinates of the given fixed centre, while $\delta\theta$ is the specified angle. In order to rotate a point to its new coordinates, we translate the figure to the origin of the two axes (0,0). Accordingly, we multiply each point's coordinates with the following rotation matrix to rotate it by $d\theta$ degrees counter-clockwise:

$$R = \begin{bmatrix} \cos d\theta & -\sin d\theta \\ \sin d\theta & \cos d\theta \end{bmatrix}$$

Finally, we translate each point back from the origin by the same distances. Rotation is implemented in Algorithm 3.

Algorithm 3 Rotation

```
Input:$inputMinutiae\{x, y, \theta\}, referencePoint\{x, y\}, d\theta$
Output:$rotatedMinutiae\{x', y', \theta'\}$
1: $\{x, y\} \leftarrow Translation(inputMinutiae\{x, y\}, -referencePoint_x, -referencePoint_y)$
2: if $d\theta <= 0$ then
3:     $d\theta \leftarrow 360 - d\theta$
4: $\theta' \leftarrow (\theta + d\theta) \mod 360$
5: $\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} \cos d\theta & -\sin d\theta \\ \sin d\theta & \cos d\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$
6: $\{x', y'\} \leftarrow Translation(\{x', y'\}, referencePoint_x, referencePoint_y)$
```

Figure 4.1 presents the transformation process for the unaligned templates.

The templates after rotation are illustrated in Figure 4.2. After the procedure of translation the form of the templates can be found in Figure 4.3, while Figure 4.4 shows the aligned templates.

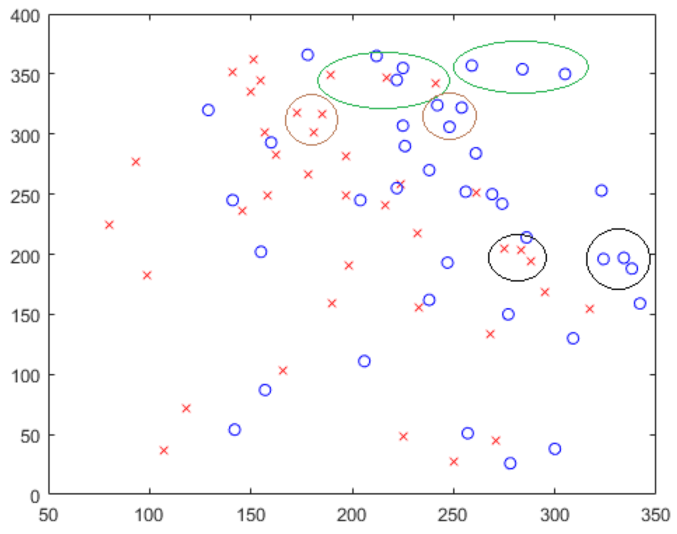


Figure 4.1: Unaligned templates

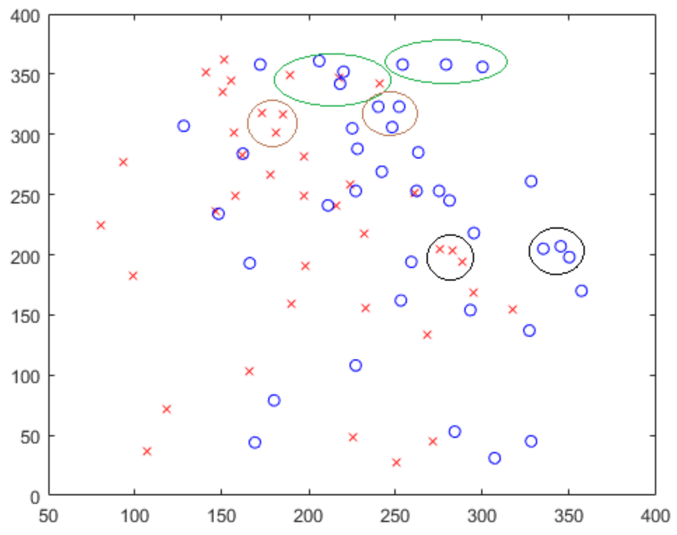


Figure 4.2: Templates after rotation

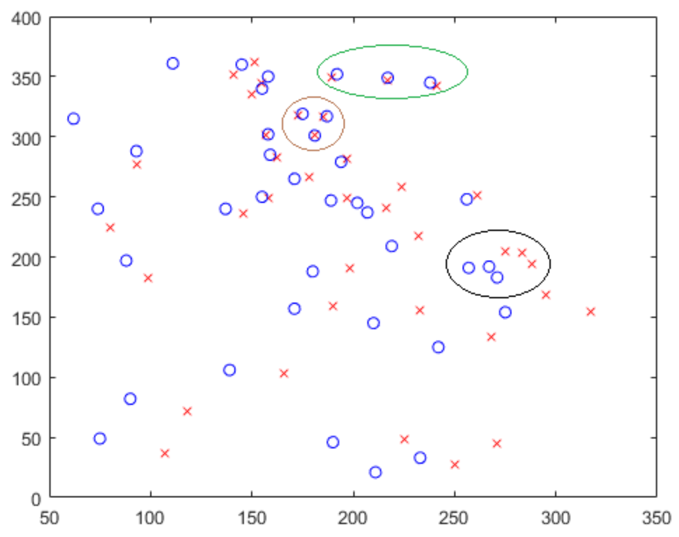


Figure 4.3: Templates after translation

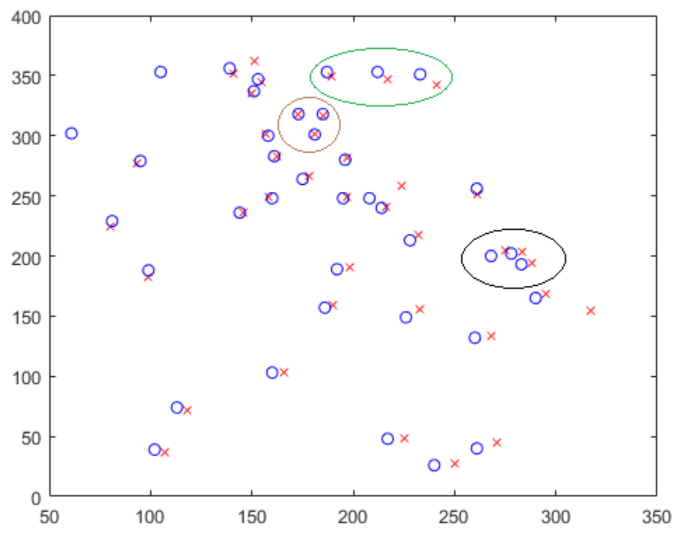


Figure 4.4: Aligned templates after rotation and translation

4.1.3 Proposed Method

Our alignment method based on triangle structures can be summarized as follows:

A *triangle minutiae structure* consists of a triplet of minutiae points which together form a triangle. Let $p1, p2, p3$ be the positions of the three minutiae points on the Cartesian coordinate system. Let $r1, r2, r3$ be the Euclidean distances between

$p1-p2$, $p2-p3$, $p1-p3$ points respectively. These three distances are the three sides of the triangle. $\theta1$,$\theta2$,$\theta3$ are the minutia orientation for each corresponding point. The external angles $\phi1$,$\phi2$,$\phi3$ of the three side vectors are also available and can be used for further matching. However, there are not explicitly needed and we are not using them in our current implementation. Figure 4.5 shows the triangle structure.

