Security, Privacy and Performance Improvements for Fuzzy Extractors

by

Renaud Brien

Thesis submitted to the University of Ottawa
In partial fulfillment of the requirements
For the Ph.D. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

With the usage of biometrics becoming commonly used in a variety of applications, keeping those biometrics private and secure is an important issue. Indeed, the convenience of using biometrics for authentication is counteracted by the fact that they cannot easily be modified or changed. This can have dire consequences to a person if their biometrics are leaked.

In the past decades, various techniques have been proposed to solve this problem. Such techniques range from using and storing randomized templates, using homomorphic encryption, or using biometric encryption techniques such as fuzzy extractors. Fuzzy extractors are a construction that allows the extraction of cryptographic keys from noisy data like biometrics. The key can then be rebuilt from some helper data and another biometric, provided that it is similar enough to the biometrics used to generate the key. This can be achieved through various approaches like the use of a quantizer or an error correcting code.

In this thesis, we consider specifically fuzzy extractors for facial images. The first part of this thesis focuses on improving the security, privacy and performance of the extractor for faces first proposed by Sutcu et al. Our improvements make their construction more resistant to partial and total leaks of secure information, as well as improve the performance in a biometric authentication setting.

The second part looks at using low density lattice codes (LDLC) as a quantizer in the fuzzy extractor, instead of using component based quantization. Although LDLC have been proposed as a quantizer for a general fuzzy extractor, they have yet to be used or tested for continuous biometrics like face images. We present a construction for a fuzzy extractor scheme using LDLC and we analyze its performance on a publicly available data set of images. Using an LDLC quantizer on this data set has lower accuracy than the improved scheme from the first part of this thesis. On the other hand, the LDLC scheme performs better when the inputs have additive white Gaussian noise (AWGN), as we show through simulated data. As such, we expect it to perform well in general on data and biometrics with variance akin to a AWGN channel.

Acknowledgements

I would like to thank my supervisor, Carlisle Adams, for all the support and insights he provided me during this journey, as well as my colleges and friends with whom I worked these past few years. I would also like to thank examiners Andy Adler, Guy-Vincent Jourdan, Burak Kantarci and Dimitrios Hatzinakos, who read this work in its entirety and provided useful feedback. Finally, I would like to thank my father for providing a fresh look to this work.

Dedication

To my love and my family, who supported and encouraged me all throughout this long journey.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

ACC	Accuracy
AWGN	Additive White Gaussian Noise
CVP	Closest Vector Problem
EER	Equal Error Rate
eTA	electronic Travel Authorization
FN	False Negative
FNR	False Negatie Rate
FP	False Positive
FPR	False Positive Rate
GMR	Gaussian Mixture Reduction
GQL	Gaussian Quadratic Loss
LDLC	Low Density Lattice Codes
LDPC	Low Density Parity Check
MM	Moment Matching
MAC	Message Authentication Code
PCA	Principal Component Analysis
PPV	Positive Predictive Value
RP	Random Projections
SVP	Shortest Vector Problem
TN	True Negative
TP	True Positive
TPR	True Positive Rate

Chapter 1

Introduction

Powerful electronic devices are now becoming more ubiquitous in society and they are making various day-to-day activities easier and more convenient for users in different spheres of life. Convenience is, among other things, being improved by replacing passwords or PINs with the use of biometric identification, from using fingerprints and pictures to unlock personal phones, to using facial recognition at border crossings or airports.

This increase in the use of biometrics for everyday purposes has helped push the question of privacy to the forefront of many computer applications. In the case of biometrics, privacy and security is a huge concern, since the leakage or compromise of a biometric renders it unusable or a liability. Indeed, it is easy to change a password, but it is impractical or impossible to change a person's biometrics like a fingerprint, face or even voice.

Typically, an efficient biometric recognition system works by storing the biometrics of the various users and using them at a later date for recognition or authentication. This comes with various possible drawbacks for the entities using such systems. This ranges from ensuring the protection of the data from leaks and hackers, to making sure all the privacy policies (such as any governmental policies or international policies like the GDPR in Europe) are respected. These policies can sometimes be costly to implement and alternatives that can better protect biometrics, or even remove the need to store them altogether, might be preferable.

This thesis will study an alternative to classical biometrics authentication: a privacy preserving technique known as *fuzzy extractors*. Fuzzy extractors were first formally defined by Dodis et al [DORS08]. Such techniques can be seen as replacing the problem of securely storing biometrics with the problem of securely storing private keys. More specifically, we will look at the fuzzy extractor for faces introduced by Sutcu et al. in [SLM09], as it is one of the few fuzzy extractor designed (and tested) specifically for facial recognition instead of general continuous features. In this thesis, we will first implement and improve this particular fuzzy extractor for faces, then we build and test a new fuzzy extractor for faces using low density lattice codes (LDLC).

1.1 Motivations and Setting

Our research was first linked to a prototype project financed in part by the Government of Canada, where we implemented Sutcu et al.'s fuzzy extractor to test in live scenarios. Two fellow students also participated in the research: David Bissessar, who mostly coordinated the direction and the concept of the project as well as its integration with the other sections of the bigger project, and Alain Tambay, who did most of the work of finding facial images and processing them to be used and imported in our algorithms for testing as well as helping in coding various parts of the implementation. My part in this particular project involved researching, implementing and improving the scheme, as well as coding parts of the implementation and running tests.

This project, called Chain of Trust, looked at using and integrating biometrics, as well as various other technologies, in a privacy preserving way to expedite the border crossing at airports for low-risk travellers. Typically, a traveller coming into Canada through an airport has to file for an electronic Travel Authorization (eTA), sometimes months or years before the travel date as an eTA is valid until the passport it is linked to expires. Once the traveller arrives on Canadian soil, they go through the standard process of border crossing by submitting a customs declaration card and having their travel documents checked by border agents.

In the Chain of Trust project, a traveller would use a cellphone application to help with the process. They would use the application to fill in their eTA using their passport information, then generate a privacy-preserving biometric identifier. This identifier was created by having the user submit a series of biometric images in the form of well structured "selfies", comparing them to the passport image to assess the identity of the user, and then using these facial biometrics to generate a cryptographic key. This key was then used to generate an encrypted token for future authentication as well as to generate a message authentication code (MAC) to "sign" the eTA or any of their relevant documents. These signed travel documents would then be given back to the traveller and stored on their device, while the submitted biometrics are discarded.

When the traveller finally arrives in a Canadian airport, they would go to a designated booth or kiosk and provide the signed documents they have on their cellphone, a picture would be taken by the kiosk and the signature would be verified. If the authentication is successful, the traveller is recognized as the author and owner of the documents and can go through an expedited process of border crossing. If the authentication fails, then the traveller simply goes through the standard procedures by interacting with a human operative.

The goal of our biometric section of this project was the privacy-preserving facial authentication section to help reduce the need for human interaction in authenticating a user's identity. The main driving factor of our part of the project was privacy and re-usability of the biometrically derived keys as authentication and identification tools. This was done by using a fuzzy extractor to generate a cryptographic key using facial biometrics.

This particular border crossing setting guided our choices of facial dataset used for testing the fuzzy extractors. Indeed, this particular application required images that had the

same format as a passport picture (portrait, front facing, little to no accessories), images that were taken on different days, and a sufficient number of images to enrol and authenticate users using a fuzzy extractor. Due to various sets of budget and legal constraints, such as not having access to large private image databases nor having the resources to build a database ourselves, we chose the free and an openly accessible AR dataset [MB98] and took a subset of images that could fit our criteria for testing. The AR dataset was a good option as it satisfied our criteria and the images did not require extensive pre-processing to be used in the algorithms.

Finally, the scheme from Sutcu et al. was chosen as it is one of the few practical constructions of continuous source fuzzy extractors dealing with facial recognition. Other constructions (see [BDHV07b, PvdG16, BDHV07a, VTO$^+$10]) are more general and theoretical continuous fuzzy extractors.

1.2 Contributions and Organization

Our implementation revealed many areas where improvements had to be made to implement the fuzzy extractor given by Sutcu et al. into an application that could be deployed in a real world environment. Furthermore, when tested on the database we had, the scheme performed poorly as no user ever authenticated regardless of the chosen parameters. These security and implementation improvements also have the effects of increasing the entropy of the keys generated from facial biometrics, which are believed to contain a small amount of information (see [YA10]). We will cover in detail the original approach as well as all the improvements we propose in chapter 3.

We will also provide results by testing our version of the system with the AR data set [MB98], a publicly available facial image database. This particular database was chosen for a few main reasons. First, it is a free and publicly available data set, which fell into our resources constraints. Secondly, it contains enough images of high quality that we can split the data set into two subsets : one to train a feature extractor with a large number of components in the extracted features, and a second set with different people to test the scheme without bias. Finally, this set contained five or more front facing images of each user each with different expressions or lighting that resembled the standard passport image format. This will be the first section of this thesis.

The second subject we will study is a major component of many fuzzy extractors: the use of a quantizer, or an error correcting code, to deal with the variance and the continuous nature of inputs like facial biometrics. In [SLM09], Sutcu et al. used a quantizer that works on each component individually. This made us consider if the use of a different quantizer that would instead work directly over the real space could improve the accuracy of the system. This section was personal research unrelated to the Chain of Trust project.

Lattices appear to be a perfect fit for quantization. The recent developments in generating and decoding low density lattice codes (LDLC) suggest this approach could be viable. Low density lattice codes are somewhat novel types of codes, introduced in [SFS08], that can achieve channel capacity and possess an efficient decoding algorithm. Furthermore,

using a lattice quantizer in a fuzzy extractor was suggested in [PvdG16], with a proposed application of a key reconciliation protocol for two devices in close proximity. Lattices have also been studied extensively, since they arise in various areas of mathematics and they have seen use in many applications from physics to cryptography.

Looking at the viability of using LDLC as a quantizer for a fuzzy extractor is the second subject we explore in this thesis. We implement lattice codes and use them to replace the quantizer in our improved version of Sutcu et al.'s fuzzy extractor. This new scheme is then tested using the same face database as the improved scheme. We also look at how to best choose the lattice code parameters to fit the setting of a fuzzy extractor for faces. We finish by testing and comparing the two schemes, when the input have additive white noise distribution by using simulated data.

The Python code written and used for this thesis is available on GitHub at:

`https://github.com/RBrie/FuzzyExtractorAndLattices`

Chapter 2

Biometrics, Lattices and Fuzzy Extractors : Background and definitions

In this chapter, we provide some background information on biometric authentication, a brief introduction to lattices and we define fuzzy extractors.

2.1 Biometrics

Biometrics consists of the (automated) method to identify an individual based on distinctive biological or behavioural characteristics [MTP11, Dun09, Set06]. A biometric can also refer to a measurement of such characteristics. These identifying characteristics are varied and many have been used in various biometrics system. Some of these characteristics include, among others, fingerprints, irises, faces, voice, written signatures, and keystroke patterns.

The process that biometric recognition and authentication follow is divided into two phases: the *enrolment* phase and the *authentication* or *verification* phase. The enrolment is the process where a sample, or many samples, of a user's biometric is collected, processed into a biometric *template* and then stored. The authentication phase is where a new sample is collected, processed into a template and then compared to the stored template(s). In an authentication setting, the template is used to authenticate the identity of the user, while in a recognition or classification setting the template is used to try and match it to an enrolled user.

The collection of a biometric consists of capturing the biometric through a *sensor* (such as a camera, a fingerprint reader, a microphone, etc.) before being processed and normalized into a digital representation b. For example, a fingerprint might be represented as a grey-scale image b, where dark areas represents the ridges and where white areas represent the valleys of the fingerprint [MMJP03].

In the particular case of captured images, like the facial biometrics studied in this work, the data is often preprocessed before a template is generated from it. This preprocessing typically includes centering or cropping the image, normalizing the brightness, combining the RGB values of each pixel into a single grey-scale pixel value and applying of various other normalizations of the images such as the application of a mask (see [Bow02]). This preprocessing is very important in a real world application, since the images will not always be captured in well controlled environment. The representation b for an image is then the vector containing all the pixel values of the processed image.

The size of the captured biometric can sometimes be too large to be used efficiently and working directly with large unprocessed inputs can sometime cause over-fitting bias in classification algorithms. This is especially true when dealing with images. It is also likely that many subsets of components of a captured biometric can be correlated and keeping these correlated values can further negatively affect classification or matching. This is sometimes referred to as the 'curse of dimensionality' and it is a problem that frequently arises in fields like data mining and artificial intelligence.

As such, to deal with both the size, the efficiency and improve the performance, a biometric *template* is constructed from the captured biometric. This template is smaller representation of the data that contains a specified subset of discriminating and identifying information, called *features*, from the biometric. This reduction can allow for easier storage and better performance when using of the biometrics. Various dimensionality reduction techniques [Lar06, Bow02], like Principal Components Analysis (PCA), can be used to extract the most relevant information of a type of biometrics.

This step of dimensionality reduction is typically called *feature extraction* and is done through a *feature extractor*, which we will denote by f_e.

Definition 2.1.1. Given a biometric b and a feature extractor f_e, the *extracted features*, or *template*, of b is $x = f_e(b)$.

A good feature extractor should have the property that if the two biometrics b and b' are similar (for example if they are from the same user), then the extracted features x and x' should also be similar. This allows us to work with the features instead of the biometrics themselves. A suitable feature extractor is something that has to be chosen or constructed depending on the application and the biometrics used.

Regarding facial recognition, many techniques and methods have been proposed and used as feature extractors. Perhaps the earliest (automated) feature extractor was given by Turk and Pentland [TP91] and called *Eigenfaces*, which is a method using Principal Component Analysis (PCA) to extract features. Various other methods have been proposed using variations of PCA (for example [YyY02, RJY15], Fisher Discriminant Analysis and Linear Discriminant Analysis [BHK97, ZKC+98, SWY+06, QYX13, CBOB14], random projections [GBN05] as well as many more. An extensive review of many approaches can be found in [WHD18].

Many modern techniques use a Deep Learning approach to feature extraction and facial recognition by using machine learning techniques like Neural Networks and Convolutional

Neural Networks, like in [KSH12]. These classifiers and extractors can achieve very good performances and are generally trained on large and extensive databases. Some examples of these deep learning systems are DeepFace [TYRW14], FaceNet [SKP15] and DeepID3 [SLWT15].

We point out that facial recognition is still complex and it is still hard for systems to deal with many factors like illumination, poses, expression, time elapsed between the images, and image resolution, among others [LHK05, LBD+09, AMU97, BBD+10]. The early approaches like Eigenfaces were notoriously affected by such variations. Although modern feature extractors and facial recognition softwares are better at dealing with these factors, the specifics on how to achieve the best performance is often proprietary or they are trained on private databases.

As such, facial recognition methods are constantly being improved on and their performance is generally better than what many public methods can achieve. The caveat is that those improved techniques are products offered by companies working in biometric recognition and these techniques (or the way to train the systems) can be considered as trade secrets. Consequently, the state of the art in facial recognition is far beyond what can be found in the publicly available literature.

In this thesis, due to various resources constraints, we will work with PCA and Random Projections (RP) since they are two techniques found and well studied in the scientific literature. These feature extractors, especially PCA, can easily be replaced by any other methods, commercial or public, in the rest of this work. A better feature extractor for facial recognition will only increase the performance of the fuzzy extractors.

We will end this section by describing the two main types of biometric systems and different metrics and statistics used to measure their performances.

Biometrics are typically used in two types of systems: recognition systems and verification systems. Both first require users U_i to be *enrolled* in the system by obtaining a biometric template x_i and storing it in the system. The difference between the two systems comes when we try to match a new template x' to an enrolled user.

In a biometric recognition system, the goal is to try to match the new template x' from user U' to a user U that was enrolled previously in the system. This is a one-to-many matching and is done by finding the template x that minimizes $d(x, x')$, for some similarity measure d. The system then has a few possible outcomes: it *matches* U' to U or, if $d(x, x')$ is too big, it can return a *no-match* or determines that x' is either the biometric of a new user or not a biometric at all.

In the case of the biometric verification or *biometric authentication* system, the biometrics are used as passwords or identifiers. This is one-to-one matching. In such a system, a user U' claims to be the enrolled user U by providing the template x'. Then for some predetermined threshold ϵ, the system either says that $U' = U$ if $d(x, x') \leq \epsilon$ and authenticates the user, or that $U' \neq U$ if $d(x, x') > \epsilon$ and rejects the authentication.

Definition 2.1.2. Let U' be a user and U be the claimed identity. Set $H_0 : U' = U$ as the hypothesis that the claim is correct and $H_1 : U' \neq U$ as the hypothesis that the claim is incorrect.

If the authentication system correctly determines H_0 or H_1, the outcome is a *true positive* (TP) or *true negative* (TN) respectively.

If the system decides H_0 when $U' \neq U$, the outcome is a *false positive* (FP), or a *false accept*.

If the system decides H_1 when $U' = U$ is true, the outcome is a *false negative* (FN), or a *false reject*.

The *False Positive Rate* of the system, or FPR, is the probability of a false positive outcome : $P[H_0|U' \neq U]$.

The *False Negative Rate* of the system, or FNR, is the probability of a false negative outcome : $P[H_1|U' = U]$.

The *Positive Predictive Value* of the system, or PPV, is the probability that a positive outcome is a true positive : $P[U' = U|H_0]$.

		True Condition	
		$U' = U$	$U' \neq U$
Test Decision	H_0	TP	FP
	H_1	FN	TN

Table 2.1: Table of Outcomes

The performance of an authentication system can then be determined by its FPR and FNR, which are greatly dependent on the chosen threshold ϵ. When given the outcomes of a test, we get that FPR $= \frac{\text{FP}}{\text{TN+FP}}$ and FNR $= \frac{\text{FN}}{\text{TP+FN}}$. The PPV can similarly be obtained as PPV $= \frac{\text{TP}}{\text{TP+FP}}$.

The *Equal Error Rate*, or EER, is also sometimes used as a performance measure. The EER is the error rate when ϵ is chosen such that the FPR and FNR are equal.

2.1.1 Principal Components Analysis

In this section, we will cover the Principal Components Analysis method of dimension reduction. PCA is a method that was used in statistics and various other fields before being suggested as a facial recognition tool in [TP91]. It was suggested for facial recognition in the early 90s and is sometimes called *eigenfaces*. PCA is a well known and studied feature extractor that is often used in the literature, but its performance in facial recognition has been surpassed by many other techniques.

PCA works by projecting the inputs onto a chosen subset of the eigenspace of the covariance matrix of the training data. This eigenspace has the particular property that it encodes the variance of the data in the eigenvectors. That is, the data varies the most along the direction of the eigenvector associated with the biggest eigenvalue. PCA works as follows. To lighten the notation, $x \in \mathbb{R}^n$ will represent both row and column vectors, with the row/column representation being implicit to the formula.

Let $\{x_1, \ldots, x_m\} \in \mathbb{R}^n$, with $m < n$, be a set of training observations and let $\overline{x} = \frac{1}{m} \sum x_i$ be the observations' mean. Set X to be the $m \times n$ matrix where the i^{th} row of X is $x_i - \overline{x}$. Each row of X thus represents how each observation differs from the sample mean $\overline{x}$.

The covariance matrix is then $\text{Cov}(X) = X^T X$ and, because we assume $m < n$, it has $m - 1$ non-zero eigenvalues. Let $\lambda_1, \ldots, \lambda_{m-1}$ be the non-zero eigenvalues of $X^T X$ and let $v_1, \ldots, v_{m-1}$ be their associated eigenvectors. Furthermore, suppose that the eigenvalues are sorted such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{m-1}$.

Dimensionality reduction is achieved by selecting the first $k \leq m - 1$ eigenvectors to cover most of the variance of the data. From this selection, we build the $k \times n$ PCA matrix M, where the i^{th}, $i = 1, \ldots, k$ row of M is the eigenvector v_i. The dimension of a new input x is reduced to k by computing the extracted features y as

$$y = M(x - \overline{x}).$$

The value k is often referred to as the *number of components*. Matching techniques can then be used on these extracted features. We consolidate this information about PCA into what we will refer as *training a PCA extractor* in TrainPCA.

Algorithm 2.1.3 (TrainPCA)**.** *The algorithm TrainPCA takes as input a training set $T = \{x_1, \ldots, x_m\} \subset \mathbb{R}^n$ and a number of components $k < \min(m, n)$, and outputs a PCA feature extractor $f : \mathbb{R}^n \to \mathbb{R}^k$ defined as*

$$f(x) = M(x - \overline{x}),$$

where M is the matrix containing the first k eigenvectors associated with the k highest eigenvalues of Cov(X) (sorted in descending order) and where $\overline{x} = \frac{1}{m} \sum_{i=1,m} x_i$.

Using PCA for facial recognition

Let $T = \{x_1, \ldots, x_m\} \subset \mathbb{R}^n$ be a set of training images of faces and let k be the chosen number of components. From T and k, we can train a PCA extractor $f \leftarrow \text{TrainPCA}(T, k)$. Given a set of users U, with each user U_i having an image u_i, we can enroll them by computing and saving a template $v_i = f(u_i)$ for each user.

Once the system has users enrolled, we can try to match a new input x as follows:

1. Compute the template $y = f(x)$.
2. Find the template v_i that minimizes the euclidean distance $d(v_i, y)$.
3. If $d(v_i, y) < \epsilon$, for a predetermined threshold ϵ, we say that x matches user U_i.
4. If $\epsilon \leq d(v_i, y) < \tau$, for another predetermined threshold τ, we can say that x is a facial image of a user that is not enrolled in the system. Depending on the situation, y can be used to enroll a new user to the system.
5. Finally, if $\tau \leq d(v_i, y)$, we conclude that x is not a facial image.

We note that the thresholds ϵ and τ are generally determined empirically and their choice directly affects the accuracy and specificity of the system.

As a final remark, we also note that the extractor will be better at recognizing images from users that were in the training set T. Indeed, the eigenvectors in the extractor matrix are derived from the data from the training set. As such, a good practice would be to test a PCA extractor on users that are not in the PCA training set.

2.1.2 Random Projection

In this section, we will cover Random Projections (RP) as a dimensionality reduction method and how they have been suggested for facial recognition. The use of random projections for facial recognition was suggested and compared to PCA in [BM01] and [GBN05] as an alternative to PCA, since a RP feature extractor does not need to be trained like PCA. Then, RP was looked into for cancelable biometrics ([TY07]) and for privacy-preserving biometric verification ([WP10]). In [WP10], the authors also analyzed the use of random projections after PCA to add changeability and privacy to biometrics. In the case of biometrics, changeability is the desirable property that a single biometric can generate a multitude of different templates, allowing a user to safely reuse their biometric for different applications or to replace a particular template if compromised.

The notion of random projection as a dimentionality reduction method comes from the Johnson-Lindenstrauss lemma ([JL84]) which states that, given a set of points in $\mathbb{R}^n$, their pairwise distance is preserved up to a factor ϵ when projected on a random m-dimensional subspace where $n > m > O(\epsilon^{-2}\log(n))$.

This means that, as a feature extractor, RP only need us to generate a suitable random matrix. This is an advantage over PCA that requires training data and can become computationally expensive with big biometrics like images. Experiments done over text data ([BM01]) and images ([GBN05]) show that RP can provide good and similar accuracy to PCA, especially in higher dimensions.

Since our goal is to look at privacy-preserving facial recognition and fuzzy extractors, we will present the RP method given in [WP10]. We first look at the feature extractor generation.

Algorithm 2.1.4 (RPGen)**.** *The algorithm RPGen takes as input : the initial dimension n, the target dimension m, two keys k_1, k_2 and a system threshold t.*

1. *Generate a translation vector $\mathbf{d} \in \mathbb{R}^n$ using key k_1, with each element $d_i >> t$.*
2. *Use the key k_2 to generate an $m \times n$ random matrix R, where each entry is independent and identically distributed (i.i.d) from a Gaussian distribution with mean 0 and variance $\frac{1}{n}$. That is, for all i, j, $r_{i,j} \sim N(0, \frac{1}{n})$.*

The feature extractor $f_{RP(k)} : \mathbb{R}^n \to \mathbb{R}^m$ is then defined as

$$f_{RP(k)}(x) = \sqrt{\frac{n}{m}} R(x + \mathbf{d}).$$

If privacy and changeability are required, each different user of the biometric authentication system has a different RP feature extractor generated. In the case of any form of compromise, a user can generate a new feature extractor using a different key k_2. The translation vector $\mathbf{d}$ can be a fixed system parameter, or can be generated for each individual user.

The authors of [WP10] then prove that generating R and the feature extractor as above satisfy the requirements for the Johnson-Lindenstrauss lemma. That is, the pairwise distance of vectors is preserved by the projection. Furthermore, the addition of the translation vector $\mathbf{d}$ is there to allow for the changeability of the extracted features, provided that the parameter t is large enough.

Privacy with RP is linked to the ratio of the initial dimension n and target dimension m of the projection. If the random matrix R is obtained along an extracted feature $y = f_{RP}(x)$, the authors first show that we need $m \leq n/2$ to prevent the exact reconstruction of any elements of x. Although even if x cannot be perfectly recovered, an approximation $\hat{x}$ can still be obtained from R and y.

The privacy of the user will be preserved if the distance between x and $\hat{x}$ is large enough: $||x - \hat{x}||^2 > \tau$. The authors then show that the probability of this happening increases when n and $||x||^2$ increase and when m decreases. Since in a facial recognition setting n and $||x||^2$ will likely be fixed, we obtain that the smaller m is, the more private the system will be.

Finally, an RP extractor can be used after another feature extractor, like PCA, to provide randomization and changeability to the extractor. We will denote by PCARP the situation where we use PCA then RP in sequence.

Algorithm 2.1.5 (PCARPGen)**.** *The algorithm PCARPGen takes as input : the initial dimension n, the PCA dimension n_2, the target dimension m, two keys k_1, k_2, a system threshold t and a set of training images $T = \{x_1, \ldots, x_l\} \subset \mathbb{R}^n$ with $l \geq n_2$.*

1. *Compute a PCA feature extractor $f_{PCA} \leftarrow TrainPCA(T, n_2)$.*
2. *Compute the RP extractor $f_{RP} \leftarrow RPGen(n_2, m, k_1, k_2, t)$.*

The feature extractor $f_{PCARP(k)} : \mathbb{R}^n \to \mathbb{R}^m$ is then defined as

$$f_{PCARP(k)}(x) = f_{RP(k)}(f_{PCA}(x)).$$

In [WP10], the authors also compare PCA, RP and PCARP in the biometric verification setting. They arrive at the conclusion that PCA provides the smallest EER for all dimensions. PCARP and RP can achieve similar EER when projecting on a space of dimension 100+, with PCARP having a slightly lower EER than RP. The differences are amplified for smaller dimension. This difference can be explained by the fact that PCA projects onto the vector space whose basis encodes the most variance in the training data, while RP projects onto a completely random space.

These differences highlight a frequent compromise in security, where increased security and privacy come at the price of lowered efficiency. Indeed, with a smaller target dimension, RP provides more privacy and security at the expense of accuracy.

Finally, we note that although RP can provide privacy, changeability and template protection to biometrics, it is still only a feature extractor. If it is used in a classical biometric recognition system, then it still means that the templates need to be stored.

2.2 Security and Privacy Issues of Biometrics

In this section we present some of the security and privacy concerns related to the use of biometrics, as well as some typical angle of attacks on biometric recognition and authentication systems. We then present a few technologies that can remedy these issues. Finally, we will define our attacker model for the two schemes we present later in this work.

Biometric authentication and recognition systems, like many complex systems, can have many vulnerabilities (see for example [Adl08, PM17, SCS11, CSC08] for more details) and angles of attacks. These attacks are varied and some examples include tricking the sensors (Spoofing or Masquerading), bypassing the sensor (Replay attack) with an input, modifying or accessing the stored templates, intercepting and modifying messages between system entities, and modifying parts of the system themselves.

While many of these vulnerabilities can be protected with typical network security methods, some vulnerabilities remain even when the system is secured from tempering. For example, no matter how well the system is protected, a well chosen fake biometric or a specifically crafted input can be used to be falsely authenticated. In a facial recognition setting, techniques such as liveness detection can be used to help prevent spoofing attacks where the attacker presents a still image and insure the picture taken is from a live person.

Another important issue for biometrics is that of privacy. If a biometric template of any user is obtained, this can have widespread consequences. It renders any other system that uses that same template vulnerable to attacks and, depending on the feature extractor, it could be used to approximate an image of the biometric. It would then mean that the biometric of this user is now known and should not be reused again. It has also been shown that images can be rebuilt from quantized match scores [Adl04] and similar scores leaked by some biometric encryption schemes [Adl05].

Various approaches were proposed to preserve privacy in biometric authentication systems and they can be separated into three types : cancellable biometrics, biometric encryption, and, more recently, homomorphic encryption schemes.

1. **Cancellable biometrics** is a technique that consists of applying a transformation to the template before storing it. Matching and authentication is done with these transformed or randomized templates. If the transformation is partially or totally invertible, information about the template can be recovered, provided that the transformation is known. As such, the transformations should be kept secret.

See [RCCB07, TY07, WP10, TN05] for various examples on fingerprints and facial images.

2. **Biometric Encryption** is the general idea of binding biometrics to cryptographic keys in such a way that only a matching biometric can regenerate the key. The fuzzy extractors formally defined by Dodis et al. in [DORS08] can be seen as a generalization of many biometric encryption scheme. Some examples include: the Mytech scheme [SRS+98], Fuzzy commitment schemes [JW99, JS06], schemes using fuzzy extractors [BDHV07c, DORS08, SLM09], and various other techniques [DFM98, MRW99, LT03, TNG04, DKM+07].

3. **Homomorphic encryption** is the most recent approach and appears to be the current area of interest in privacy-preserving biometrics. The objective is to encrypt the templates with a homomorphic encryption scheme and then do all the recognition and authentication operations on the ciphertexts and afterward only decrypting the results. Some examples are found in [PM17, Abi17, Dro15, HH16, ABS15].

In this thesis we are interested in fuzzy extractors for facial biometrics over continuous inputs. Fuzzy extractors and similar techniques for discrete sources, like fingerprints, have seen a lot of study. Many biometric encryption schemes cited above are for discrete biometrics.

The literature for continuous source feature extractors for biometrics is smaller. Sutcu et al. [SLM09] proposed a fuzzy extractor for faces and tested it on the ORL database [Cam94]. In [BDHV07b], the authors build on the work of Dodis et al. and present a general construction of a fuzzy extractor for continuous features and [VTO+10, BDHV07c] further presented general constructions. In [PvdG16] they give a construction using lattice codes as the underlying error correcting code for the extractor.

Since our research was linked to a project where we generated keys from facial biometrics, we chose to examine and improve Sutcu et al.'s scheme. It was, at the time of the research, one of the only continuous fuzzy extractors that was specifically designed and tested for facial biometrics, while the others were more general constructions.

2.2.1 Attacker Model

Finally, we define our attacker model for the fuzzy extractor schemes presented in this work. We will consider how the schemes can provide more privacy and security compared to a regular biometric authentication setting against a semi-passive attacker.

An attacker can have two different success conditions: authenticating as another user or recovering the user's biometrics. We will consider a situation where the attacker:

1. has full knowledge of the scheme,
2. has access to all the public data,
3. can get access through leaks, partially or totally, to the data that is stored or kept secret,

4. *cannot* influence or modify the scheme or any messages between entities, and
5. has *not* compromised any part of the system like the sensors.

With such an attacker model, we consider how much additional privacy and security the schemes can offer. In the case of fuzzy extractors, full knowledge of the scheme will mean that the attacker knows how each part works (i.e. what type of quantizer, which feature extractor, how the keys are built, etc.), but will not know the specific random or user-specific values. Knowledge of those user-specific values falls into having partial or total knowledge of the stored data. Access to the public data will mean knowing the helper data, or any information that might be carried with the user themselves such as knowing the claimed identity linked that the helper data is linked to.