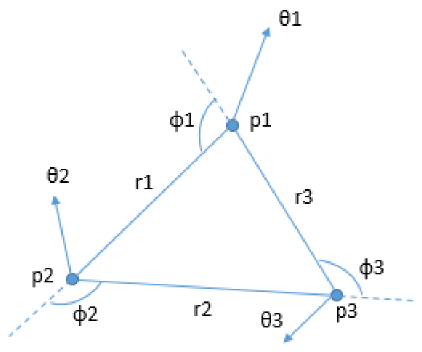


Figure 4.5: Triangle structure

The Euclidean distance between two points $p1(x1, y1)$ and $p2(x2, y2)$ in two dimensions is given by the Pythagorean Theorem as:

$$d(p1, p2) = \sqrt{(y1-y2)^2 + (x1-x2)^2}\,. \tag{4.1}$$

The external angles can be computed by law of cosines, where $\arccos x$ is the inverse cosine $\cos^{-1} x$ as:

$$\phi1 = 180^\circ - \arccos\frac{r1^2 + r3^2 - r2^2}{2 \cdot r1 \cdot r3}\,. \tag{4.2}$$

4.1.4 Triangle Structures Construction

The selection of the minutiae that we include in our triangle structures is performed under the following two criteria. The quality of the minutiae and secondly, a minimum distance (threshold) between all combinations of the selected minutiae. The algorithm's input consists of a number of minutiae in the format:

$$x, y, \theta, quality, type, quantizedMinutia \tag{4.3}$$

The format 4.3 is used in our system to portray each minutia, and its elements are as follows:

- x : value of x-coordinate (*abscissa*)
- y : value of y-coordinate (*ordinate*)
- θ : minutia orientation in $0^\circ - 359^\circ$ degrees
- *quality* : the quality of the corresponding minutia. A higher value represents a good quality minutia, whereas a lower value a bad quality one.

- *type* : the minutia type. 0 for ridge ending and 1 for bifurcation.
- *quantizedMinutia* : a concatenation of x, y, θ values after being quantized, in order to be used inside the fuzzy vault.

The final input of the algorithm is an integer n which refers to the number of triangle structures to be constructed under the specified minutiae. The algorithm can be briefly described in the following steps.

The minutiae are sorted according to their quality in a decreasing order. Each triangle needs three minutiae points. A major problem that we faced during this implementation is the fact that for security reasons (not disclosing information about the protected template), the minutiae used for alignment cannot be included in the fuzzy vault. This affects the performance of this scheme, since a smaller set of minutiae will reduce the positive matches of the fuzzy vault, and the minutiae removed from the matching are also from the highest quality ones. If good quality minutiae are used inside the vault, this will negatively affect the accuracy of the alignment, and vice-versa. Since we have to use good quality minutiae in both the alignment and the fuzzy vault, after experimentation, we have concluded that an ideal trade-off is to save the first 6 best quality minutiae for the vault and use the remaining for alignment, as long as the total number of minutiae is over 18. If the total number is less than 18, we can safely assume that this is not a sufficient number and consider it a Failure-to-Capture.

If the number of minutiae is not enough to construct n triangle structures, we reduce the number of triangles n as:

$$n = \lfloor \frac{sizeof(minutiae)}{3}) \rfloor \,. \tag{4.4}$$

The algorithm selects three minutiae points among the list of all available minutiae. These minutiae are well-separated, something that means that the lengths of the three sides of the triangle are not very close together, according to a minimum distance (threshold) that we named *minDistance*. Each triangle consists of the minutiae coordinates, their respective distances (sides of the triangle), the minutiae orientations, the external angles of each edge, and the minutiae types. The algorithm outputs a list of created triangles. Algorithm 4 summarizes the creation of triangles. A triangle structure format consists of:

$$\begin{gathered} m1_x, m1_y, m2_x, m2_y, m3_x, m3_y, r1, r2, r3, \\ m1_\theta, m2_\theta, m3_\theta, \phi 1, \phi 2, \phi 3, m1type, m2type, m3type \,, \end{gathered} \tag{4.5}$$

where m stands for minutia.

4.1.5 Retrieval of triangle structures

When a new query template needs to be aligned with a protected one, an algorithm tries to transform the new template in order to match the enrolled one, based on the triangle structures that were created previously, from the enrolled template. The

Algorithm 4 Create Triangle Structures

Input: $minutiae\{x, y, \theta\, quality, type\}, n, minDistance$

Output: $TriangleStructures(\leq n)$

```
1: $sortedMinutiae \leftarrow sortRows$(minutiae, quality) ▷ sort by quality in desc. order
2: if $sizeof(minutiae) > 18$ then
3:     $sortedMinutiae \leftarrow sortedMinutiae(7 : end)$ ▷ remove first 6 elements
4: if $sizeof(sortedMinutiae) < 3 \cdot n$ then
5:     $n \leftarrow \lfloor \frac{sizeof(minutiae)}{3}) \rfloor$
6: for $i = 1 \rightarrow n$ do
7:     $m1 \leftarrow sortedMinutiae(1)$; $p1 \leftarrow \{m1_x, m1_y\}$
8:     $m2 \leftarrow sortedMinutiae(2)$; $p2 \leftarrow \{m2_x, m2_y\}$
9:     $r1 \leftarrow EuclideanDistance(p1, p2)$
10:    $sortedMinutiae \leftarrow sortedMinutiae(3 : end)$
11:    $m3 \leftarrow sortedMinutiae(1)$; $p3 \leftarrow \{m3_x, m3_y\}$
12:    $r2 \leftarrow EuclideanDistance(p2, p3)$
13:    $r3 \leftarrow EuclideanDistance(p1, p3)$
14:    $tempCount \leftarrow 1$; $index \leftarrow 1$
15:    while $|r2-r1| < minDistance$ and $|r3-r1| < minDistance$ and $|r2-r3| < minDistance$ do
16:        if $tempCount \geq sizeof(sortedMinutiae)$ then
17:            $m3 \leftarrow sortedMinutiae(tempCount)$; $p3 \leftarrow \{m3_x, m3_y\}$
18:            $index \leftarrow tempCount$
19:            break while
20:        $m3 \leftarrow sortedMinutiae(tempCount)$
21:        $index \leftarrow tempCount$
22:        $tempCount \leftarrow tempCount + 1$
23:    $sortedMinutiae \leftarrow sortedMinutiae - \{sortedMinutiae(index)\}$ ▷ remove element located in $index$ position
24:    $r2 \leftarrow EuclideanDistance(p2, p3)$
25:    $r3 \leftarrow EuclideanDistance(p1, p3)$
26:    $\phi1 \leftarrow 180° - \arccos \frac{r1^2 + r3^2 - r2^2}{2 \cdot r1 \cdot r3}$
27:    $\phi2 \leftarrow 180° - \arccos \frac{r1^2 + r2^2 - r3^2}{2 \cdot r1 \cdot r2}$
28:    $\phi3 \leftarrow 180° - \arccos \frac{r2^2 + r3^2 - r1^2}{2 \cdot r2 \cdot r3}$
29:    $TriangleStructures(i) \leftarrow \{p1_x, p1_y, p2_x, p2_y, p3_x, p3_y, r1, r2, r3, m1_\theta, m2_\theta, m3_\theta, \phi1, \phi2, \phi3, m1type, m2type, m3type\}$
```

algorithm's input consists of a list of triangle structures (portrayed in 4.5 format), a list of minutiae derived from the query template (4.3 format), and two thresholds δr and $\delta\theta$ that define the minimum distance between triangle sides and minutiae orientation difference in order for two triangles to be considered as matched.

For every saved triangle structure, the algorithm tries to locate almost identical triangles (with small variations according to the two thresholds mentioned) based on relative differences among edges and orientations and matching on minutiae types. Firstly, for every minutiae of the query template, the algorithm attempts to find another minutiae, whose distance is close to a side of the saved triangle. If two minutiae are found, the algorithm searches for a third minutia whose distance to the two previously found minutiae is close to the other two sides of the triangle. If the relative distances are below the threshold δr, the algorithm has successfully found two almost identical triangles, based on their sides and proceeds by arranging the three minutiae points in the correct order according to the saved triangle and comparing minutiae orientations and types. If the relative differences between the three minutiae orientations of the two triangles are below the threshold $\delta\theta$, the algorithm continues by comparing the three minutiae types. If the three minutiae types match, a possible set of transformation helper data is extracted, and it consists of:

$$dx, dy, referencePoint_x, referencePoint_y, \delta\theta \,. \tag{4.6}$$

dx and dy (this constitute mean values of relative differences between x and y coordinates of the three points of the two matching triangles) serve as input for the Translation algorithm, whereas $referencePoint_x$, $referencePoint_y$ and $\delta\theta$ are used in Rotation. The first point of the newly found triangle is selected as the reference point, whereas $\delta\theta$ is based on the mean value of the three relative differences between minutiae orientations.