Next, we assume that the attacker cannot break into or modify the scheme in any way. As such, we do not consider the cases where the attacker could, for example, intercept a user's biometric during the enrolment phase or modify the code implementation to obtain any additional advantage.

Attackers with such capabilities are out of the scope for this particular work, because they can be considered as general security problems in computer science. The research on protecting against these general attack vectors on computer systems is a field of its own and many solutions have already been suggested. For example, the use of cryptography and MACs can prevent or mitigate a man-in-the-middle attack, while code integrity checks could prevent the tampering of the scheme implementation.

For similar reasons, we will assume that the system itself has not been compromised by the attacker. Additionally, it is of note that if a part of the biometric system is compromised, like the sensors, then an attacker can simply obtain the biometrics from the data used by the algorithms.

These are reasonable assumption for this work, as we are looking at how privacy and security is improved by using a fuzzy extractor compared to classical biometric authentication.

2.3 Lattices

In this section, we will cover some definitions on lattices as well as briefly introduce lattice codes. Our interest in lattice codes lies in using them to construct quantizers in a fuzzy extractor. We refer to [CS98] as a good reference on lattices and various problems associated with them.

Lattices can be seen as a regular grid covering the space $\mathbb{R}^n$ and they arise in various areas and problems. Three such problems, which are covered extensively in [CS98], are the sphere packing problem, the kissing number and the covering problem. The sphere packing problem asks "What is the densest arrangement of (non-overlapping) n-dimensional spheres in $\mathbb{R}^n$". The kissing number in a sphere packing is about the number of spheres that can touch a given sphere in a packing. The covering problem asks "How can we arrange n-dimensional spheres to cover the whole space, while minimizing the amount of overlap".

Solutions to these problems have applications in areas like physics, crystallography, chemistry and coding theory. In particular, solutions to the sphere packing problem can be linked to good channel codes, while the covering problem can be linked to good error-correcting codes.

Lattices have also been linked to the security of various cryptographic primitives such as the NTRU cryptosystem [HPS98], Gentry's fully homomorphic encryption [Gen09] and the learning with errors problem [Reg06, LPR10] also used as another fully homomorphic encryption primitive.

Definition 2.3.1. A *k-dimensional lattice* $\mathcal{L}$ *in* $\mathbb{R}^n$ is an additive subgroup of $\mathbb{R}^n$ consisting of all integer linear combinations of basis vectors in $\mathbb{R}^n$. That is, the lattice generated by the linearly independent vectors $v_1, \ldots, v_k \in \mathbb{R}^n$ is defined as :

$$\mathcal{L} = \{a_1 v_1 + \cdots + a_k v_k | a_1, \ldots, a_k \in \mathbb{Z}\}$$

The matrix $\mathbf{G} = [v_1 \ \ldots \ v_k]$ is called the *generator matrix* for $\mathcal{L}$.

The *Gram matrix* of a lattice $\mathcal{L}$ is the matrix $\mathbf{A} = \mathbf{G}\mathbf{G}^T$ and the *determinant* of a lattice is

$$\det(\mathcal{L}) = \det(\mathbf{A}).$$

If $\mathbf{G}$ is square, then $\det(\mathcal{L}) = \det(\mathbf{G})^2$.

The *dual lattice* of $\mathcal{L}$ is the lattice

$$\mathcal{L}' = \{x \in \mathbb{R}^n | x \cdot y \in \mathbb{Z}, \forall y \in \mathcal{L}\}.$$

If the generator matrix $\mathbf{G}$ for $\mathcal{L}$ is square, then the generator matrix for $\mathcal{L}'$ is the matrix $\mathbf{G}' = (\mathbf{G}^{-1})^T$.

Any given lattice can have more than one representation, so two lattices $\mathcal{L}$ and $\mathcal{L}'$ are said to be *equivalent* if

$$\mathbf{A}' = cU\mathbf{A}U^T,$$

for some real $c > 0$ and where U is an integer matrix with $\det(U) = \pm 1$.

For a lattice $\mathcal{L}$ in $\mathbb{R}^n$, the set

$$\{\alpha_1 v_1 + \cdots + \alpha_n v_n | \alpha_1, \ldots, \alpha_n \in [0, 1)\}.$$

is called the *fundamental region*. The volume of this fundamental region is given by $\sqrt{\det(\mathcal{L})}$ and is sometimes called the volume of the lattice.

Finally, given any discrete set of points $\mathcal{P} = \{P_1, P_2, \ldots\}$ in $\mathbb{R}^n$, the *Voronoi cell of* P_i, $V(P_i)$, is the set of points that are at least as close to P_i as to any other point P_j :

$$V(P_i) = \{x \in \mathbb{R}^n | d(x, P_i) \leq d(x, P_j), \text{ for all } j\}.$$

If the set of points $\mathcal{P}$ is a lattice $\mathcal{L}$, then all the Voronoi cells are congruent and their volume is equal to $\sqrt{\det(\mathcal{L})}$, the volume of the fundamental region.

2.3.1 Lattice codes

Given a lattice in $\mathbb{R}^n$ with representation $\mathbf{G}$ and a point $y \in \mathbb{R}^n$, we can ask the natural question of "what is the closest point in $\mathcal{L}$ to y?". This problem is also known as the Closest Vector Problem (CVP) and solving this question has various applications. Some direct applications of a solution to the CVP include using a lattice with representation $\mathbf{G}$ as a quantizer for the space, or using it for a lattice code in the euclidean space.

A solution to the CVP can also be used to solve another important problem in lattices : the Shortest Vector Problem (SVP). The SVP is the problem asking "Given a lattice L with basis $\mathbf{G}$, find the shortest non-zero vector in L." Both the SVP and the CVP are well known to be hard problems on arbitrary lattices, and they are linked to the security of various lattice-based cryptosystems (for example [LPR10]).

The hardness of these two problems means that, unless a lattice $\mathcal{L}$ has a representation $\mathbf{G}$ where the CVP is easy to solve, or other good properties, it is in general very hard to quantize an arbitrary input to it's closest lattice point.

Intuitively, the hardness of these problem can be directly linked to the representation $\mathbf{G}$ of the lattice. As such, to efficiently use lattices as quantizers or error-correcting codes, we must choose lattices that have both suitable properties and a suitable basis $\mathbf{G}$.

Low Density Lattice Codes (LDLC) were the first practical construction of a lattice code directly built in the Euclidean space. They are thus using a type of lattices where the CVP can be efficiently solved (with low error probability) using an iterative decoder. The act of decoding an arbitrary vector to its closest lattice point is known as *lattice decoding*.

Although lattice codes and their relation to various error correcting codes are described in [CS98], we will base this short section on [SFS08]. These authors provided a first construction, as well as an iterative decoding method, for lattice codes with inputs and outputs over the real numbers. These low-density lattice codes are based on the low-density parity check (LDPC) codes introduced by Gallager in [Gal63]. These LDLC codes can approach the channel capacity of an additive white Gaussian noise (AWGN) channel.

In particular, the authors look at an unconstrained power AWGN channel with noise variance σ^2. The capacity of such a channel was defined in [Pol94] as a measure of the density of codewords in the space. When translated to an n-dimensional lattice code $\mathcal{L}$ generated by $\mathbf{G}$, this means that error correction with small probability of error is possible only when

$$\sigma^2 < \frac{\sqrt[n]{\det(\mathcal{L})}}{2\pi e} = \frac{\sqrt[n]{\det(\mathbf{G})^2}}{2\pi e}.$$

Recall that a linear (n, k)-code can be defined by a $n \times k$ generator matrix $\mathbf{G}$ over a finite field $\mathbb{F}_q$ and the codewords are the vectors $c = \mathbf{G}x$, for $x \in \mathbb{F}_q^k$. The parity check matrix of a linear code is the $n \times (n-k)$ matrix $\mathbf{H}$ such that $\mathbf{H}y = \mathbf{0}$ if and only if $y \in \mathbb{F}_q^n$ is a codeword.

Decoding a message $y = c + e$ received over a noisy channel involves computing the syndrome $s = \mathbf{H}y = \mathbf{H}(c + e) = \mathbf{H}e$ and using it along with y to recover x. The specific

decoding algorithm for a code depends on the channel and the code used. We refer the reader to the extensive literature on error correcting codes (like [Moo05]) for more details on linear codes.

The approach in [SFS08] is an extension of the process for block codes to lattices over the reals. An n-dimensional lattice code is defined by a square $n \times n$ generator matrix $\mathbf{G}$ and codewords are vectors $c = \mathbf{G}x$ for $x \in \mathbb{Z}^n$. The parity check matrix is $\mathbf{H} = \mathbf{G}^{-1}$. The syndrome of $y = c + e$ is defined as $s = \text{frac}(\mathbf{H}c)$, where $\text{frac}(x) = x - \lfloor x \rceil$ is the fractional part of x. Like in block codes, s will be zero only when y is a codeword; otherwise it is used in the decoding algorithm.

The decoding algorithm for LDLC code is a belief propagation algorithm and has seen many improvements (see [KD10, YF09, HA16]) which we will discuss in Chapter 4. We will similarly leave the construction of the LDLC codes presented in [SFS08] to Chapter 4.

2.4 Fuzzy Extractors

One of the main drawbacks to classical biometric authentication is that a copy of a biometric, or its extracted features, needs to be stored in order to do identification or authentication. This creates issues for privacy and security, especially if the system is compromised. Indeed, a compromised biometric should not be used again and, unlike passwords, a biometric is for all intents and purposes impossible to change.

A generalized solution to this issue was proposed in [DORS08], where they formally define a primitive called a *fuzzy extractor* over discrete spaces. Following that, [BDHV07b] presented a fuzzy extractor construction over a continuous domain and [PvdG16] looked at using lattice codes as the underlying error correcting code for the extractor.

Informally, a fuzzy extractor is a construction that allows, from an input w, the extraction of a nearly uniform random string k and some public data h. Then, the secret k can be recovered from h and a second input w' as long as w' is close enough to w. Additionally, the value h should not reveal information about k.

Before we define discrete and continuous fuzzy extractors as in [BDHV07b], we first need to present the concepts of *min-entropy* and *statistical distance*.

Definition 2.4.1. Let A, B be two discrete random variables. The *predictability* of A is $\max_a(Pr[A = a])$ and the *min-entropy* of A, $H_\infty(A)$ is

$$H_\infty(A) = -\log(\max_a(Pr[A = a])).$$

Given $b \leftarrow B$, the conditional predictability of A given B is

$$H_\infty(A|B) = \max_a(Pr[A = a|B = b]).$$

Definition 2.4.2. Let A, B be two random variables. The *statistical distance*, or Kolmogorov-Smirnov statistic, between the variables A and B is given by

$$\text{SD}(A, B) = \sup_x |P[A = x] - P[B = x]| .$$

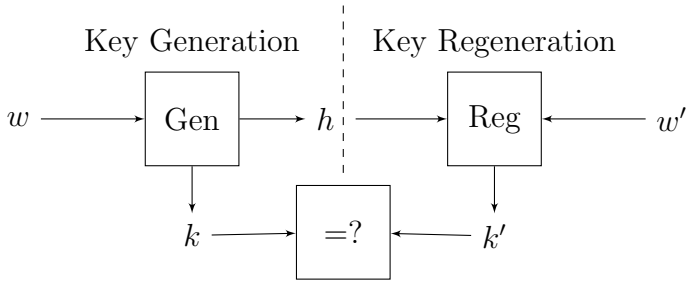

Figure 2.1: A general (discrete) fuzzy extractor. h is the public helper data and k is the key generated using the input w. A key is regenerated using a new input w' and the helper data h.

The notion of min-entropy encodes how many nearly uniforms bits can be extracted from A in the worst case. The notion of statistical distance, on the other hand, is a measure of the difference between the two random variables. The smaller that distance is, the more similar the two variables are.

With this we can define a fuzzy extractor as in [BDHV07b]. We denote by $\mathcal{U}_l$ the uniform distribution over $\{0,1\}^l$.

Definition 2.4.3. [BDHV07b] Let M be a discrete metric space with metric d. A (discrete) (M, m, l, t, ϵ)-*fuzzy extractor* is a pair of randomized procedures **Gen** and **Reg** such that:

1. The (necessarily randomized) generation procedure **Gen** takes $w \in M$ as input and outputs a pair of strings (k, h), where $k \in \{0,1\}^l$ is the private extracted string and $h \in \{0,1\}^*$ is a public string. **Gen** is such that, for all distributions W over M with min-entropy $H_\infty(W) \geq m$ and independant variables $(k, h) \leftarrow \mathbf{Gen}(W)$, it holds that $\mathrm{SD}((k, h), (\mathcal{U}_l, h)) < \epsilon$.
2. The regeneration procedure **Reg** takes $w' \in M$ and $h \in \{0,1\}^*$ as inputs and outputs a string $k \in \{0,1\}^l$ in such a way that for any $w, w' \in M$ such that $d(w, w') \leq t$ and any possible pairs $(k, h) \leftarrow \mathbf{Gen}(w)$ it holds that $k = \mathbf{Reg}(w', h)$.

The **Gen** procedure is such that any key k obtained from an input w appears uniform, even if the string h (sometimes called the *helper data*) is revealed. This provides security as, if ϵ is small enough, h does not reveal anything about k.

The **Reg** procedure ensures that a key k obtained from w can be recovered with an input w' if $d(w, w') \leq t$, while there is no guarantee of correctly regenerating k if $d(w, w') > t$.

Since the definition of min-entropy does not translate well into a continuous distribution setting, some work is required to adapt the definition of fuzzy extractor. A good solution is to discretize the distribution by quantizing it. Let χ_g be the global distribution of the population of users and let D_g be the quantized distribution of χ_g. With this discrete distribution, we can get a meaningful min-entropy by setting $m = H_\infty(D_g)$.

The authors of [BDHV07b] present an upper bound on the statistical distance ϵ to $\mathcal{U}_l$ in relation to both l and the min-entropy:

$$\epsilon(m, l) = \begin{cases} 0 & \text{if } l = m, \\ 2^{-l} & \text{if } l - 1 < m < l, \\ 2^{-m} - 2^{-l} & \text{if } m \leq l - 1. \end{cases}$$

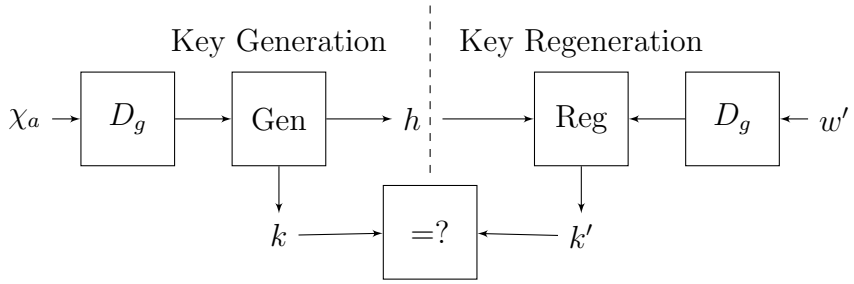

Figure 2.2: A continuous source fuzzy extractor, where the global distribution space is discretized by D_g. A key is generated using information on a user's distribution χ_a, while the key is regenerated from a measurement (or sample) w' of χ_a.

Each user is described by a distribution χ_a and elements of D_g are assigned to the particular users they are most likely to represent. Thus, some quantization errors are bound to happen. Let e_{FRR} be the probability of falsely rejecting a user due to quantization error.

Definition 2.4.4. [BDHV07b] A (χ_g, m, l, e_{FRR})-*continuous source fuzzy extractor* is a pair of randomized procedures **Gen** and **Reg** such that:

1. The (necessarily randomized) generation procedure **Gen** takes χ_a as input and outputs a pair of strings (k, h), where $k \in \{0,1\}^l$ is the private extracted string and $h \in \{0,1\}^*$ is a public string. **Gen** is such that, for any user distribution χ_a, if $(k, h) \leftarrow \mathbf{Gen}(\chi_a)$, it holds that $\mathrm{SD}((k, h), (\mathcal{U}_l, h)) < \epsilon(m, l)$.
2. The regeneration procedure **Reg** takes a measurement w' sampled from χ_a and $h \in \{0,1\}^*$ as inputs and outputs a string $k = \mathbf{Reg}(w', h) \in \{0,1\}^l$, where $(k, h) \leftarrow \mathbf{Gen}(\chi_a)$, with probability equal to the detection probability $p_d = 1 - e_{FRR}$.

We note that since $m = H_\infty(D_g)$, the choice of how the global distribution χ_g is quantized to D_g is quite important to the fuzzy extractor. As mentioned earlier, the authors of [PvdG16] instantiated a continuous source fuzzy extractor using a lattice code $\mathcal{L}$ to determine D_g. Their approach is to build D_g by decoding values of χ_g to the nearest lattice point of $\mathcal{L}$, with $\det(\mathcal{L}) = 1$.

Definition 2.4.5. [Lattice Fuzzy extractor from [PvdG16]] Let χ_g be the global distribution of the features, χ_a be the distribution of features of a user and $\mathcal{L}$ be a lattice with generator matrix $\mathbf{G}$. Furthermore, let D_g be the distribution χ_g quantized by $\mathcal{L}$ with $H_\infty(D_g) = m$. The procedures **Gen** and **Reg** are defined as follows:

1. **Gen** takes χ_a as input and outputs a pair of strings (k, h), where $k \in \{0,1\}^l$ is the private extracted string and $h \in \mathbb{R}^n$ is a public vector.
   The string k is the string representation of a randomly selected point $p \in \mathbb{Z}^n$, and set $r = \mathbf{G}p$ to be the associated point on the lattice.
   The public value h is obtained from $h = r + x_a$, with x_a being a feature sampled from χ_a
   It must hold that $\mathrm{SD}((k, h), (\mathcal{U}_l, h)) < \epsilon(m, l)$.

2. **Reg** takes a measurement y sampled from χ_a and $h \in \mathbb{R}^n$ as inputs and outputs the correct string k with probability equal to the detection probability $p_d = 1 - e_{FRR}$. This is done by decoding $h - y$ to its nearest lattice point in $\mathcal{L}$, represented by $p' \in \mathbb{Z}^n$. k is recovered as the string representing p'.

Finally, the strength or entropy of the key derived from a fuzzy extractor relies on the information content of the inputs. This dictates how and where the key can be used, since a key containing only 40 bits of entropy should not be used in a setting requiring 128 or 256 bits of security. For example, using a low entropy biometric key only to authenticate a user would be no worse than simply using classical biometric recognition, but such a key should not be blindly used as a symmetric cypher key to encrypt many communications.

This is particularly relevant with facial images, as they were shown in [YA10] to contain an average amount of information of around 40 to 60 bits. To generate a strong cryptographic key from such biometrics, it is necessary to increase the final entropy of the generation scheme.

For instance, this can be done by introducing additional randomness into the key generation like we do in this thesis, or by combining the resulting key with various other factors. This later option could be generating a key from multiple biometrics using one or more dedicated fuzzy extractors or by adding a PIN or password in the key generation and verification procedures.

Chapter 3

Fuzzy Extractor for Faces

In this chapter we present our improvements to the fuzzy extractor for faces presented by Sutcu, Li and Memon in [SLM09]. The original scheme has various privacy, security and usability issues when implemented directly. Some of these issues include the necessity of enrolling all or most users as the set up phase of the system and the randomization procedure not providing sufficient privacy and security.

In fact, when tested on our chosen subset of the AR database the original scheme performed very poorly when randomization is used to protect the biometric templates, with no user ever authenticating successfully. We thus propose various improvements that make the scheme more practical, more secure and more private.

In this chapter we first go over the fuzzy extractor for faces presented by Sutcu, Li and Memon in [SLM09] in section 3.1. In section 3.2, we provide some analysis on its security and propose improvements on some of the weaknesses. Finally, we describe the improved scheme in section 3.3 and we give test results using the AR data set in section 3.4 for both our improved scheme and the original scheme.

3.1 Sutcu et al.'s approach

The approach in [SLM09] to deal with the continuous data of the extracted features is by using a *quantizer*. A quantizer is a function $Q : \mathcal{U} \to \mathcal{M} \subset \mathcal{U}$, with $\mathcal{U}$ a metric space that acts as the generalization of a rounding function. That is, for any $u \in \mathcal{U}$, $Q(u)$ represents the closest value $m \in \mathcal{M}$ to u.

In practice, we will look at situations where $\mathcal{U}$ is $\mathbb{R}^n$ and where $\mathcal{M}$ is some easily describable discrete set. We note that implementing a quantizer Q is closely related to decoding an error-correcting code.

Consequently, the fuzzy extractor takes $Q(x)$, instead of the biometric x, as input to **Gen** and **Rep**. This allows for consistent key extraction when dealing with continuous inputs.

The process can be divided into three main phases: the setup phase, the enrolment phase, and the verification phase. The setup phase consists of computing and choosing various values from which the individual user's parameters will be derived.

In the enrolment phase for a given user, many biometrics are taken as inputs to generate a user-specific quantizer. The quantizer and the input biometrics are then used to generate a secret (in the form of a cryptographic key) and some helper data. Privacy is thus achieved in exchange for an increased number of biometric captures at enrolment.

In the verification phase, a biometric is provided along with a claimed identity and the helper data. The quantizer associated with this identity is then applied to the biometric and a secret is recovered. If the secrets are identical, the biometric matches the claimed user. If not, we conclude the biometric came from a different user.

Before we present the extractor for faces given in [SLM09], we will go over some notation. We set $\mathcal{B}$ as the space of captured biometrics (after the pre-processing). Let $f_E : \mathcal{B} \to \mathbb{R}^n$ be a (possibly randomized) feature extractor and let $Q : \mathbb{R}^n \to C \subset \mathbb{R}^n$ be a quantizer.

For example, $\mathcal{B}$ can be the set of gray-scale images with dimension 400x500 pixels, f_E can be a trained PCA extractor and Q can be a rounding function on each component.

Finally, if $b \in \mathcal{B}$ is a biometric, we will denote by $x = f_E(b)$ the extracted features, or the biometric template, of b. We will set n as the number of components of x, or the dimension of x.

In their paper, Sutcu et al. tested their fuzzy extractor with a randomized feature extractor unique for each user. They used PCA, followed by a randomization by a n-by-n matrix R_i, where each entry of R_i is i.i.d. uniformly in $[-1, 1]$. That is, for the i^{th} user, their feature extractor f_E^i is :

$$x = f_E^i(b) = R_i f_E(b),$$

where $R_i \in \mathrm{Mat}_{n,n}([-1, 1])$ and f_E is a PCA extractor.

3.1.1 The setup phase

Suppose we have I users and denote by U_i the i^{th} user. Furthermore, let $x_i = \{x_{i,1}, \ldots, x_{i,k}\} \in \mathbb{R}^n$ be the extracted features of a biometric from U_i.

Assume that, for each user U_i and for each component $j = 1, \ldots, n$, we have that $x_{i,j}$ fall into some range of values $[\min_{ij}, \max_{i,j}]$. For all components $j = 1, \ldots, n$, we set $\overline{x}_{i,j} = \frac{\max_{i,j}+\min_{i,j}}{2}$ as the *mid-point* of the interval and $\delta_{i,j} = \frac{|\max_{i,j}-\min_{i,j}|}{2}$ as the radius of the interval.

Under these assumptions, this means that if y is a biometric for U_i, we have that $y_j \in [\overline{x}_{i,j} - \delta_{i,j}, \overline{x}_{i,j} + \delta_{i,j}]$.

In practice, it is impossible to know the values of all these intervals for each user. There might not even be bounds to these intervals if the user biometric follows a normal distribution, as modeled in [BDHV07b]. If the biometric does follow a normal distribution

or is unbounded, we instead define $[\overline{x}_{i,j} - \delta_{i,j}, \overline{x_{i,j}} + \delta_{i,j}]$ as the bounded intervals that are likely to contain most biometrics from the user (for a suitable definition of "most").

In both cases, the values for $\max_{i,j}$ and $\min_{i,j}$ can be approximated empirically from a set of biometrics belonging to each user as the maximum and minimum values, respectively, of each of the n components of the set of extracted features.

From this information on the users' biometrics, Sutcu et al. construct a *global codebook* from which each user's individual quantizer will be derived. The global codebook is defined as $\mathcal{C} = C_1 \times \cdots \times C_n$, where for each component $j = 1, \ldots, n$,

$$C_j = \{\mathrm{MN}_j - r_j, \mathrm{MN}_j - r_j + \delta_j, \ldots, \mathrm{MN}_j - r_j + L_j\delta_j\}$$

and where $\mathrm{MN}_j = \min_i(\overline{x_{i,j}} - \delta_{i,j})$, $r_j \in \mathbb{R}$ is a randomly chosen value, $\delta_j < \min_i(\delta_{i,j})$ and $L_j \in \mathbb{Z}_{>0}$ is such that $\mathrm{MN}_j - r_j + L_j\delta_j > \max_i(\overline{x_{i,j}} + \delta_{i,j})$. In their paper, the authors suggest taking $\delta_j = \alpha_j \min_i(\delta_{i,j})$, for some scaling parameter α_j.

In other words, MN_j is the smallest value of $x_{i,j}$ observed over all the biometrics of all the users, δ_j is smaller than any observed radius, r_j is a random offset to hide the values of the biometrics, and L_j is chosen so that the global codebook covers all the observed biometrics.

The global codebook and these values are chosen so that we can build quantizers for each users by taking a well chosen subset of $\mathcal{C}$.

Finally, we note that for the construction of a good global codebook that will fit all the possible users, we need interval information from many, if not all, the users that will enrol in the system. This makes it so that the enrolment phase and the setup phase need to happen simultaneously for a good number of users.

3.1.2 Enrolment phase

The enrolment step is where the secret for each user is generated. Consider the i^{th} user, U_i, and let the values $\overline{x}_{i,j}$ and $\delta_{i,j}$ be as in the setup phase. The set of values $\overline{x}_{i,j}, \delta_{i,1}, \ldots \delta_{i,n}$ is called the *user's template*.

For this step, we first compute the *user codebook* $C^i = C^i_1 \times \cdots \times C^i_n \subset \mathcal{C}$, where

$$C^i_j = \left\{ \mathrm{MN}_j - r_j + k(2d_{i,j} + 1)\delta_j \in C_j \,\middle|\, k \in \mathbb{N}, \text{ and where } d_{i,j} = \left\lceil \frac{\delta_{i,j}}{\delta_j} \right\rceil \right\}.$$

We note that $C^i_j \subset C_j$ is chosen such that the distance between two points in this set is at least $2\delta_{i,j} + 1$. This is done by choosing one out of every $2d_{i,j} + 1$ elements in C_j. This codebook allows an easy construction of a quantizer $Q^i : \mathbb{R}^n \to C^i$, where $Q^i(x) = c = (c_1, \ldots, c_n)$ is the element $c \in C^i$ that is the closest to x. That is,

$$\forall j = 1, \ldots n, |c_j - x_j| = \min_{c' \in C^i_j}(|c' - x_j|).$$

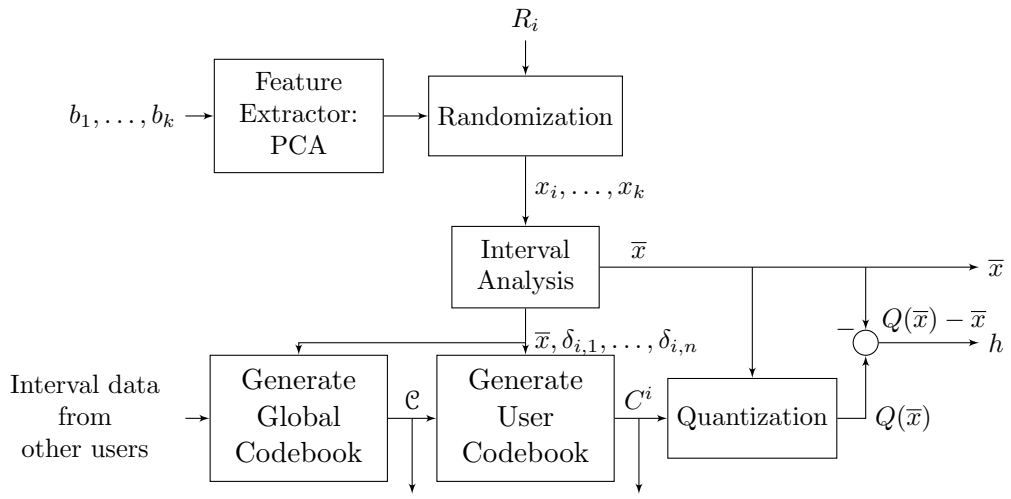

Figure 3.1: Enrolment process for a user U_i in Sutcu et al.'s original scheme. This process is done in parallel for many users at the setup phase to generate the global codebook. The random matrix R_i, the global codebook $\mathcal{C}$, and the user codebook C^i are kept in a secure storage by the system. h is the public helper data. $\overline{x}$ is the secret. The biometrics $b_1, \ldots, b_k$ are discarded after enrolment.

With the quantizer constructed, let $\overline{x}_i = (\overline{x}_{i,1}, \ldots, \overline{x}_{i,n})$ be the mid-point vector of the user and set the helper data h as the difference between the secret and its quantization :

$$h = Q^i(\overline{x}_i) - \overline{x}_i.$$

The helper data h itself does not reveal anything about the biometrics of the user. However it is essential to rebuild $\overline{x}_i$ at a later time.

The secret is $\overline{x}_i$, which can be hashed to produce a secret key, and the public helper data is h. The codebook C^i needs to be stored for future verification. A diagram of the enrolment process can be found in Figure 3.1.2.

3.1.3 The verification phase

A user can try to authenticate to the system by claiming to be a certain user U_i by providing a biometric y and the helper data h. The verification is done by computing

$$\overline{x}'_i = Q^i(y) - h = Q^i(y) - (Q^i(\overline{x}_i) - \overline{x}_i).$$

If y is a biometric such that for $j = 1, \ldots, n$ we have that $y_j \in [\overline{x}_{i,j} - \delta_{i,j}, \overline{x_{i,j}} + \delta_{i,j}]$, then we get that $Q^i(y) = Q^i(\overline{x}_i)$ and $\overline{x}'_i = \overline{x}_i$. This allows us to reconstruct the secret.

If the secret is rebuilt correctly, we authenticate the user. Otherwise, the authentication fails. A diagram of the authentication process can be found in Figure 3.1.3.

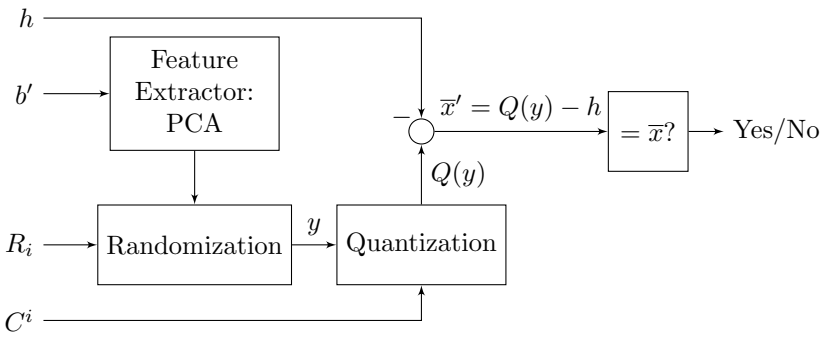

Figure 3.2: Authentication process for a user U_i in Sutcu et al.'s original scheme. The new biometric b' and the helper data h are provided by the user trying to authenticate. The randomization matrix R_i and the user codebook C^i are taken from the system's secure storage.

3.2 Implementation Improvements

Our goal was to implement a privacy protecting facial recognition system and we chose to use Sutcu et al.'s framework and scheme. We chose this particular scheme as it is one of the only fuzzy extractor specifically designed and tested for facial recognition, instead of being designed for general continuous features like [BDHV07b] or [PvdG16]. A direct implementation of Sutcu et al.'s scheme has various possible security and implementation concerns. Therefore, we present in this section those findings as well as our proposed solutions to these issues.