The algorithm removes duplicate values of helper data and outputs a list of possible transformations arranged according to 4.6.

4.1.6 Alignment Evaluation

In order to evaluate the performance of the considered alignment technique, we have developed a simple evaluation method by investigating the proximity between minutiae points after a successful transformation. Assume that we evaluate the alignment between a query template $T_Q = \{m_Q^1, ..., m_Q^N\}$ and an enrolled template $T_E = \{m_E^1, ..., m_E^M\}$. For every helper data available, we transform the query template accordingly, and compute the distance of every T_E point to every T_Q point. The minimum distance of every T_Q point to T_E points is stored in a vector as $minDists = \{d_Q E^1, ..., d_Q E^N\}$ and the k-th percentile of $minDists$ is computed. After we find the k-th percentile for every helper data available, we consider the percentiles' minimum value $minPercentile$ as the best possible transformation and discard the rest. For various values of a threshold θ, we count the number of points with distances $minDists$ less than θ divided by the total number of points, after applying the best possible transformation as previously found :

$$thetaPercentage = \frac{number\ of\ points\ with\ minDist < \theta}{number\ of\ total\ points} \cdot 100\,. \tag{4.7}$$

If no helper data is found, we assume $minPercentile = 150$ and $thetaPercentage = -1$.

By calculating the EER of $minPercentile$ for true and false attempts, we can evaluate the performance of our alignment method. By calculating the EER for various θ values, we can select the optimal θ based on the lowest EER value.

Algorithm 5 calculates FAR and FRR by comparing true and false attempts against a threshold. $|S|$ denotes the cardinality of the set S (number of elements in the set). Algorithm 6 calculates EER and $threshold$ values, using a dichotomic approach for efficiency, as well as FAR and FRR using Algorithm 5. In both algorithms, if true attempts are generally higher than false attempts polarity should be set to -1, otherwise polarity should be set to 1.

Algorithm 5 Compute_FAR_FRR

```
Input: $trueAttempts, falseAttempts, threshold, polarity$
Output: $FAR, FRR$
1: if $polarity = 1$ then
2:     $FAR \leftarrow \frac{|falseAttempts \leq threshold|}{|falseAttempts|}$
3:     $FRR \leftarrow \frac{|trueAttempts > threshold|}{|trueAttempts|}$
4: else
5:     $FAR \leftarrow \frac{|falseAttempts \geq threshold|}{|falseAttempts|}$
6:     $FRR \leftarrow \frac{|trueAttempts < threshold|}{|trueAttempts|}$
```

4.2 Fuzzy Vault

In this section, we will present our fuzzy vault implementation. The first subsection describes the creation of the fuzzy vault by encoding a secret using fingerprint minutiae (protection process), while the second subsection is dedicated to the recovery of an encoded secret inside a vault (key release process). The system's parameters are also discussed. For experimentation purposes, we implement our fuzzy vault system in Matlab, by extending the capabilities of a fuzzy vault prototype found in [33].

4.2.1 Protection

For security purposes, the minutiae being used for alignment are excluded from the process and they are not stored inside the vault. During the protection process, a

Algorithm 6 Compute_Threshold_FAR_FRR_EER

```
Input: $trueAttempts, falseAttempts, polarity$
Output:$threshold, FAR, FRR, EER$
1: $tolerance \leftarrow 10^{-4}$
2: $allAttempts = \begin{bmatrix} trueAttempts \\ falseAttempts \end{bmatrix}$    ▷ merge all attempts into a column vector
3: $minThreshold \leftarrow min(allAttempts)$
4: $maxThreshold \leftarrow max(allAttempts)$
5: $FAR \leftarrow 0$
6: $FRR \leftarrow 1$
7: $iteration \leftarrow 0$
8: while $|FAR - FRR| > tolerance$ & $iteration < 60$ do
9:     $iteration \leftarrow iteration + 1$
10:    $threshold \leftarrow \frac{(minThreshold + maxThreshold)}{2}$
11:    $FAR, FRR \leftarrow Compute\_FAR\_FRR(trueAttempts, falseAttempts, threshold, polarity)$
12:    if $polarity = -1$ then
13:        if $FAR > FRR$ then
14:            $minThreshold \leftarrow threshold$
15:        else if $FRR > FAR$ then
16:            $maxThreshold \leftarrow threshold$
17:        else
18:            $EER \leftarrow FAR$
19:            $break$
20:    else
21:        if $FAR > FRR$ then
22:            $maxThreshold \leftarrow threshold$
23:        else if $FRR > FAR$ then
24:            $minThreshold \leftarrow threshold$
25:        else
26:            $EER \leftarrow FAR$
27:            $break$
28:    $EER \leftarrow \frac{FAR + FRR}{2}$
```

polynomial is generated by using the elements of a secret value as the polynomial coefficients. Every minutiae point is projected onto the polynomial. Random chaff points that do not lie on the polynomial, are also created. These two sets of points are shuffled, creating a fuzzy vault. This procedure is explained in the following steps more thoroughly:

Polynomial Generation

The size of the secret value that needs to be protected, depends on the finite field and the polynomial degree that is used. A finite field $F(2^m)$ is selected and all calculations are performed over this field. Our implementation uses a d degree polynomial to encode a secret value consisting of $d+1$ elements over $F(2^m)$. Let $S = [c_1 c_2 ... c_{d+1}]$ be the secret value, where $c_x \in F(2^m)$ for all x. The length of the encoded secret can be calculated in bits as:

$$length(S) = (d+1) \cdot m \,. \tag{4.8}$$

Each element of the secret value is used as a coefficient of a polynomial, as:

$$p(x) = c_1 \cdot x^d + c_2 \cdot x^{d-1} + ... + c_{d+1} \,. \tag{4.9}$$

Scaling and Quantization

Each minutia consists of its {x,y} coordinates, orientation and type. These values need to be encoded into a single m-bit representation. To achieve this, each value needs to be quantized, in order to reduce its size. Since we use a fixed $F(2^{16})$ finite field in our implementation, we encode minutiae into 16-bit strings by quantizing x, y, θ values into 6,5,5 bit size values respectively. We avoid the usage of minutia type into the vault.

Due to the fact that minutiae are not scattered across a fingerprint image uniformly, for each minutia, x and y values are scaled, in order to distribute minutiae over the whole range of the fingerprint image. X and y values are scaled as:

$$x_{scaled} = round(x \cdot \frac{2^{length(x)} - 1}{x_{max} - 1}) \tag{4.10}$$

$$y_{scaled} = round(y \cdot \frac{2^{length(y)} - 1}{y_{max} - 1}) \,, \tag{4.11}$$

where $length(x)$ and $length(y)$ is the size of x and y values in bits, and x_{max} and y_{max} are the maximum sizes of x and y values in pixels.

After scaling, depending on the size of x,y and θ, each value is quantized as:

$$l = \lfloor \frac{r}{2^n} \rfloor \,, \tag{4.12}$$

where r is the value of x_{scaled},y_{scaled},θ respectively, and n is the number of bits to remove. n can be calculated as follows:

$$n_x = length(x) - 6 \tag{4.13}$$

$$n_y = length(y) - 5 \tag{4.14}$$

$$n_\theta = length(\theta) - 5\,. \tag{4.15}$$

After quantization, x is represented in 6 bits, y in 5 bits, and θ in 5 bits respectively. These three values are concatenated into one, in order to form a 16-bit representation of a minutia:

$$m = l_x|l_y|l_\theta\,. \tag{4.16}$$

The decimal representation of this value is an integer between 0 and 65535.