We begin by looking at security concerns with the randomized user templates that are obtained through a $n \times n$ uniform random matrix R. This choice of R brings various issues.

First of all, there are no guarantee that such a choice of R will preserve distance or clustering of the input data. This means that there is the possibility that authentication can fail after randomizing the biometrics even if the biometrics were similar enough to authenticate the user without randomization. It also means that other users could be falsely authenticated.

The second problem is that, as explained in [WP10], using a square matrix can allow an attacker to rebuild biometrics (or the PCA features) if the matrix R is obtained. This means that if the user template is obtained and the user's matrix R is leaked, an attacker can reconstruct (or approximate) the average face of this particular user.

To prevent such a reconstruction from a single feature vector, [WP10] shows that the matrix used in the random projection, R, has to be $m \times n$, with $m \leq \frac{n}{2}$. Such a choice of $m < n$ makes the randomization fall into the setting of random projection.

The next set of security concerns is related to the codebooks used in the scheme. In our experimentation and direct implementation of the scheme, the codebooks of each user were often quite limited in size, with some C^i_j sometimes having as few as 1 or 2 elements. This can prove to be a problem if a user's codebook is leaked. Given a relatively small codebook C^i and the public helper data h, a brute force attack can easily be carried out

by testing $q - h$ for all $q \in C^i$. A successful attack leaks some of the user's template, since $\overline{x}$ would be recovered. The user's $\delta_{i,j}$ values can also be approximated from the codebook. Furthermore, if the user codebook is small in size, then it is more likely that any input will quantize to the correct value, increasing the rate of false positives of the system.

A similar situation arises if the global codebook $\mathcal{C}$ is compromised. Although bigger than each individual user's codebook, the size of the global codebook in our experimentation can still be small enough that a brute force attack is viable. This could compromise all the enrolled users. Furthermore, depending on the choices of the parameters α_j, r_j and L_j at the setup phase, the global codebook might reveal partial information about the biometrics of certain enrolled users.

Finally, we ran into an important concern for an implementation of the system. This concern is that the scheme requires many users to be enrolled at the setup phase, since the global codebook is generated using data from each user. This makes such a scheme less desirable if some or most users using the system are expected to enroll after the setup phase.

3.2.1 Proposed Improvements

To solve these issues, we propose a few changes to the scheme. Our first improvement is with the randomization matrix R_i, where we choose to use random projection as the randomization step. That is, we replace the uniform $n \times n$ matrix R with a $m \times n$, $m \leq \frac{n}{2}$, matrix with i.i.d Gaussian elements as in algorithm 2.1.4. Such a choice of matrix R_i preserves the distance of the input biometrics while providing security, privacy, and changeability to the scheme through known results ([WP10]). By using PCA followed by such a random projection, we obtain a PCARP feature extractor.

The second improvement we propose is to make the global codebook unbounded in size and make it so that it does not reveal any information about the enrolled users. Recall that the global codebook was defined as $\mathcal{C} = C_1 \times \cdots \times C_n$, where for each $j = 1, \ldots, n$,

$$C_j = \{\mathrm{MN}_j - r_j, \mathrm{MN}_j - r_j + \delta_j, \ldots, \mathrm{MN}_j - r_j + L_j\delta_j\},$$

with MN_j being the smallest observed value among the users enrolled at setup, r_j being a random value, and $\delta_j < \min_i(\delta_{i,j})$.

If we look at each of the parameters, we see that MN_j and L_j are used and chosen so that all the values for each user's biometrics will be included in the interval $[\mathrm{MN}_j - r_j, \mathrm{MN}_j - r_j + L_j\delta_j]$. These boundaries are removed if the codebook is made unbounded. The parameter r_j is there to blur and obfuscate the MN_j values.

Finally, the last parameter δ_j is chosen small enough to allow a user U_i to generate a user codebook that has a minimum distance as close to $2\delta_{i,j} + 1$ as possible, because if the minimum distance is significantly greater than $2\delta_{i,j} + 1$, we increase the risk of false acceptance for that particular user.

With this in mind, we propose to construct the global codebook as follows. We first arbitrarily choose δ_j to be much smaller than what we expect $\min_i(\delta_{i,j})$ to be. A suitable

value can be empirically determined before the setup phase by testing on an available database or by previous knowledge of how the features of users may behave under certain feature extractor.

For example, we tested a PCARP extractor with PCA reducing to 200 components and RP down to 100 on the AR data set [MB98] for many different random matrix seeds, then looked at the smallest radius values observed across all components and users. We found that, even when using different RP extractors, the smallest observed radius, $\min_i(\delta_{i,j})$, was approximately 200 for all j. In such a situation, we saw that choosing $\delta_j = 3$, or a δ_j about a hundredth of the minimal observed value, would appear to be a small enough that the condition $\delta_j < \delta_{i',j}$ is satisfied for any future user $U_{i'}$ in any possible tests. In a similar fashion, when testing with RP-RP, we chose a $\delta_j = 0.3$, as the smallest values observed were smaller at approximately 20.

The particular choice of the values for δ_j will be dependant on the feature extractor used, and what images are given to the system (with regards to size, preprocessing, etc.). The value only needs to be small enough so that no user will ever have a interval radius that is less than δ_j, or the performance of the scheme will be affected for this particular user.

With δ_j chosen, we sample r_j uniformly from $[0, \delta_j)$ and we build the global codebook by setting

$$C_j = \{r_j + \alpha\delta_j | \alpha \in \mathbb{Z}\}.$$

for all values of j. Such a construction provides us with a non-bounded global codebook, does not reveal any information about the users, and it can be easily be stored as the two vectors $\mathbf{r} = (r_1, \ldots, r_n)$ and $\delta = (\delta_1, \ldots, \delta_n)$. The codebook is then effectively unbounded in size, although it will in practice be bounded by the possible outputs of the feature extractor. In a computer implementation, the value r_j can have a finite precision (i.e. 32-bits floating points number).

The quantizer for the global codebook, $Q : \mathbb{R}^n \to \mathcal{C}$, can be obtained by

$$Q(x) = Q((x_1, \ldots, x_n)) = (Q_1(x_1), \ldots, Q_n(x_n)) \in \mathcal{C},$$

with

$$Q_j(x_j) = \left\lfloor \frac{x_j - r_j}{\delta_j} \right\rceil \delta_j + r_j$$

where $\lfloor \cdot \rceil$ is the rounding function. It is easy to check that $Q(x)$ is the closest point in $\mathcal{C}$ to x.

The codebook for user U_i is constructed from $\mathcal{C}$, $\overline{x_i}$ and $\delta_{i,1}, \ldots, \delta_{i,n}$ by setting

$$\begin{aligned} k_j &= 2 \left\lceil \tfrac{\delta_{i,j}}{\delta_j} \right\rceil + 1, \\ t_{i,j} &= Q_j(\overline{x}_{i,j}) - \left\lfloor \tfrac{Q_j(\overline{x}_{i,j}) - r_j}{k_j} \right\rceil k_j \delta_j. \end{aligned}$$

The user codebook is then

$$C_j^i = \{t_{i,j} + \alpha k_j \delta_j | \alpha \in \mathbb{Z}\}.$$

Like for the global codebook, we only need to store the two vectors $\mathbf{t}_i = (t_{i,1}, \ldots, t_{i,n})$ and $\vec{\rho_i} = (\rho_{i,1}, \ldots, \rho_{i,n}) = (k_1\delta_1, \ldots, k_n\delta_n)$.

This construction is similar to the one presented by [SLM09], in that C_j^i is obtained by taking one out of every k_j elements from C_j to give us a good minimum distance. For each j, the value of $t_{i,j}$ is, by construction, the only codeword of C_j^i such that $t_{i,j} \in [0, k_j\delta_j)$. That is, the $t_{i,j}$'s are the smallest non-negative codewords of C_j^i.

This choice of $\mathbf{r}_i$ and $\vec{\rho_i}$ allows us to easily obtain a quantizer Q^i for this user codebook in the same way as for the global codebook :

$$Q^i(x) = Q((x_1, \ldots, x_n)) = (Q_1(x_1), \ldots, Q_n(x_n)) \in \mathcal{C},$$

with

$$Q_j^i(x_j) = \left\lfloor \frac{x_j - r_{i,j}}{\delta_j} \right\rceil \delta_{i,j} + t_{i,j}.$$

The third change we propose is to prevent any efficient brute force attacks on the user codebooks. For this, we add a random shift in the quantization step, just before generating the secrets. Recall that the secret generated in the fuzzy extractor was $\overline{x} = Q^i(\overline{x}) + h$, with h being the public helper data. The intent is to add randomization without affecting $Q^i(\overline{x})$.

Let k_{shift} be a key to generate the random integers

$$p_1, \ldots, p_n \in \{-\lambda, \ldots, -1, 0, 1, \ldots, \lambda\}$$

for some positive integer λ. Then set $\mathbf{v}_{k_{shift}} = (p_1\rho_{i,1}, \ldots p_n\rho_{i,n})$ and we compute the secret as

$$s = Q^i(\overline{x}) + \mathbf{v}_{k_{shift}} + h.$$

This amounts to moving away from $Q^i(\overline{x})$ by p_j codewords in the j^{th} component. Such a shift ensures that if only the user codebook is leaked, the search space to find the secret with a brute force search will be at least of size $(2\lambda)^n$. The value of λ can then be chosen based on n to provide sufficient security. For example, if $n = 100$, choosing $\lambda = 2^{12}$ gives us a search space of size at least 2^{1300} codewords. This addition also has the side effect of increasing the final entropy of the key, provided that k_{shift} stays secret.

The shift also renders a brute force attack inefficient if only the global codebook is leaked, since the global codebook is bigger than any user codebook. This added security comes at the cost of storing one additional key for each user.

Our next suggested change is to randomize the public helper data h. Instead of setting $h = Q^i(\overline{x}) - \overline{x}$, we chose to uniformly randomly sample $h \leftarrow U \subset \mathbb{R}^n$, for some bounded set U. This makes h completely random and unrelated to any biometrics at all. Although using $h = Q^i(\overline{x}) - \overline{x}$ does not leak biometric information by itself it may cause issues if more data is leaked. More specifically, we propose this change to add some more privacy to two situations at no additional cost to the system: one where the codebook is leaked and one where the key is obtained from the encrypted token (for example).

In the first situation, already covered above, a brute force search on a leaked codebook using h can allow the reconstruction of the secret. When $h = Q^i(\overline{x}) - \overline{x}$ is used, then the attacker recovers $\overline{x}$ directly if no random codeword shift is used or if k_{shift} is also leaked. In the case where h is random, the recovered secret s is not directly related to the biometrics.

In the second situation, we consider the case where an attacker "inverts" the key generation procedure to recover the secret. This could, for example, be done by using a previously unknown vulnerability in the used hash function that allows one to invert it. In a similar fashion, $\overline{x}$ could be recovered if h is not a random vector.

Finally, we note that throughout the updated construction, there are only a few parameters that can be chosen or modified : the target dimension m of the PCARP feature extractor, the intermediate dimension n_2 of the extractor and the choice of $\vec{\delta}$ in the construction of the global codebook. None of these three parameters can be changed in the system after being chosen at the setup phase and they do not allow the fine tuning of the FPR or FNR for any given application.

Consequently, to allow for some amount of fine tuning of the system, we introduce an additional scaling parameter $\vec{\beta} = (\beta_1, \ldots, \beta_n)$, with $\beta_i > 0$ in the construction of the user codebooks and the quantizer $Q^i(x)$. The β_i are used to scale the computed values of $\delta_{i,j}$ which are derived from a sampled set of biometrics: instead of using $\delta_{i,j}$ for the user codebook, we use $\beta_i \delta_{i,j}$. Choosing $\beta_i < 1$ will increase the strictness of the system and lower the FPR, while choosing $\beta > 1$ will decrease the strictness and increase the FPR.

We end this section by listing all the changes discussed previously.

1. We change the construction of the random matrix R so that it becomes an RP extractor.
2. We change the construction of the global codebook so that it is unbounded in size and does not depend on users being enrolled at the setup phase. The construction of the user codebooks reflects the change in the global codebook.
3. A random shift in codewords at the enrollment step is added to protect against brute force attack if a codebook is compromised. In addition, this random shift, if safely kept secret, increases the entropy of the extracted key above what might be extracted from only the biometrics.
4. We make the helper data h be random and unrelated to the user's biometrics.
5. We add a scaling parameter $\vec{\beta}$ (global or user-based) to allow the fine tuning of FPR and FNR in the system by allowing control of the strictness of the authentication.

Many of those changes allow the system to be set up without requiring users to be pre-enrolled.

Security Analysis

These changes make the scheme more secure under the attacker model from Section 2.2.1, as explained in this section. Recall that we consider a semi-passive attacker that

1. has full knowledge of the scheme,
2. has access to all the public data,
3. can get access through leaks, partially or totally, to the data that is stored or kept secret,
4. *cannot* influence or modify the scheme or any messages between entities, and
5. has *not* compromised any part of the system like the sensors.

An attacker is successful if they can either recover a user's biometric or if they can authenticate as the user. We look at the improvements to the privacy and security of the scheme compared to Sutcu et al.'s original scheme.

We first consider the situation where only the public data is obtained, that is the helper data h and the token encrypted using the secret key. In this situation, the template and the derived key are secure under the assumptions that the hash function and the chosen cryptosystem are themselves secure.

An attacker can also theoretically use knowledge of the entropy and information contained in the biometrics features to gain an advantage in finding or guessing the key, since the entropy of the final key is dependant on the entropy of the input biometrics. This is partly alleviated in our improved scheme as we introduce additional independent noise and entropy to the hashed key with the random codeword shift. Provided that the random codeword added using k_{shift} stays secret, this can remove any advantages to guessing the key if the parameter λ for the shift is chosen large enough.

We now consider the situation where the user codebook is leaked. In this situation, an attacker can use h and search the codebook until the secret key can be recovered instead of simply guessing the key. In the original scheme, the codebooks were often small and finite in size, making this a viable approach.

In the improved scheme, the unbounded nature of the codebooks along with the added random codeword shift can make the search at least as hard as guessing the key by choosing λ large enough that the search space for the correct codeword is at least as large as that of the secret key.

If the random codebook shift is leaked along with the codebook, the key and a randomized biometric template may be obtained through a smaller search space on the codebook. The size of the search space will depend on the feature extractor used. The use of RP, as shown in [WP10], can then protect the user's biometrics and make it changeable and cancellable, even if the RP matrix is also leaked.

As such, if the seeds or random elements are kept secret, the scheme is more private and secure than traditional biometric authentication system, since no biometrics are stored. These added seeds also make it more secure than the original scheme by Sutcu et al. Furthermore the addition of the codeword shift and the use of RP also makes our improved scheme more secure to partial leaks than the original scheme.

This increase in security and privacy comes at the very small cost of securely storing 3 pseudo-random number generator seeds and regenerating the RP matrix and vector at authentication, or by securely storing the RP extractor and codeword shift. This amounts

to storing two keys or vectors more than the original scheme by Sutcu et al., since the original scheme used a uniform matrix for randomizing the templates.

3.3 The updated scheme

In this section, we collect the changes from the previous section and present our updated scheme. Throughout this section, let n be the input size and m be the final size of the extracted features. We will present the scheme using a PCARP feature extractor, but it can easily be modified to only use RP. The PCA extractor in PCARP can also be replaced by any other feature extractor.

3.3.1 The setup phase

The setup phase consists of training the PCA extractor and setting the global parameters of the system. Let $n_2 \geq 2m$ be the target dimension for PCA and let $T = \{b_1, \ldots, b_l\} \subset \mathbb{R}^n$ with $l > n_2$ be a set of training biometrics.

1. Compute a PCA feature extractor $f_{PCA} \leftarrow \text{TrainPCA}(T, n_2)$.
2. Generate the global codebook by choosing values for $\vec{\delta} = (\delta_1, \ldots, \delta_m)$, then randomly generate $\mathbf{r} = (r_1, \ldots, r_m)$, $r_i \leftarrow [0, \delta_i)$.
3. Select threshold parameter t for the RP extractor.
4. Choose values for scaling parameter $\vec{\beta} = (\beta_1, \ldots, \beta_m)$.
5. Set N to be the number of images required to enroll a user.
6. Set λ as the security parameter for the codebook shift.

Store f_{PCA}, the global codebook $(\mathbf{r}, \vec{\delta})$, t, $\vec{\beta}$, N and λ as server parameters. We note that although we set $\vec{\beta}$ as a global parameter, it can be changed for each user at the enrollment step if the application requires it.

3.3.2 The enrollment phase

In the enrollment phase, a user U_i is registered into the system. It requires a set of enrollment images $E_i = \{b_1, \ldots, b_N\} \subset \mathbb{R}^n$ and three seeds, or keys, k_1, k_2, k_3 to be used in randomization.

1. Use the keys k_1, k_2 to generate a user specific RP extractor $f_{RP,i} \leftarrow \text{RPGen}(n_2, m, k_1, k_2, t)$.
2. Obtain the user-specific PCARP extractor $f_i(x) = f_{RP,i}(f_{PCA}(x))$.
3. Extract the features of the biometrics and obtain $(x_1, \ldots, x_N)$, with $x_j = f_i(b_j)$.
4. From $(x_1, \ldots, x_N)$, compute $\overline{x} = \frac{1}{N}\sum_i x_i$ and $\delta_j = \frac{1}{2}(\max_i(x_{i,j}) - \min_i(x_{i,j}))$ for $j = 1, \ldots, m$.

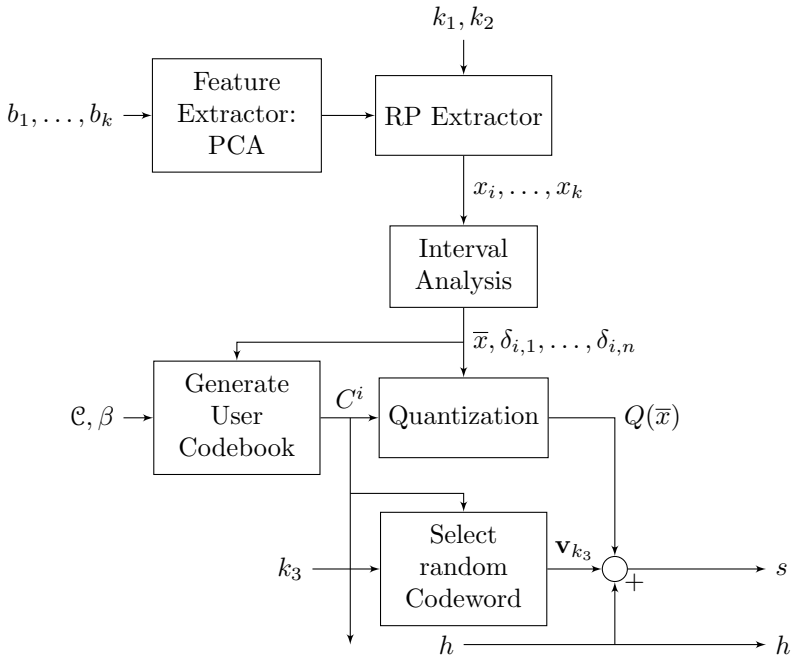

Figure 3.3: Enrolment process for a user U_i in our improved scheme. The global codebook $\mathcal{C}$ is a system parameter chosen at setup and the β parameter can be system or user specific. The keys (or seeds) k_1, k_2, k_3 used for randomization procedures are stored securely on the system, as is the user codebook C^i. The public helper data h is a random vector given to the user, while $s = Q(\overline{x}) + \mathbf{v}_{k_3} + h$ is the secret. The biometrics $b_1, \ldots, b_k$ are discarded after the enrolment is complete.

5. Generate the user codebook using $\overline{x}$, $\delta_1, \ldots, \delta_m$, the scaling factor $\vec{\beta}$ and the global codebook. The user codebook is the pair of vectors $(\mathbf{t}_i, \vec{\rho}_i)$ defined as

$$\begin{aligned} k_j &= 2\left\lceil \tfrac{\beta_j \delta_{i,j}}{\delta_j} \right\rceil + 1, \\ t_{i,j} &= Q_j(\overline{x}_{i,j}) - \left\lfloor \tfrac{Q_j(\overline{x}_{i,j}) - r_j}{k_j} \right\rfloor k_j \delta_j, \\ \vec{\rho}_{i,j} &= k_j \delta_j. \end{aligned}$$

6. Randomly sample the helper data $h \leftarrow \mathbb{R}^m$, and use the key k_3 to randomly sample $p_1, \ldots, p_n \in \{-\lambda, \ldots, -1, 0, 1, \ldots, \lambda\}$.
7. Compute the secret $s = Q^i(\overline{x}) + \mathbf{v}_{k_3} + h$ using the user codebook's quantizer, Q^i, where $\mathbf{v}_{k_3} = (p_1\vec{\rho}_{i,1}, \ldots, p_1\vec{\rho}_{i,m})$.

The user codebook $(\mathbf{t}_i, \vec{\rho}_i)$ and the keys k_1, k_2, k_3 are securely stored in the system. The helper data h is given to the user, and s can be used to generate an encryption key and encrypt a predetermined message α as $c_i = \mathrm{Enc}_s(\alpha)$; this ciphertext is stored for future authentication of the user. A diagram of this enrolment process can be found in Figure 3.3.2.

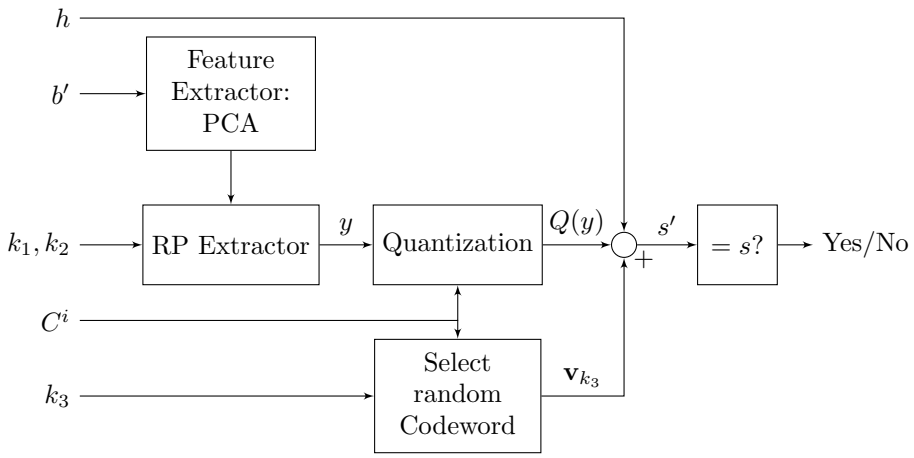

Figure 3.4: Authentication process for a user U_i in our improved scheme. The new biometric b' and the helper data h are provided by the user trying to authenticate. The randomization keys (or seeds) k_1, k_2, k_3 and the user codebook C^i are taken from the system's secure storage.

3.3.3 Authentication phase

In the authentication phase, a user U' claims to be a user U_i by providing a biometric b' and helper data h. The system accesses the stored data for user U_i : the three keys k_1, k_2, k_3, the user codebook $(\mathbf{t}_i, \vec{\rho}_i)$ and the ciphertext c_i.

1. Use the keys k_1, k_2 to generate the user specific RP extractor $f_{RP,i} \leftarrow \text{RPGen}(n_2, m, k_1, k_2, t)$.
2. Obtain the PCARP extractor $f_i(x) = f_{RP,i}(f_{PCA}(x))$.
3. Extract the features of the biometric and obtain $x' = f_i(b')$.
4. Use the key k_3 to regenerate $p_1, \ldots, p_n \in \{-\lambda, \ldots, -1, 0, 1, \ldots, \lambda\}$.
5. Compute $s' = Q^i(x') + \mathbf{v}_{k_3} + h$, where $\mathbf{v}_{k_3} = (p_1\vec{\rho}_{i,1}, \ldots, p_1\vec{\rho}_{i,m})$.
6. Verify if $\text{Dec}_{s'}(c_i) = \alpha$. If the equality holds, authenticate U' as user U_i. Otherwise, reject the authentication.

Figure 3.3.3 contains a diagram of this authentication process.

3.4 Experimental results

In this section we experiment with our improved version of the fuzzy extractor on a subset of the AR face data set [MB98]. We will compare it when using a PCARP feature extractor (as described in the previous section), as well as when using PCA with no randomization and finally using an RP-RP extractor. This RP-RP extractor uses a fixed global RP extractor to replace PCA, while keeping a user specific RP extractor to add randomness, privacy and changeability. We chose RP-RP instead of a single user-specific RP extractor for

efficiency purposes, since generating a huge RP matrix for each user can be computationally expensive.

The code for the experiments was written in Python using the NumPy [Oli06] and OpenCV [Bra00] libraries. OpenCV was used to import and work with the images, as well as provide an open source implementation of PCA. The rest of the code was original.

3.4.1 The Dataset

This research was linked to a project done with the Government of Canada that dealt with border crossing travellers at an airport, and this affected our choice of testing database. For testing the fuzzy extractors in this project, we needed an open and freely accessible dataset due to various resources restrictions. Furthermore, the dataset needed to have:

1. pictures had to resemble passport pictures (portrait, front facing, little to no accessories),
2. the pictures needed some time differential between some of them, if possible, and
3. a large number of users, each with a sufficiently large amount of pictures, to train a PCA extractor and enrol users with the fuzzy extractor.

With these restrictions, the AR data set was a great choice for this study since each user has a large number of images and it gives some amount of variation in each individual's faces. It contained a sufficient amount of well labelled images that allowed up to obtain a large enough subset of users and images to both train a PCA extractor with at least 200 components and enrol many users in the fuzzy extractor. Furthermore, the AR dataset contained images of user with a good amount of variation in the pictures necessary to obtain some information on each individual's biometric distribution.

The subset of images we used in our experiments was the set of users who, after filtering the images that fitted our requirements, still had a good number of images. The subset we kept was divided into two sets. The first set contained 67 persons with five images (neutral, angry, neutral with left lighting, neutral with right lighting, neutral with lighting on both sides). The second set container 26 persons with seven images (the same five as before, with added neutral and angry taken two weeks later). The set of 67 persons was the training image set for the PCA extractor, while the set of 26 persons was the testing set used for enrolment/authentication.

The only preprocessing applied to these images was to convert them to gray-scale and to crop the images from 768×576 pixels to 432×576 pixels. The cropping was done to match the ratio of standard portrait images, such as passport pictures. The list of image names used in the two sets can be found in Appendix A.

We note that preprocessing techniques for facial recognition is something that has been well studied and is an area of research of its own, with many techniques generally implemented in facial recognition softwares. In this work, we only did minimal preprocessing to the images due to various resource and time constraints, as well as a lack of free or open

source code for extensive preprocessing of facial images. This meant that, with our limited resources, we lacked easy access to the state of the art in preprocessing.

As such, we opted to do only minimal preprocessing to compare our scheme. Additional preprocessing can only improve on the performance of the scheme, in a similar way that using a commercial feature extractor can improve facial recognition.

3.4.2 Test Methodology

The tests were done as follows. If PCA was used as the feature extractor, PCA was first trained on the set of 67 users. Since training PCA is deterministic, the PCA extractors were saved for ease of use. Then, we separate the images for each users of the testing set into two disjoint sets : the enrolment images and the authentication image(s).

Each test was done by first selecting:

1. the final size of the templates, or the number of components,
2. the enrolment and authentication image sets, and
3. the feature extractor used (PCA, PCARP, RP-RP).

Once the parameters are chosen, the users were then enrolled and we tried to authenticate each user using the authentication images of all users. We note that when RP is used, each user gets assigned their own random RP extractor. The result of each authentication is saved along with the "true" expected result.

To reduce the number of variables in subsequent tests, we generated two global codebooks for each template size, one when using PCA or PCARP with $\delta_j = 3$ for all j and one for RP-RP with $\delta_j = 0.3$ for all j. These global codebooks were saved and reused throughout all tests involving their respective feature extractors.

A fuzzy extractor authentication system is binary in nature (accept or reject) and provides no quantitative information on why an authentication image leads to an accept or reject. As such, to allow for measurements in our test, we compute a score for each authentication by computing how far away an authentication image's features y are from the enrolment codeword $Q^i(\overline{x})$ used to create the secret s. This score also allows us to look at equal error rate (EER) of the system.

This score is given by $score = \max_j\{\frac{|Q^i(\overline{x})_j - y_j|}{\rho_{i,j}}\}$. Intuitively, this score represents "how many codewords away from the 'correct' codeword the biometric y is, in the component that is the furthest". We note that if a score is less than 0.5, then the image y will successfully reconstruct the secret, while if the score is at least 0.5 then the secret is not successfully reconstructed. Furthermore, the score gives us an idea about how to select the scaling parameter β to control the false positive rate (FPR) and false negative rate (FNR) of the system. Indeed, the choice of β only affects the numerical value of the score, but does not affect the EER in our tests. As such, throughout all our tests β has a fixed value.

We chose to do the tests and comparisons for a final component size of 20, 30, 40, 50, 75 and 100. When using RP as a randomization step, we chose to reduce by half of the

first extractor, as was done in [WP10] to compare PCA and PCARP. This means that in the case of a PCARP or RP-RP, the target size of the first extractor is, respectively 40, 60, 80, 100, 150 and 200.

With regards to the images used in the test, each of the 26 users have seven (7) images. Each of those images was labelled from 0 to 6, and either one or two images were chosen for the authentication set, while the rest were used in the enrolment procedure. We choose the same image label for enrolment and authentication sets across all users of a test.

3.4.3 Original Scheme

We begin by testing Sutcu et al.'s original scheme on the AR dataset. This is so that we can have some level of comparison between the two schemes. The scheme was implemented as described in [SLM09] and Section 3.1 of this work.

The only liberty we took with the implementation is with the construction of the user codebook, where we used a similar approach to the one in our improved scheme to construct a suitable codebook. We still take one out of every few codewords from the global codebook, but we made sure that we "started" at the global codeword closest to $\overline{x}$ instead of at the smallest global codeword. This should improve the accuracy of the quantization.

The parameters we can control in the original scheme are : the final number of components of PCA, the enrolment and authentication images used, and the parameter α, which is used to compute the global codebook quantization step δ_j.

Our first test using this scheme yielded bad results. For any enrolment set of six images, any of the number of components tested and various values of α between 0.01 and 0.5 all gave an EER of around 0.5, with small differences due to the random matrices used and the chosen value of α. Furthermore, across all these test, no user was ever authenticated (either as a true positive or a false positive).

	Authentication Image						
Number of Components	0	1	2	3	4	5	6
20	0.083	0.115	0.077	0.077	0.051	0.062	0.023
30	0.083	0.115	0.077	0.077	0.051	0.062	0.023
40	0.128	0.066	0.077	0.080	0.042	0.009	0.038
50	0.135	0.103	0.115	0.069	0.038	0.077	0.069
75	0.115	0.077	0.095	0.103	0.058	0.060	0.049
100	0.145	0.154	0.115	0.149	0.077	0.045	0.038

Table 3.1: EER for different authentication sets of one image for the original scheme by Sutcu et al. The tests were done using only PCA as the feature extractor, no randomization, and with $\alpha = 0.15$.

We then did the same tests by foregoing the randomization. The scheme performed a lot better then, as can be seen in Table 3.1. This big difference in performance can be attributed to the randomization matrix used. The original scheme uses a square matrix

with uniform entries over $[-1, 1]$, and such a matrix offers no guarantee that the randomized data preserved their clustering. This translated into the observed performance where no authentications are observed when using the randomized data.

Although the scheme performs well without the randomization, not using it means that the templates are not protected and neither changeability nor privacy is provided. As we will see in the next section, our improvements make the scheme more usable.

3.4.4 The Improved Scheme

In this section, we experiment on our improved scheme to see how it performs.

Testing the enrolment and authentication sets

The first series of test involves looking at the effect of the enrolment set, both in content and size, on our improved scheme.