Point Projection and Chaff Creation

After quantizing and concatenating all minutiae, a number (based on a *genuineNumber* variable) of minutiae is selected to be used inside the vault. Genuine minutiae are sorted according to quality, in a descending order, and the first *genuineNumber* minutiae are selected. Each quantized minutia (which we will refer to as m_q) is projected onto the polynomial p as:

$$p(m_q) = c_1 \cdot m_q + c_2 \cdot m_q + ... + c_{d+1} \cdot m_q\,. \tag{4.17}$$

The genuine point list is comprised of:

$$GenuinePoints = \begin{bmatrix} m_{q1} & p(m_{q1}) \\ \dots & \dots \\ m_{q_{genuineNumber}} & p(m_{q_{genuineNumber}}) \end{bmatrix}. \tag{4.18}$$

Next, a number (based on a *chaffNumber* variable) of random chaff points and fake projections of them, are created. Each chaff point needs to be distanced at least *minDist* pixels from any other genuine and chaff point. The reason to keep a distance from chaff points as well, is to avoid the revelation of genuine points based on their distance, from an adversary who knows the value of the *minDist* variable.

$$ChaffPoints = \begin{bmatrix} chaffX_1 & chaffY_1 \\ \dots & \dots \\ chaffX_{chaffNumber} & chaffY_{chaffNumber} \end{bmatrix}. \tag{4.19}$$

Finally, GenuinePoints and ChaffPoints are merged and the rows of the new matrix are shuffled. The final matrix constitutes a fuzzy vault.

4.2.2 Key Release

During key release, the system loads a fuzzy vault matrix and partitions the concatenated input values to retrieve the original potential quantized x, y, θ values. Then, it compares these values along with the input template's corresponding values and selects those vault's potential minutiae that are closer to the input template's minutiae, according to some thresholds. A larger subset of the vault's minutiae than the number needed for polynomial reconstruction is selected, in order to try all possible combinations between them and improve acceptance. For each possible combination of points, a polynomial is reconstructed based on these points and a candidate secret is retrieved. In order to verify if the retrieved secret is the correct or not, a hash digest of the secret can be stored separately, and verification is achieved by comparing the two hashes.

If the input minutiae number is less than a threshold $inputMinutiaeNumber$ or less than $d + 1$, we consider it as a Failure to Capture and we do not proceed in key release since there are not enough points to unlock the vault. If the number of minutiae of the input template is less than $d + 1$ after quantizing them (in case there are duplicate values after quantization), we consider it a Failure to Quantize, and abort. In such a case in a real life application, the system would ask the user to enter his fingerprint again in order to acquire a better sample.

Partitioning

During key release, the system firstly partitions the fuzzy vault's input values into separate x, y, θ values by reversing the quantization process that occurred during protection. The 16-bit quantized minutiae is partitioned into a 6-bit string(x), a 5-bit string(y) and a 5-bit string(θ). These bit strings are converted into decimals.

Distances and Subsets

The system tries to locate the vault's points that are closer to the input points. While $d + 1$ points are needed for reconstructing a polynomial of degree d, more than $d + 1$ points are selected in order to explore all combinations between them and increase the chances of a vault unlocking. The number of selected points depends on a $subsetsNumber$ variable. If the number of the input minutiae is less than $subsetsNumber$, then $subsetsNumber$ is reassigned as the number of the input minutiae.

The system tries to locate points according to a maximum distance of 1,1 and 3 for x,y and θ respectively. If the number of minutiae matched under these maximum distances is less than the correct one needed ($subsetsNumber$), the maximum distances are increased by 2, until the correct number is found. The corresponding vault indices of the closest points are stored as $tempVaultIndices$, after removing possible duplicate entries. Subsequently, these indices need to be ordered in an ascending order based on their distance to input points, in order to try the best candidate points first and avoid unnecessary computations when trying combinations among them. This is achieved by calculating the distance between all $tempVaultIndices$ points and input points. The distance between an input minutiae $m_q(x, y, \theta)$ and a vault minutiae $v_q(x, y, \theta)$ is calculated as follows:

$$dist = \sqrt{(m_q(y) - v_q(y))^2 + (m_q(x) - v_q(x))^2} + angleWeight \cdot \delta\theta \,, \tag{4.20}$$

where $angleWeight$ is a constant that we set to 0.2 and

$$\delta\theta = min(|m_q(\theta) - v_q(\theta)|, 360 - |m_q(\theta) - v_q(\theta)|) \,. \tag{4.21}$$

The *subsetsNumber* vault elements with the minimum *dist* value are selected for polynomial reconstruction. At least $d+1$ correct elements are needed in order to recover the correct secret. The total number of combinations among these elements is equal to

$$\frac{subsetsNumber!}{(d+1)!(subsetsNumber-(d+1))!}, \tag{4.22}$$

where ! denotes a factorial. For each combination of vault elements (inputs and outputs), a candidate polynomial is reconstructed. If the correct secret is retrieved, there is no need to search for further combinations.

Polynomial Reconstruction

A combination of $d+1$ points can reconstruct a polynomial of degree d over a field F^q. This can be achieved by solving a system of $d+1$ equations with $d+1$ unknown variables, a method known as a direct method of interpolation [31]. Each equation is essentially the polynomial p with fixed coefficients as constants and different input m_q and projection $p(m_q)$ values according to each point in the set. We assume that our set of points consists of:

$$CandidatePoints = \begin{bmatrix} m_1 & p(m_1) \\ \dots & \dots \\ m_{d+1} & p(m_{d+1}) \end{bmatrix}, \tag{4.23}$$

Then the system of $d+1$ equations to solve can be modelled as:

$$\begin{aligned} c_1 \cdot m_1^d + c_2 \cdot m_1^{d-1} + ... + c_{d+1} &= p(m_1) \\ \dots & \\ c_1 \cdot m_{d+1}^d + c_2 \cdot m_{d+1}^{d-1} + ... + c_{d+1} &= p(m_{d+1}) \end{aligned}. \tag{4.24}$$

Since all m_x and $p(m_x)$ values are known, this system can be solved. A simple way to solve this system is the matrix inversion method. Assume the coefficient matrix:

$$C = \begin{bmatrix} m_1^d & m_1^{d-1} & \dots & 1 \\ & \dots & & \\ m_{d+1}^d & m_{d+1}^{d-1} & \dots & 1 \end{bmatrix} \tag{4.25}$$

and the right side of the equations matrix:

$$R = \begin{bmatrix} p(m_1) \\ \dots \\ p(m_{d+1}) \end{bmatrix}. \tag{4.26}$$

By multiplying R with the inverse matrix C^{-1}, we solve the system and acquire the unknown c_x coefficients as secret S:

$$S_{column} = C^{-1} \cdot R = \begin{bmatrix} c1 \\ c2 \\ ... \\ c_{d+1} \end{bmatrix} . \tag{4.27}$$

The transposition of S_{column} constitutes the encoded secret S:

$$S = S^T_{column} = [c_1 c_2 ... c_{d+1}] . \tag{4.28}$$

Chapter 5

Results

In order to evaluate the performance of the considered template protection scheme, we have implemented an alignment and a fuzzy vault simulator in Matlab, as described in the previous chapter. In this chapter, we present the experiments we performed on our biometric system and the results of those experiments under different parameters. We also describe the minutiae database and the system specifications of the machine we used to run the experiments.

5.1 Experimental Setup

All experiments were performed on a Dell Inspiron 5558 laptop with: CPU Intel Core i3-4005U 1.70 GHz, 12 GB of RAM memory, 500 GB hard disk drive and Matlab R2016a 64-bit, on a Windows 8.1 64-bit system. No extra Matlab Toolboxes were needed, except for the default ones.

5.2 Database and Evaluation Protocol

In our experiments, we used a minutiae database provided by Kayaoglu et al.[20]. We used the FVC2002DB1A database found in [1], which provides fingerprint minutiae extracted from the FVC2002 (Second International Competition for Fingerprint Verification Algorithms) database [2]. The FVC2002-DB1 database was created along with three databases, for the purposes of an international competition for fingerprint verification algorithms in 2002 and is one of the most widely used databases in literature.

The DB1 database that we used consists of fingerprint images which were collected using an optical sensor "TouchView II" by Identix. The image size of each fingerprint is 388x374 (142 Kpixels). The database contains 8 fingerprint impressions from the same finger of 100 users (800 fingerprint impressions overall). Impressions 1,2,7,8 are considered a sample of good quality, whereas impressions 3,4,5,6 are considered a bad quality sample, since the users were asked to provide a distorted and misaligned fingerprint on the sensor. The average number of minutiae for each template is 39.1 minutiae, with a minimum value of 9 and a max of 92. The minutiae extracted from this database are stored in "Finger Minutiae Record Format" as defined by international standard ISO/IEC 19794-2 Information technology - Biometric data interchange formats - Part 2: Finger minutiae data [9].