	Authentication Image						
Number of Components	0	1	2	3	4	5	6
20	0.077	0.115	0.077	0.077	0.063	0.062	0.038
30	0.115	0.095	0.098	0.115	0.062	0.022	0.022
40	0.152	0.077	0.077	0.089	0.046	0.009	0.038
50	0.115	0.078	0.115	0.077	0.038	0.077	0.077
75	0.115	0.094	0.115	0.115	0.057	0.069	0.068
100	0.134	0.154	0.126	0.154	0.077	0.049	0.052

Table 3.2: EER for different authentication sets of one image, when using PCA as the feature extractor. As can be seen, the enrolment images and authentication image used can greatly affect the error rates of the system. Some combinations performs badly (i.e.: image 0) and others perform better (i.e.: image 5 or 6).

	Authentication Images						
Number of Components	0,1	0,5	1,2	1,3	2,6	4,5	5,6
20	0.108	0.074	0.423	0.115	0.058	0.250	0.158
30	0.115	0.077	0.407	0.096	0.077	0.250	0.173
40	0.129	0.096	0.404	0.077	0.077	0.231	0.159
50	0.096	0.115	0.421	0.073	0.077	0.208	0.138
75	0.132	0.115	0.442	0.135	0.077	0.174	0.192
100	0.135	0.115	0.425	0.165	0.115	0.173	0.191

Table 3.3: EER for different authentication sets of two images, when using PCA as the feature extractor. With a smaller enrolment set of 5 images, the scheme generally performs worse than when using a set of size 6. Some combinations still seem to offer relatively good performance.

The results of this first test can be seen in Tables 3.2 and 3.3 when we use a PCA extractor with no randomization. As can be seen, the enrolment images used and the authentication image used can greatly affect the performance and accuracy of the scheme. In general, we see that a bigger enrolment set of size 6 performs better than a smaller set of size 5. We see similar results when using PCARP and RP-RP.

This is an expected result, as more or better selected images can provide more information about the underlying distribution of a user's biometric. This can be particularly true when using feature extractors like PCA that can be easily affected by the content of the images (like lighting or accessories).

The best number of images to use for enrolment, as well as their content, in a real application is something that should be studied further, as it will depend both on the feature extractors used as well as any additional preprocessing applied to the images. The same reasoning applies for what the authentication image should be.

Since the enrolment set can greatly affect the performance of the fuzzy extractor, we will only look at one particular set of enrolment images of size 6. This is done to better compare the effect of the other parameters on the performance of the fuzzy extractor.

Finally, we can compare the results of our non-randomized improved scheme in Table 3.2 with those of the non-randomized original from Table 3.1. As we can see, our improved scheme performs as well as, if not sometimes better than, the original scheme on the same dataset. This confirms that our changes to the construction of the codebooks, as well as making them unbounded, does not negatively affect the performance of the scheme. As will be seen in the next section, using an RP extractor to randomize the templates is a better option than using a square uniform matrix as the effect on the performance is relatively small.

Feature Extractors and Number of Components

The second series of tests was to compare the behaviour of the fuzzy extractor using PCA, PCARP and RP-RP. In particular, we can see in Table 3.4 the EER by final number of components of each feature extractor. In the case of PCARP and RP-RP, we perform the tests six times and we present the maximum and minimum observed EER for each number of components.

We can consider the EER for the PCA feature extractor as a baseline performance of the scheme with this particular image data set, since no randomization is introduced during the tests. This gives us some insight into how accurate the scheme could be given this particular data set. This allows us to observe the effect of using RP as a template randomizer technique.

From these tests we can see that, as expected, the performance of both PCARP and RP-RP are affected by the user's random matrices. Indeed, the variance in EER for PCARP across different tests is due solely to the RP extractor each user received, as all the other parameters were fixed. For PCARP, the randomization generally presents itself

Number of Components:	20	30	40	50	75	100
PCA EER	0.058	0.021	0.010	0.074	0.067	0.047
PCARP min EER	0.067	0.074	0.074	0.040	0.037	0.037
PCARP max EER	0.111	0.105	0.132	0.100	0.091	0.074
RP-RP min EER	0.077	0.074	0.074	0.074	0.037	0.063
RP-RP max EER	0.175	0.148	0.111	0.111	0.111	0.113

Table 3.4: Min-Max EER Comparison for the PCA, PCARP and RP-RP feature extractor for various template sizes. The results for PCA can be seen as a "baseline" result for the dataset. We see that PCARP and RP-RP are less affected by the number of components than PCA. PCARP performs similarly to PCA, while RP-RP has more variance in its results due to having more randomization in it.

as a slightly worse performance than PCA in most cases, although there are some situations where PCARP ends up with a slightly better performance than PCA.

Furthermore, the randomness of PCARP and RP-RP seems to be less affected by the choice of the number of components than PCA. The RP-RP feature extractor appears to perform similarly or worse than the PCARP feature extractor, although with more variability due to the first RP extractor also adding randomness to the final results. RP-RP does appear to perform better when a higher number of components is used.

We have observed similar behavior for PCA, PCARP and RP-RP with other sets of enrolment images. Our observations match the results of [WP10]. That is, PCA generally offers better performance than the other two methods due to a lack of randomization. On the other hand, PCARP and RP-RP, although slightly worse than PCA, offers similar performances to one another, especially as the number of components increases. We also note that the added randomization can sometimes allow PCARP and RP-RP to perform no worse or better than PCA with regards to the EER.

Performance at Low FPR

Although the EER is one value that can be used to compare various schemes, it is not always the best metric, as some applications might prioritize minimizing the FPR or the FNR at the risk of having the other value increased. This is the case of the setting that motivated this research : the problem of correctly identifying low-risk travellers at the border to expedite their processing.

In the setting of the project, a false positive can have far bigger consequences (i.e. a criminal successfully entering the country by passing as another traveller) than a false negative (which only means that the traveller now has to be treated by a border agent through the normal process). In this particular case, a low FPR is more important than a low FNR.

As such, we compare the three feature extractors at a given FP count (and thus the same FPR) by using the positive predictive value (PPV) and the FNR. Some results when using PCA, PCARP and RP-RP can respectively be found in Table 3.5, 3.6 and 3.7.

FP (FPR)	TP	PPV	FNR
0 (0.000)	7	1.000	0.731
2 (0.003)	10	0.833	0.615
5 (0.008)	13	0.722	0.500
10 (0.015)	14	0.583	0.462
15 (0.023)	17	0.531	0.346

Table 3.5: TP amount, PPV and FNR for PCA at various fixed FP values, for 20 components. When no randomization is present, the fuzzy extractor can perform relatively well in a low FPR setting, with a high PPV and a somewhat high FNR. A small increase in tolerated FPR can drastically lower the PPV and FNR.

	Highest			Lowest		
FP (FPR)	TP	PPV	FNR	TP	PPV	FNR
0 (0.000)	0	N/A	1.000	10	1.000	0.615
2 (0.003)	7	0.778	0.731	14	0.875	0.462
5 (0.008)	8	0.615	0.692	18	0.783	0.308
10 (0.015)	15	0.600	0.423	19	0.655	0.269
15 (0.023)	16	0.516	0.385	19	0.559	0.269

Table 3.6: TP amount, PPV and FNR for PCARP at various fixed FP values, for 20 components. The table contains the two tests with the highest and lowest observed FNR, respectively. A lot of variation can be observed between a good and bad scenario, with the FPR being quite different to achieve the same levels of PPC or FNR.

	Test 1			Test 2			Test 3		
FP (FPR)	TP	PPV	FNR	TP	PPV	FNR	TP	PPV	FNR
0 (0.000)	0	N/A	1.000	1	1.000	0.962	0	N/A	1.000
2 (0.003)	1	0.333	0.962	6	0.750	0.769	4	0.667	0.846
5 (0.008)	1	0.167	0.962	6	0.545	0.769	8	0.615	0.692
10 (0.015)	7	0.412	0.731	6	0.375	0.769	10	0.500	0.615
15 (0.023)	7	0.318	0.731	12	0.444	0.538	14	0.483	0.462

Table 3.7: TP amount, PPV and FNR for RP-RP at various fixed FP values, for 20 components. This table contains the results of three different tests. As can be seen, the high randomness induced by an RP-RP extractor makes the scheme perform somewhat erratically at low FPR values.

These results were obtained using the same enrolment set of 6 images and 1 authentication image per user. The FPR values we look at have been arbitrarily chosen as some plausible benchmark values to compare the schemes; an "acceptable" value for the FPR is something that needs to be determined for each situation.

The numbers were obtained as follows. First we did the tests using a feature extractor to obtain a list of scores and truth values. Then, from that list, we can find the score at which a certain FPR is attained. Anything with a score less than that selected score is

considered as successfully authenticated by the scheme, anything above is a rejection. We can then obtain various statistics form these numbers.

As previously, the test using PCARP and RP-RP were both done multiple times to take into account the effect of the randomness. For PCARP, the outcomes were generally consistent across all tests regarding the changes in the FNR as the FPR increases. That is, when one test performed at least as well as another for a given FPR, it performed also performed at least as well at the other values of FPR. As such, we present the two tests with the highest and lowest observed FNR in Table 3.6. Like what we noticed with the EER, PCARP performs similarly as PCA at a low FPR threshold. The added randomness sometimes makes PCARP perform better, and some other times slightly worse than PCA.

On the other hand, RP-RP is more affected by the randomness and is more erratic in its performance at low FPR. It also performed worse than PCA and PCARP at these low FPR thresholds. This can be seen in Table 3.7, where we present the outcomes for tree different tests. Thus, the RP-RP extractor does not appear to be a suitable feature extractor when a low FPR is required.

3.5 Conclusion

In this chapter, we presented an improved version of Sutcu et al.'s fuzzy extractor scheme. This improved version is more secure and private than the original. It also performs better than the original, as the randomization of the original made it so that no users were authenticated when using the AR database. Finally, our improved scheme is also more easily implementable in a real world application, as it does not require any user to be enrolled as the system is set up.

We first tested the original scheme to see how it would perform on the database. Then, we then tested the improved scheme using a subset of the AR data set while using three different feature extractors : PCA, PCARP and RP-RP. These various test results have shown that using PCARP or RP-RP, as well as our other changes, provide a good performance for a fuzzy extractor. Using PCARP in particular can provide similar error rates than when using PCA, both regarding the EER and a small FPR threshold. It sometimes performed slightly better than PCA and sometimes slightly worse. The variation in performance compared to PCA is an expected result of adding randomization to provide both privacy and changeability to the scheme.

When using RP-RP, the EER is similar or slightly worse than PCA and PCARP, and the results are, as can be expected, more affected by the randomization. This extractor is also less suited for applications needing a low FPR threshold as it appears to generally perform worse than either PCA or PCARP in that situation.

Finally, we note that the set of enrolment images (its size and content) as well as the authentication image appears to be one of the main factors for the good performance of the scheme. In a similar vein, the feature extractor used also affects performance and can also help dictate the content and size of the enrolment images. For example, PCA is known to

be affected by lighting and glasses and using a different technique (or a commercial feature extractor) would deal with them differently.

Regardless of the feature extractor or enrolment set used, the other parameters are still important. Indeed, the choice of the final number of components still has an effect on the performance, albeit a lesser one. Similarly, the choice of the quantization steps δ_j can also have a small impact on the performance, while the value of β will need to be chosen specifically for each application. The optimal values for these parameters in any given situation would need to be determined through prior knowledge on the inputs, in an empirical fashion, or by using knowledge-based systems.

These observations raise various questions and avenues on how to optimize the performance of our fuzzy extractor. In particular, studying the amount and types of images the enrolment set needs, in relation to the feature extractor used, is a necessity to balance the performance and burden at enrolment for a user of the system. This can be particularly useful in a setting, like a border crossing scenario, where we can control to a certain extent the content of the provided images. Additionally, preprocessing of the images is also something that should be studied to improve the performance of the scheme, as we only did minimal preprocessing in this work.

Chapter 4

Decoding of Low Density Lattice Codes

In this chapter, we look at Low Density Lattice Codes (LDLC), how to construct them and their decoding algorithms. Low density lattice codes were first presented by Sommer, et al., in [SFS08], as practical construction for a capacity achieving lattice codes over the additive white Gaussian noise (AWGN) channel. These codes can be seen as the Euclidean space analog of the binary low density parity check (LDPC) codes proposed by [Gal63]. Similar to LDPC codes, LDLC can be efficiently decoded through the iterative belief propagation algorithm, which is linear in the block length of the code.

Following [SFS08], LDLC have generated a lot of research interest and various improvements were proposed to the decoding algorithm. These improvements are linked to the representation of Gaussian messages and Gaussian mixtures in the algorithm, as well as how many Gaussian mixtures should be kept throughout the process to still obtain good performance of the decoder [KD08, KD10, YF09, HA16, LHV$^+$18].

In the rest of this chapter, we will first present definitions related to LDLC, how to decode them and finally how to construct them.

4.1 Low Density Lattice Codes

Low Density Lattice Codes (LDLC) were first presented by Sommer et al. in [SFS08] as a practical and efficiently decoded lattice code. These LDLC were designed as a generalization to the Euclidean space of the low density parity check codes introduced by Gallager [Gal63].

Recall that an m-dimensional lattice in $\mathbb{R}^n$ is the set of all integer linear combinations of m linearly independent vectors in $\mathbb{R}^n$ and can be described by the generating matrix $\mathbf{G}$.

Definition 4.1.1. LDLC [SFS08] An *n-dimensional LDLC* is an n-dimensional lattice code with a nonsingular lattice generator matrix $\mathbf{G}$ satisfying $|\det(\mathbf{G})| = 1$, for which the parity check matrix $\mathbf{H} = \mathbf{G}^{-1}$ is sparse.

$$\begin{pmatrix} 0 & 0 & 0 & -0.5 & 0 & -0.25 & 0 & 1 \\ 0 & 0.5 & 0 & 0 & 0 & 0 & 1 & 0.25 \\ 0.5 & 0.25 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -0.25 & 0 & 0 & -0.5 & 0 & 0 \\ 0 & 0 & -1 & -0.25 & -0.5 & 0 & 0 & 0 \\ 0.25 & 0 & 0 & 0 & 1 & 0 & 0 & 0.5 \\ 1 & 0 & 0.5 & 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 1 & 0.5 & 0 \end{pmatrix}$$

Figure 4.1: Latin Square LDLC matrix $\mathbf{H}$ with generating sequence: $1, 0.5, 0.25$

The degree of a row r_i, $i = 1, \ldots, n$, is defined as the number of non-zero elements in row i of $\mathbf{H}$, and the degree of a column c_i, $i = 1, \ldots, n$, is defined as the number of non-zero elements in column i of $\mathbf{H}$.

A n-dimensional LDLC is *regular* if all the row degrees and column degrees of $\mathbf{H}$ are equal to a common degree d.

To construct an LDLC, it is advantageous to construct a sparse $\mathbf{H}$ and compute $\mathbf{G}$ afterward. The parity check matrix is chosen to be sparse to have a more efficient decoding algorithm. The drawback is that if $\mathbf{H}$ is sparse there is no guarantee that $\mathbf{G}$ will be sparse. LDLC thus have a slightly more expensive encoding in exchange for efficient decoding.

Definition 4.1.2. A regular n-dimensional LDLC of degree d is called a *Latin square LDLC* if every row and column of the parity check matrix $\mathbf{H}$ have the same d non-zero values, with the possibility of random signs.

The sorted sequences of d values $h_1 \geq h_2 \geq \cdots \geq h_d > 0$ for a Latin square LDLC is called the *generating sequence*.

We give an algorithm for constructing a sparse Latin square matrix that can be used in a LDLC in section 4.4. That algorithm constructs parity check matrices that allow the decoding algorithm for LDLC to both converge and be efficient. An example of a matrix generated by the algorithm can be found in Figure 4.1.

We also note that a sparse Latin square matrix can be stored by only keeping track of the non-zero elements in each row (or column) of the matrix. This is a similar approach to storing LDPC codes as given in [Moo05], but instead of only storing the position of the non-zero elements, we also need to track the value. This can be done by, for example, representing each non-zero element in a given row i as a triple $(j, k, \pm 1)$, encoding that in column j we have the value $\pm h_k$. For example, the first row of the matrix in Figure 4.1 could be encoded as the list $\{(4, 2, -1), (6, 3, -1), (8, 1, 1)\}$.