Our evaluation protocol is based on the following matching comparisons between samples to compute FNMR and FMR rates:

- In the multi-enrollment experiment, sample 8 of each user is matched against sample 1 of the same user to compute FRR for single enrollment. Sample 8 is considered the query template and sample 1 the enrolled one. In the case of multiple enrollment, samples 1 and 2 are used as enrolled templates for an experiment of two enrolled templates, and samples 1, 2 and 7 are used as enrolled templates for an experiment of three enrolled templates. Sample 8 is used as the query template in all cases.

- In the multi-enrollment experiment, sample 1 of each user is matched against sample 1 of the next user in the database to compute FAR for single enrollment. Sample 1 of the next user is considered the query template and sample 1 of the current user, the enrolled one. In the case of multiple enrollment, samples 1 and 2 of the current user are used as enrolled templates for an experiment of two enrolled templates, and samples 1, 2 and 7 of the current user are used as enrolled templates for an experiment of three enrolled templates. Sample 1 of the next user in the database is used as the query template in all cases.
- In the multi-query experiment, to compute FRR, sample 1 of each user is used as the enrolled template against samples 2, 7, 8 of the same user for experiments of 1, 2 and 3 query templates respectively. To compute FAR, sample 1 of each user is used as the enrolled template against samples 2, 7, 8 of the next user in the database. In both FRR and FAR experiments, template 1 is used as query in a 1-query experiment, templates 2 and 7 in a 2-query experiment and templates 2, 7, 8 in a 3-query experiment.

5.3 Alignment Evaluation

With the evaluation method developed in Chapter 4.1.6, we evaluate our alignment method for specific parameters. We chose 4 as $trianglesNumber$ and 20 for δr and $\delta\theta$ thresholds. We use the 60-th percentile to illustrate under which value the 60% of the minimum distances between the minutiae of two matching templates fall. Figure 5.1 illustrates the histogram of these minimum values for genuine and false attempts, while Table 5.1 summarizes these results. In 87% of the genuine attempts, potential triangles as helper data was found. No helper data was found in 13% of the genuine attempts. This indicates that the matching will fail in these genuine attempts because the vault cannot be unlocked without proper alignment. In 59% of the false attempts no helper data was found. In only 41% of the false attempts, possible triangles were found. The value 150 in the histogram illustrates attempts with no helper data. Cases where no helper data were found are not taken into account in all results portrayed in this section.

Table 5.1: Alignment Performance

trianglesNumber	percentile	found GAR	found FMR
4	60	87%	59%

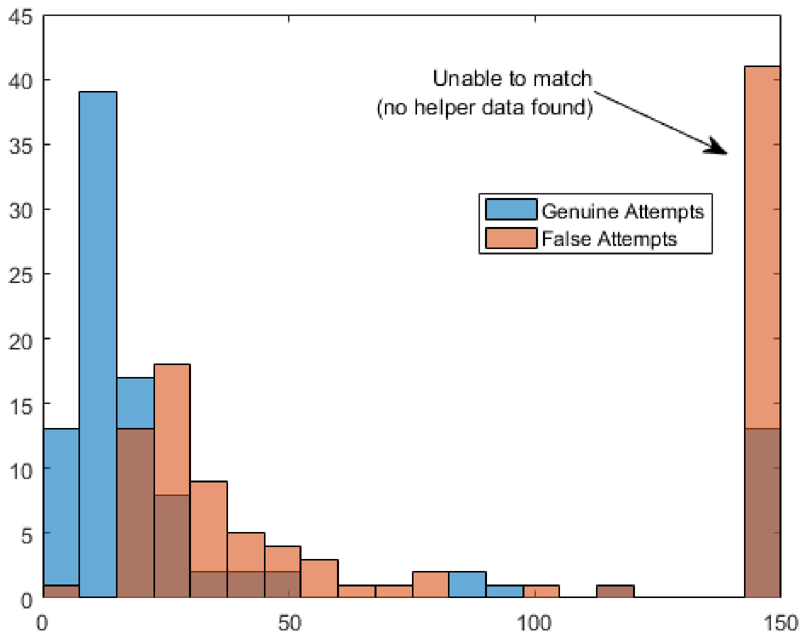


Figure 5.1: Histogram of minimum distance values under the 60th percentile

Table 5.2 illustrates FRR, FAR, EER, Threshold values of the previous results, based on Algorithms 5 and 6.

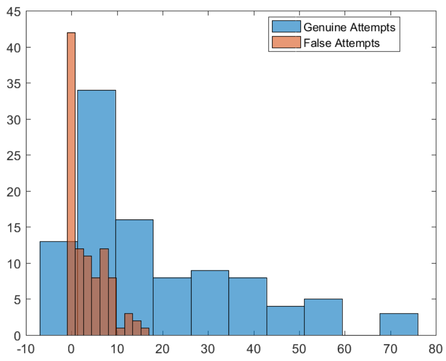
Table 5.2: FRR, FAR, EER, Threshold

FRR	FAR	EER	Threshold
0.2169	0.2034	0.2101	22.2036

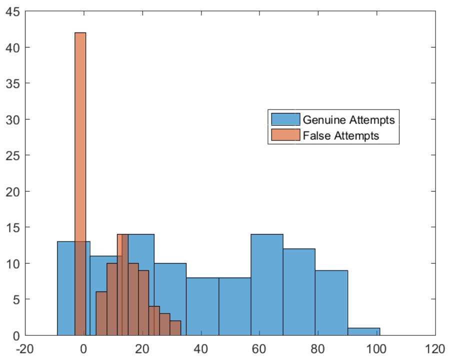
Table 5.3 shows FRR, FAR, EER, Threshold values of $ThetaPercentage$ for different θ values. It is shown that the best θ selection is $\theta = 20$ with an $EER = 0.1827$. Figure 5.2 shows histograms of $ThetaPercentage$ values under different θ values.

Table 5.3: FRR, FAR, EER, Threshold for different values of θ

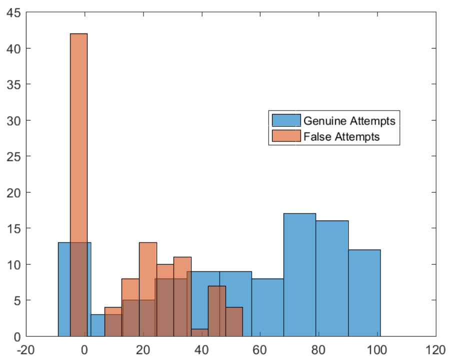
θ	FRR	FAR	EER	Threshold
5	0.2432	0.2444	0.2438	8.5106
10	0.2278	0.2245	0.2262	19.5122
15	0.2073	0.1964	0.2019	39.6552
20	0.1899	0.1754	0.1827	54.7619
25	0.2432	0.2321	0.2377	64.4444
30	0.2361	0.2321	0.2341	71.4286
35	0.2537	0.2364	0.2450	77.5510
40	0.2667	0.2767	0.2697	80.7692



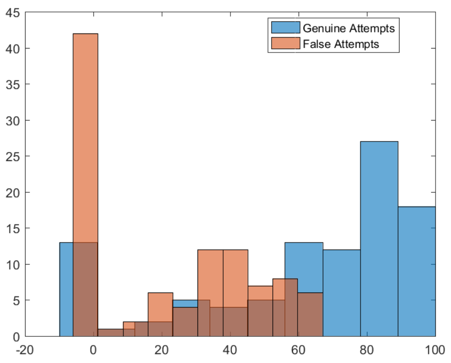
(a) $\theta = 5$



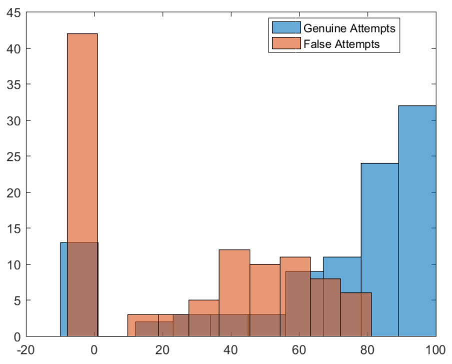
(b) $\theta = 10$



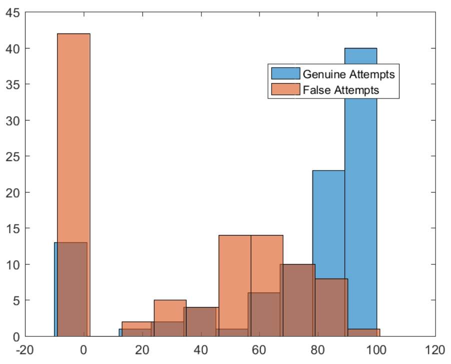
(c) $\theta = 15$



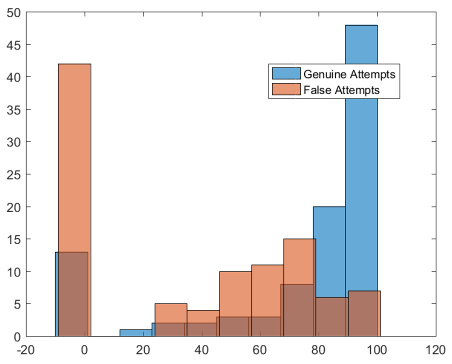
(d) $\theta = 20$



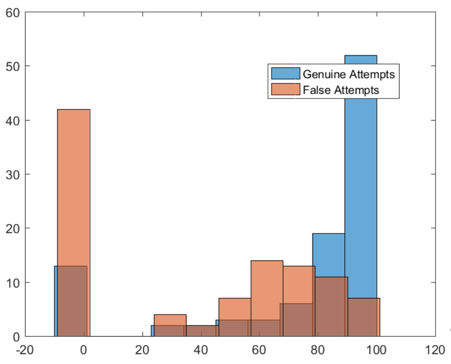
(e) $\theta = 25$



(f) $\theta = 30$



(g) $\theta = 35$



(h) $\theta = 40$

Figure 5.2: Histogram of $ThetaPercentages$ under various θ values

5.4 Fuzzy Vault Evaluation

5.4.1 Failure Threshold

The alignment and the fuzzy vault need a certain number of minutiae inside a template, in order to function correctly. Specifically, the 4 triangles that we used for alignment need 12 minutiae (3 distinct minutiae for each triangle). Fuzzy vault needs at least $d+1$ minutiae to unlock a vault. But since we use a larger set of points to try all combinations among them, fuzzy vault actually needs at least $subsetNumber$ number of minutiae. Hence, we introduce a new $failureThreshold$ variable which indicates the minimum number of minutiae that a template needs. If a template has less minutiae than $failureThreshold$, we consider it a Failure to Capture. In our experiments, we consider $failureThreshold = 25$. This threshold is based on the necessity for at least 12 minutiae for the alignment and 13 minutiae for the vault. Based on this value, 8.625% of our complete database's templates will be considered FTC, which is an acceptable score. Further FTA rates under false acceptance and false rejection experiments are shown in the results below as FTA FAR and FTA FRR respectively.

5.4.2 Fixed Parameters

Due to the large number of parameters in our system, testing all combinations of different values among them proves infeasible. Thus, we use some fixed values in our experiments, as shown in Table 5.4. Using a seventh-degree polynomial, along with the $GF(2^{16})$ finite field, our system can protect a $(7+1)\cdot 16 = 128$-bit secret, which is an ideal key size for widely used cryptographic ciphers, such as AES.

Table 5.4: Fixed Parameters

Degree	numberOfChaffs	FailureThreshold	trianglesNumber	subsetsNumber
7	$200 - minutiaeNo$	25	4	13

5.4.3 Fuzzy Vault Results

Table 5.5 summarizes the performance rates of the fuzzy vault using the fixed parameters mentioned before. We performed two experiments. In our first experiment, we used a multi-enrolment approach, whereas in the second, a multi-query approach. Regarding the alignment parameters, we chose $trianglesNumber = 4$ and $\delta r = 20$, $\delta\theta = 20$, as mentioned in the Alignment Evaluation section of this chapter. *Enrolled* stands for the number of enrolled templates of the same user, whereas *Query* stands for the number of query templates of the same user. $gen.No$ stands for the number of genuine minutiae that were stored inside the vault. All rates are percentages (%). FTQ and FTA stand for Failure to Quantize and Failure to Acquire rates during false rejection (FRR) and false acceptance (FAR).

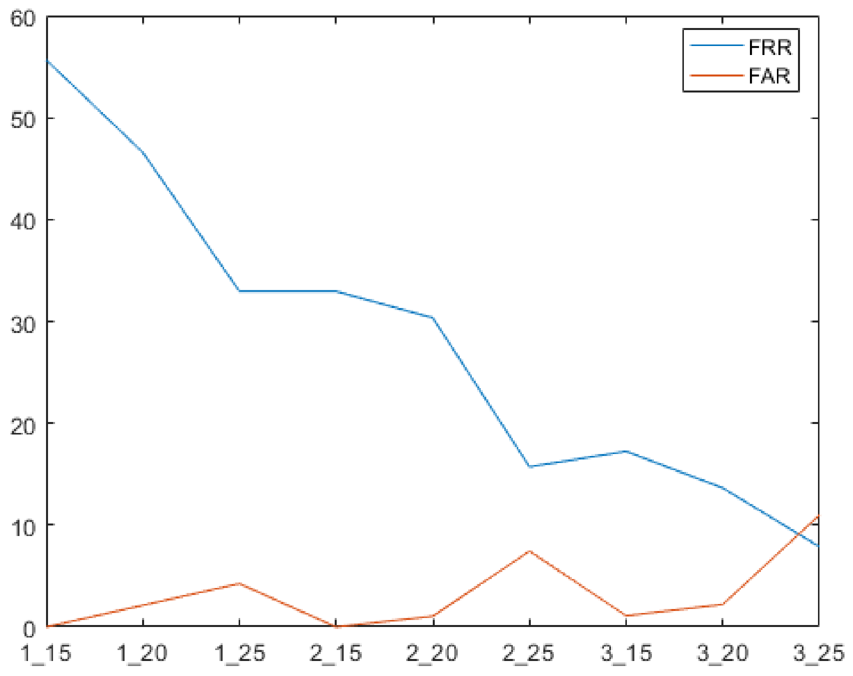


Figure 5.3: FRR and FAR rates of multi-enrolment experiment

Table 5.5: Fuzzy Vault Performance Rates of Multi-Enrolment Experiment

Enrolled	gen.No	FRR	FAR	FTQ FRR	FTQ FAR	FTA FRR	FTA FAR
1	15	55.68	0	2	4	10	2
1	20	46.59	2.13	2	4	10	2
1	25	32.95	4.26	2	4	10	2
2	15	32.95	0	2	4	10	2
2	20	30.34	1.06	1	4	10	2
2	25	15.73	7.44	1	4	10	2
3	15	17.24	1.11	3	8	10	2
3	20	13.64	2.22	2	8	10	2
3	25	7.87	11	1	7	10	2

A plot of FRR and FAR of multi-enrolment experiment is shown in Figure 5.3.

Table 5.6: Fuzzy Vault Performance Rates of Multi-Query Experiment

Query	gen.No	FRR	FAR	FTQ FRR	FTQ FAR	FTA FRR	FTA FAR
1	15	38.30	1.08	3	4	3	3
1	20	29.47	2.15	2	4	3	3
1	25	24.21	4.30	2	4	3	3
2	15	26.37	1.22	1	2	8	16
2	20	19.35	2.44	1	2	6	16
2	25	14.89	6.10	1	2	5	16
3	15	22.34	1.15	1	3	5	10
3	20	14.43	2.30	0	3	3	10
3	25	13.27	6.82	0	2	2	10

A plot of FRR and FAR of multi-query experiment is shown in Figure 5.4.

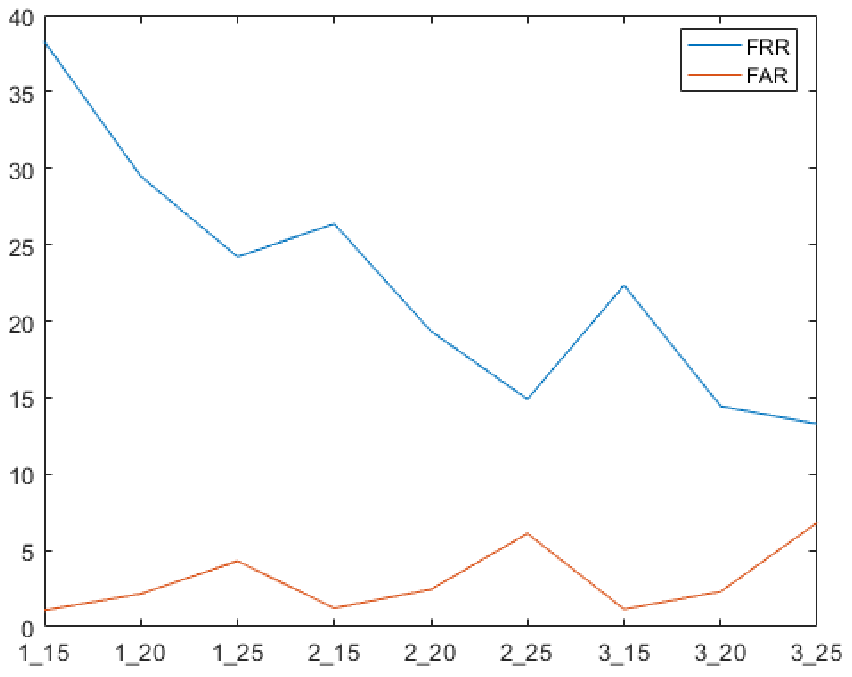


Figure 5.4: FRR and FAR rates of multi-query experiment

Figure 5.5 portrays the FRR and FAR rates of both experiments. A multi-query approach provides better FRR rates using 1 and 2 query templates, but worse rates using 3 templates, while the FAR rates remain at the same level. Using 25 genuine points greatly increases the FAR rates in all cases, especially using 3 templates. In many use cases these FAR rates would be unacceptable, and thus the use of 25 genuine points inside the vault should be avoided.

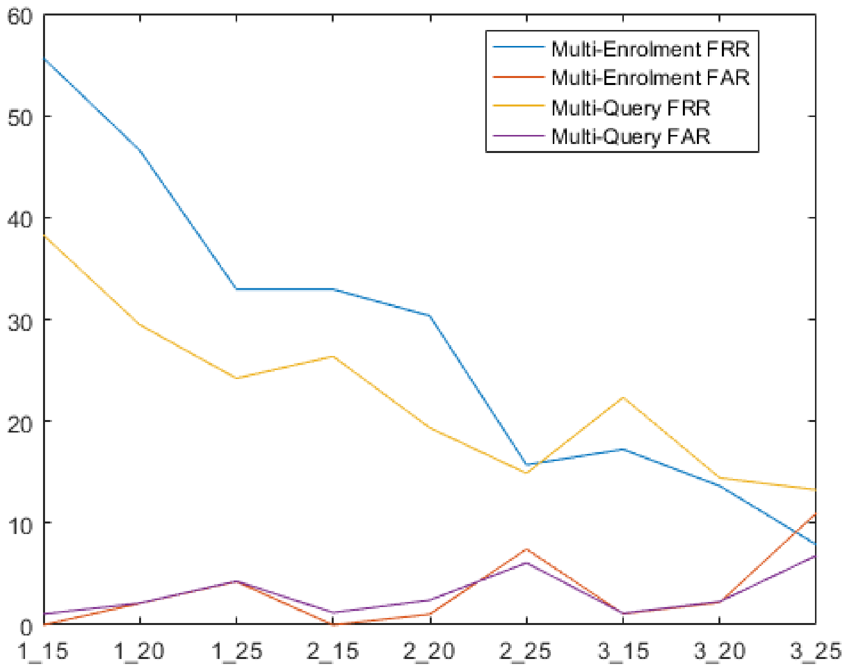


Figure 5.5: FRR and FAR rates of both experiments

5.4.4 Security Analysis

The worst-case scenario of a full brute-force attack against our fuzzy vault scheme, where an adversary would try all possible combinations, would require time $\binom{200}{8}$, which provides approximately 45 bits of security.

Using 15,20 and 25 genuine minutiae inside the vault, the brute force search space size is $\frac{\binom{200}{8}}{\binom{15}{8}}$, $\frac{\binom{200}{8}}{\binom{20}{8}}$, $\frac{\binom{200}{8}}{\binom{25}{8}}$, which is equivalent to approximately 33, 29 and 26 bits of

security respectively.

Chapter 6

Discussion and Conclusion

In this chapter, we conclude this work by discussing the advantages and possible limitations of our fuzzy vault system. We present its applicability and usability and finally, we recognize some perspectives for future research.

6.1 Advantages and Limitations

The greatest advantage of the fuzzy vault scheme is the order invariance feature. This is an ideal feature for fingerprints, since it is considered to be extremely difficult to put minutiae in the exact and correct order in a reliable way. Although fuzzy vault needs the two templates to be aligned in order to match, it does not require a specific alignment method. Thus, the alignment can be decoupled from the rest of the system, allowing us to implement our own alignment method and evaluate it separately.

As we mentioned above, there are two types of fingerprint alignment, minutiae-based and non-minutiae methods. Minutiae methods use actual minutiae to assist in the alignment, whereas non-minutiae based methods use non-minutiae patterns found on the fingerprint image. A biometric system does not usually store fingerprint images in its database. This avoids the necessity to perform feature extraction on the enrolled template during every matching. The templates are stored, after feature extraction, usually on a minutiae-based format as the database that we use for our experiments, and the fingerprint image is discarded. Unfortunately, already implemented systems that do not store the fingerprint image cannot use non-minutiae alignment methods, since the image is not available anymore and no further data can be extracted from it. These systems need to re-enroll their users in order to use the fuzzy vault. If there is a huge number of users already enrolled, re-enrollment could be impossible in practice. For instance, a government service might have millions of enrolled users, and re-enrollment for all of them would lead to a huge expense and public discomfort. For that reason, our minutiae-based method is ideal for already implemented systems with minutiae templates stored.

One clear limitation is the performance degradation, security and privacy risks derived from the use of minutiae for alignment. To avoid the unlocking of the vault with helper data minutiae, we use separate minutiae for the alignment and the vault. This mitigates the security risk, since the vault cannot be unlocked, but a partial privacy risk remains due to the fact that helper data minutiae remain exposed and their protection cannot be based on the fuzzy vault scheme. In this case, helper data must be protected using traditional cryptography.

Moreover, after our security analysis, the fuzzy vault system provides little entropy in comparison to modern non-biometric encryption schemes. Little entropy is a common issue found on fingerprints (Young et al.[39]). Hence, the fuzzy vault should be protected further using traditional cryptography. A simplest way to do that is by using a two-factor authentication method, where fingerprints are the second factor. The first factor could be something the user knows or has (for example a password or a card). This first factor serves as a cryptographic key for the encryption of the helper data and the fuzzy vault using a block cipher.

6.2 Usability and Applicability

As a biometric cryptosystem, fuzzy vault can protect both the user's minutiae and a secret value at the same time. This is considered to be an extremely useful feature, since it introduces new applications for biometric-based user recognition and authentication. In this section, we provide two examples on how a fuzzy vault can be used; first as a user authentication mechanism, and second, as a key protection mechanism for data encryption.

A simple user authentication mechanism is as follows: A secret value S is encoded as the polynomial's coefficients, using the user's minutiae m inside the vault. A hash digest of the secret, $H(S)$, is computed using a cryptographic hash function and stored. When a user enters his fingerprint for authentication, the system unlocks the vault using the minutiae m' and it retrieves a secret S'. If $H(S) == H(S')$, the user is authenticated.

For the second example, the fuzzy vault is used to protect a cryptographic key, used by a cryptographic cipher for data encryption. In a cloud-based service, a user stores his/her files and wants to encrypt those files using fingerprints. A random AES-128 key is created and the files are encrypted with this key. Since our fuzzy vault can protect a 128-bit secret, the AES key can be protected as the vault's secret value.

Since nowadays fingerprints are the most popular biometric modality, an attractive use-case scenario is for mobile applications that include an integrated biometric sensor. Mobile devices can also assist in the protection of the fingerprint templates by performing all fingerprint data manipulation within their Trusted Execution Environment. TEE is an isolated protected environment inside the main processor of a device, consisting of its own processor, memory and storage. All sensitive operations are executed inside the TEE and sensitive data never leave it. Most recent mobile phones are equipped with a hardware-based TEE, making it a perfect application for biometrics [7].

6.3 Future Work

In this thesis, we implement a fuzzy vault scheme using a minutiae-based alignment method. We then proceed to evaluate it under different parameter values, highlighting the security capabilities of the implementation.

In the context of the specific research, due to the high quantity of computations, testing all combinations of parameters is considered to be infeasible for our experimental setup. An implementation in a low-level language, executed on a high-performing machine would provide much faster results, allowing further experimentation on the influence of the parameters on the FAR and FRR trade-offs.

Since the complexity of a brute-force attack is based on both the degree of the polynomial and the number of points in the vault, the most important parameters to experiment with would be the degree of the polynomial and the number of genuine and chaff points inside the vault. Additionally, the increase and decrease of the system's entropy can be monitored and evaluated under these parameters.

In general terms, future research should be focused on non-minutiae based alignment methods, since these methods perform better and can lead to new more privacy-friendly implemented systems. An ideal fingerprint-based fuzzy vault system could provide both methods to maximize its efficiency and utilize both systems' advantages.

Finally, further linkability analysis should be performed based on the alignment helper data. The correlation of helper data originating from different databases poses a serious privacy threat, due to the fact that they can reveal the identity of the user and link the user to other applications. Helper data cannot be protected using our implemented system and their protection relies on the use of a second authentication factor based on other cryptographic schemes.

Bibliography

[1] URL: http://www.ekds.gov.tr/bio/databases.html, last checked on 2017-03-10.

[2] URL: http://bias.csr.unibo.it/fvc2002/, last checked on 2017-03-10.

[3] R. M. Bolle, J. Connell, S. Pankanti, N. K. Ratha, and A. W. Senior. *Guide to biometrics*. Springer Science & Business Media, 2013.

[4] J. Breebaart, B. Yang, I. Buhan-Dulman, and C. Busch. Biometric template protection. *Datenschutz und Datensicherheit - DuD*, 33(5):299–304, 2009.

[5] P. Campisi. *Security and Privacy in Biometrics*, volume 24, chapter 8.5 Matching Performance and Security. Springer, 2013.

[6] P. Campisi. *Security and Privacy in Biometrics: Towards a Holistic Approach*, pages 1–23. Springer London, London, 2013.

[7] J.-E. Ekberg, K. Kostiainen, and N. Asokan. The untapped potential of trusted execution environments on mobile devices. *IEEE Security & Privacy*, 12(4):29–37, 2014.

[8] Y. Feng and P. C. Yuen. Biometric template protection: Towards a secure biometric system. *Handbook of Pattern Recognition and Computer Vision*, page 455, 2010.

[9] Information technology – Biometric data interchange formats – Part 2: Finger minutiae data. Standard, International Organization for Standardization, Geneva, CH, 2005.

[10] A. Jain, R. Bolle, and S. Pankanti. *Biometrics: personal identification in networked society*, volume 479. Springer Science & Business Media, 2006.

[11] A. K. Jain, K. Nandakumar, and A. Nagar. Biometric template security. *EURASIP Journal on advances in signal processing*, 2008:113, 2008.

[12] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. On the similarity of identical twin fingerprints. *Pattern Recognition*, 35(11):2653–2663, 2002.

[13] A. K. Jain and A. Ross. Introduction to biometrics. In A. Jain, P. Flynn, and A. Ross, editors, *Handbook of Biometrics*, pages 1–22. Springer, New York, 2008.

[14] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, 14(1):4–20, 2004.

[15] J. Jeffers and A. Arakala. Minutiae-based structures for a fuzzy vault. In *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the*, pages 1–6. IEEE, 2006.

[16] J. Jeffers and A. Arakala. Fingerprint alignment for a minutiae-based fuzzy vault. In *Biometrics Symposium, 2007*, pages 1–6. IEEE, 2007.

[17] X. Jiang and W.-Y. Yau. Fingerprint minutiae matching based on the local and global structures. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*. IEEE, September 2000.

[18] A. Juels and M. Sudan. A fuzzy vault scheme. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, page 408. IEEE, 2002.

[19] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36. ACM, 1999.

[20] M. Kayaoglu, B. Topcu, and U. Uludag. Standard fingerprint databases: Manual minutiae labeling and matcher performance analyses. *CoRR*, abs/1305.1443, 2013.

[21] A. Kholmatov and B. Yanikoglu. Biometric cryptosystem using online signatures. In *International Symposium on Computer and Information Sciences*, pages 981–990. Springer, 2006.

[22] D. Maltoni and R. Cappelli. Fingerprint recognition. In A. Jain, P. Flynn, and A. Ross, editors, *Handbook of Biometrics*, pages 31–35. Springer, New York, 2008.

[23] D. Maltoni, D. Mario, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*, chapter 1.4, pages 8–11. Springer, New York, second edition, 2009.

[24] A. Nagar, K. Nandakumar, and A. K. Jain. Securing fingerprint template: Fuzzy vault with minutiae descriptors. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[25] K. Nandakumar. *Multibiometric Systems: Fusion Strategies and Template Security*. PhD thesis, Michigan State University, Department of Computer Science and Engineering, 2008.

[26] K. Nandakumar and A. K. Jain. Multibiometric template security using fuzzy vault. In *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, pages 1–6. IEEE, 2008.

[27] K. Nandakumar, A. K. Jain, and S. Pankanti. Fingerprint-based fuzzy vault: Implementation and performance. *IEEE transactions on information forensics and security*, 2(4):744–757, 2007.

[28] M. T. Nguyen, Q. H. Truong, and T. K. Dang. Enhance fuzzy vault security using nonrandom chaff point generator. *Information Processing Letters*, 116(1):53–64, 2016.

[29] C. Rathgeb and A. Uhl. A survey on biometric cryptosystems and cancelable biometrics. *EURASIP Journal on Information Security*, 2011(1):3, 2011.

[30] A. Ross and A. K. Jain. Human recognition using biometrics: An overview. *Annals of Telecommunications*, 62(1):11–35, 2007.

[31] S. B. Sadkhan and K. Ruma. Evaluation of polynomial reconstruction problem using lagrange interpolation method. In *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, volume 1, pages 1399–1403. IEEE, 2006.

[32] W. J. Scheirer and T. E. Boult. Cracking fuzzy vaults and biometric encryption. In *Biometrics Symposium, 2007*, pages 1–6. IEEE, 2007.

[33] P. Tang. Prototype implementation of fuzzy vault cryptosystem in matlab. URL: https://github.com/paulswtang/vault-tec, last checked on 2017-03-10.

[34] A. B. J. Teoh and J. Kim. Error correction codes for biometric cryptosystem. 32(6):39–49, 2015.

[35] C.-A. Toli and B. Preneel. A survey on multimodal biometrics and the protection of their templates. In *IFIP International Summer School on Privacy and Identity Management*, pages 169–184. Springer, 2014.

[36] U. Uludag, S. Pankanti, and A. K. Jain. Fuzzy vault for fingerprints. In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pages 310–319. Springer, 2005.

[37] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain. Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, 92(6):948–960, 2004.

[38] S. Yang and I. Verbauwhede. Automatic secure fingerprint verification system based on fuzzy vault scheme. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 5, pages v–609. IEEE, 2005.

[39] M. R. Young, S. J. Elliott, C. J. Tilton, and J. E. Goldman. Entropy of fingerprints. *International Journal of Science, Engineering and Computer Technology*, 3(2):43, 2013.