As mentioned in a previous chapter, it was shown in [Pol94] that for an unconstrained AWGN channel, the capacity of the channel can be defined as a measure of the density of codewords. When applied to n-dimensional lattice code generated by $\mathbf{G}$, this means that

error correction with small probability of error is possible only when

$$\sigma^2 < \frac{\sqrt[n]{\det(\mathbf{G})^2}}{2\pi e}.$$

By the definition of LDLC, we need that $|\det(\mathbf{G})| = |\det(\mathbf{H})| = 1$, which allows these codes to correct errors for a channel noise $\sigma^2 < \frac{1}{2\pi e} \approx 0.0585$. The construction for Latin square LDLC in the next section will be such that $|\det(\mathbf{H})| \approx 1$.

To obtain an LDLC for an arbitrary noise σ^2, we need to scale the matrices. Assume that $|\det(\mathbf{G})| = 1$ and that $\sigma^2 < \frac{\alpha}{2\pi e}$, then we can show that choosing a lattice with $\mathbf{G}' = \sqrt{\alpha}\mathbf{G}$ will give an LDLC that can correct errors with channel noise σ^2 :

$$\frac{\sqrt[n]{\det(\mathbf{G}'^2)}}{2\pi e} = \frac{\sqrt[n]{\det(\sqrt{\alpha}\mathbf{G})^2}}{2\pi e} = \alpha\frac{1}{2\pi e} > \sigma^2.$$

Solving for $\sqrt{\alpha}$ in the last part yields : $\sqrt{\alpha} > \sqrt{2\pi e\sigma^2}$. Since we will be building the codes from $\mathbf{H}$, we can thus scale the parity check matrix by $\frac{1}{\sqrt{\alpha}} = \frac{1}{\sqrt{2\pi e\sigma^2}}$ for a channel noise of σ^2. Practically, $\sqrt{\alpha}$ can be seen as a gain factor for the transmitted codewords.

4.2 Belief Propagation Algorithm

In this section, we present the belief propagation decoding algorithms for LDLCs. We will refer mostly to [SFS08] for the original decoding algorithm on LDLC, and [KD10, YF09, KD08, HA16] for its various improvements. Since these codes are a generalization of the binary Low Density Parity Check (LDPC) codes, we also refer the reader to [Moo05], and other works on error correcting codes, for a more in depth coverage of the message passing algorithms.

Consider a (binary or lattice) code with a parity check matrix $\mathbf{H}$. Associated to this matrix is a bipartite graph called the *Tanner graph* which has two sets of nodes: the check nodes and the variable (or bit) nodes. The check nodes correspond to the check equations, i.e. a row in $\mathbf{H}$, while the variable nodes correspond to elements of the codewords. Then, an edge connects the variable node x_i to the check node c_j if and only if $H_{i,j} \neq 0$. That edge is also given the value $H_{i,j}$. Finally, a k-loop will be defined as a loop of length k.

The belief propagation algorithm is a soft decoder that works by propagating probabilities through the graph. The general idea is the following. The variable nodes send probability density functions (pdf) as messages through the graphs to the check nodes. Each check node takes all these pdfs and combines them according to the decoder used. The check nodes then send (possibly different) messages back to each variable nodes, where each variable node combine all the messages they received to update their pdf. These steps are repeated for some number of iterations. The final decoding consists of estimating the transmitted message by selecting the most likely values from each of the variable node's pdf.

4.2.1 Lattice Decoding

In this section, we present the original lattice decoding algorithm given in [SFS08]. For the rest of this chapter, suppose $x = \mathbf{G}b$ is the transmitted codeword, b is the encoded message and $y = x + z$ is the received codeword where z is a vector of i.i.d. Gaussian with noise variance σ^2.

Let $\mathbf{H}$ be the square parity check matrix for a LDLC and denote the variable nodes by $x_1, \ldots, x_n$ and the check nodes by $c_1, \ldots, c_n$. We will denote the pdf for a normal, or Gaussian, distribution with mean m and variance v by $N(z; m, v) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(z-m)^2}{2v}}$.

The algorithm begins with all the variable nodes sending an initial message to each of their adjacent check nodes, with node x_k sending the message $f_k(x) = N(x; y_k, \sigma^2)$.

Then, the first part of one iteration consists of computing the messages from the check nodes. Each check node will send a message to each of its connected variable nodes. For a given check node c_i, let $x_{m_j}, j = 1, \ldots, d$ be the nodes connected to it and let f_j denote the message received from x_{m_j}. Here d is the degree of the i-th row of $\mathbf{H}$ and set $h_j = \mathbf{H}_{i,m_j}$.

For each check node c_i, the LDLC matrix $\mathbf{H}$ gives a check equation of the form $\sum_{k=1}^{d} h_k x_{m_k} = integer$. Solving for x_{m_j} we obtain that

$$x_{m_j} = \frac{1}{h_j} \left(integer - \sum_{k=1, k \neq j}^{d} h_k x_{m_k} \right).$$

Under the assumption that all the x_k are independents variables, we get that the pdf for x_{m_j} is a convolution of all the other stretched pdfs received by the check node that is then expanded and repeated over all possible integers. As such, the check node c_i computes the message $Q_j(x)$ that is sent back to the variable node x_{m_j} as follows:

1. All the incoming messages, except f_j, are convolved as follow:
$$\tilde{p}_j(x) = f_1(\frac{x}{h_1}) * \cdots * f_{j-1}(\frac{x}{h_{j-1}}) * f_{j+1}(\frac{x}{h_{j+1}}) * \cdots * f_d(\frac{x}{h_d})$$
2. The result is stretched by $-h_j$: $p_j(x) = \tilde{p}_j(-h_j x)$
3. This result is then extended to a periodic function with period $\frac{1}{|h_j|}$:
$$Q_j(x) = \sum_{k \in \mathbb{R}} p_j(x - \frac{k}{h_j}).$$
This extension is so that all the possible integer values are considered in decoding the message.

The second part of an iteration consists of computing the next messages sent by the variable nodes. Each variable node will send a message to each of its connected check nodes. As previously, for a given variable node x_j, denote by $c_{m_i}, i = 1, \ldots, d'$ the check nodes connected to x_j and let $Q_i(x)$ be the message received from them. d' is the degree of the j-th column of $\mathbf{H}$. The message sent back to node c_{m_j} is $f_j(x)$ and it is computed as follows.

1. All the incoming messages, except $Q_i(x)$, are multiplied as follow:

$$\tilde{f}_j(x) = e^{-\frac{(x-y_j)^2}{2\sigma^2}} \prod_{k=1,k\neq i}^{d'} Q_k(x)$$

2. The result is normalized into a pdf: $f_j(x) = \frac{\tilde{f}_j(x)}{\int_{-\infty}^{\infty} \tilde{f}_j(x)dx}$

This process of updating all the check nodes messages then updating all the variable nodes pdfs is one iteration of the decoder. This is repeated for a desired number of time. Then, when it is time to finalize the decoding and estimate the transmitted message $\mathbf{b}$, the variables nodes are updated in a slightly different manner. For a given variable node x_j, the process is as follows.

1. All the incoming messages $Q_i(x)$ are multiplied as follow:

$$\tilde{f}_j(x) = e^{-\frac{(x-y_j)^2}{2\sigma^2}} \prod_{k=1}^{d'} Q_k(x)$$

2. Find the peak of the pdf $\tilde{f}_j(x)$ and obtain the estimator $\hat{x}_j = arg\max_x \tilde{f}_j(x)$.

The message is then decoded by computing $\hat{\mathbf{b}} = \lfloor \mathbf{H}\hat{\mathbf{x}} \rceil$, where $\lfloor \cdot \rceil$ is rounding each component to the nearest integer.

A thorough analysis on the convergence of the decoder can be found in [SFS08]. The analysis revolves around the fact that all the messages that are passed between the nodes (both $Q_i(x)$ and $f_j(x)$) can be represented as Gaussian mixtures $M(x) = \sum_{i=1}^{\infty} a_i N(x; \mu_i, \sigma_i^2)$. The authors then prove that, under suitable constraints on $\mathbf{H}$, these mixtures will converge to a lattice point with some possibility of errors. These particular necessary conditions on $\mathbf{H}$ are all satisfied by the construction presented in section 4.4.

Regarding the implementation of such a decoding algorithm, we note that it might be impractical or inefficient to work directly with the exact pdfs. As such, the practical implementation proposed in [SFS08] approximates the pdfs by using a discrete set of points over a large enough range and with a resolution of Δ.

In particular, they suggest approximating the pdfs using a resolution of $\Delta = \frac{1}{64}$ over a range of 12σ centered around the observed value from the channel, which yields vectors of length $L = 256$. The most computationally expensive part of the algorithm ends up being the convolution step, which is done using fast Fourier transforms and inverse fast Fourier transforms.

Similar to Gallager's LDPC codes, LDLC achieve decoding complexity that is linear in the block size n. This particular decoder implementation over Latin square LDLC has a complexity of $O(n \cdot d \cdot t \cdot \frac{1}{\Delta} \log(\frac{1}{\Delta}))$, where t is the number of iterations in the decoder.

4.3 Single Gaussian Message Decoder

The original decoder for LDLC can be expensive in both computational and memory requirements due to the use of Fourier transforms and the fact that each message passed by the algorithm is a vector of L values.

Improvements to the decoder were then proposed in [KD08] and [YF09] by representing all the messages internally as Gaussian mixtures instead of vectors of values. A technique of Gaussian mixture reduction is used at the nodes to approximate large mixtures with smaller mixtures, with only minimal performance loss to the decoder. Finally, a single Gaussian decoder was proposed in [KD10] and [HA16], where all messages were single Gaussians and where internal representations of the pdfs were reduced to mixtures of 1, 2 or 3 Gaussian at the nodes.

In this section we will present the single Gaussian decoder of [KD10] as well as the algorithms used in the Gaussian mixture reductions.

4.3.1 Gaussian Mixture Reduction

Before we describe the reduction algorithm, we begin by presenting various results on Gaussians and Gaussian mixtures. A Gaussian with mean m and variance v is denoted as

$$N(z; m, v) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(z-m)^2}{2v}}.$$

A *Gaussian mixture* of size N is a function $f(z)$ such that

$$f(z) = \sum_{i}^{N} c_i N(z; m_i, v_i),$$

and where the coefficients are such that $c_i > 0$ for all i and $\sum_i c_i = 1$. A Gaussian mixture $f(z)$ can thus be represented as a list of triples $\{(m_1, v_1, c_1), \ldots, (m_N, v_N, c_N)\}$. Gaussian functions also have the nice properties that convolution and products of Gaussian are also Gaussian.

Proposition 4.3.1. *Consider n Gaussian with respective means m_i and variance v_i, for $i = 1, \ldots, n$. Then the convolution of these n Gaussian is a Gaussian with mean $m = \sum_i^n m_i$ and variance $v = \sum_i^n v_i$.*

Proposition 4.3.2. *The product of two (scaled) Gaussian $c_1 N(z; m_1, v_1)$ and $c_2 N(z; m_2, v_2)$ is the scaled Gaussian $cN(z; m, v)$ where*

$$\begin{aligned} v &= (\tfrac{1}{v_1} + \tfrac{1}{v_2})^{-1} = \tfrac{v_1 v_2}{v_1 + v_2} \\ m &= v(\tfrac{m_1}{v_1} + \tfrac{m_2}{v_2}) \\ c &= \frac{c_1 c_2}{\sqrt{2\pi(v_1+v_2)}} e^{-\frac{(m_1-m_2)^2}{2(v_1+v_2)}} \end{aligned}$$

The approach proposed in [KD08] consists of successively replacing pairs of Gaussian in a mixture by another Gaussian in such a way as to minimize the Kullback-Leibler (KL) divergence. The KL divergence, or relative entropy, between two distributions $p(x), q(x)$ is defined as $\mathrm{KL}(p||q) = \int_{\mathbb{R}} p(x)\log(\frac{p(x)}{q(x)})dx$. When replacing a distribution $p(x)$ by a Gaussian $q(x)$, the Gaussian that minimizes the KL divergence is the one that matches the first two moments of $p(x)$ (see [RW05] Appendix A).

More specifically, given a Gaussian mixture $p(z) = \sum_i^N c_i N(z; m_i, v_i)$, then $q(z) = N(z; m, v)$ where

$$\begin{aligned} m &= \textstyle\sum_i^N c_i m_i \\ v &= \textstyle\sum_i^N c_i(v_i + m_i^2) - m^2, \end{aligned}$$

is the single Gaussian minimizing the KL divergence with $p(z)$. This particular Gaussian found through moment matching will be denoted by MM. When using list of triples to represent the mixtures, we will write

$$t = (m, v, 1) = \mathrm{MM}(t_1, \ldots, t_N)$$

for the Gaussian $q(z)$ obtained from $p(x)$ by moment matching.

The squared difference will be used as a measure of penalty to replacing a mixture with a single Gaussian. More specifically, given a mixture $p(z)$ and $q(z)$ the Gaussian obtained by moment matching, the *Gaussian quadratic loss* $\mathrm{GQL}(p)$ is defined as

$$\mathrm{GQL}(p) = \int_{\mathbb{R}} (p(z) - q(z))^2 dz.$$

When considering the case where $p(z)$ is a mixture of two Gaussian, we obtain the following result.

Proposition 4.3.3. *Consider the mixture of two Gaussians t_1, t_2, where $t_i = (m_i, v_i, c_i)$ and $c_1 + c_2 = 1$, and let $t = (m, v, 1) = MM(t_1, t_2)$. The Gaussian quadratic loss is given by:*

$$\begin{aligned} GQL(t_1, t_2) = \; & \tfrac{1}{2\sqrt{\pi v}} + \tfrac{c_1^2}{2\sqrt{\pi v_1}} + \tfrac{c_2^2}{2\sqrt{\pi v_2}} \\ & + \tfrac{2c_1}{\sqrt{2\pi(v+v_1)}} e^{-\frac{(m-m_1)^2}{2(v+v_1)}} + \tfrac{2c_2}{\sqrt{2\pi(v+v_2)}} e^{-\frac{(m-m_2)^2}{2(v+v_2)}} \\ & + \tfrac{2c_1c_2}{\sqrt{2\pi(v_1+v_2)}} e^{-\frac{(m_1-m_2)^2}{2(v_1+v_2)}} \end{aligned}$$

When the mixture of t_1, t_2 is not normalized, that is when $c_1 + c_2 = c \neq 1$, The GQL is defined to be the same as if the mixture was first normalized. That is, if we set $t_i' = (m_i, v_i, \frac{c_i'}{c})$, $i = 1, 2$, then $\mathrm{GQL}(t_1, t_2) = \mathrm{GQL}(t_1', t_2')$.

Finally, in a situation where we are replacing *parts* of a Gaussian mixtures with another Gaussian, we may not necessarily have that $\sum c_i = 1$ for moment matching. In that situation, we can similarly normalize the coefficients before returning a scaled Gaussian through moment matching.

The algorithms

We describe the Gaussian mixture reduction (GMR) algorithm from [KD08] in Algorithm 1. The algorithm is a greedy algorithm that takes a mixture of N Gaussians and outputs a mixture with at most M_{Max} Gausssian. It is done by successively choosing the pair of Gaussians with the smallest GQL and replacing them by the moment matching Gaussian.

The GQL serves as a measure of how much error is incurred by the replacement. The algorithm continues to reduce the mixture as long as the lowest GQL is less than a specified threshold θ or the current size of the mixture is bigger than M_{Max}.

Algorithm 1 Gaussian Mixture Reduction Algorithm

Input: Mixture as a list of triples $L = \{t_1, \ldots, t_N\}$, target mixture size M_{Max}, threshold θ.

Output: List of triples of size $\leq M_{Max}$ that is the reduction of the mixture L.

Initialize mixture to reduce and its size:
$C \leftarrow L$
$M \leftarrow N$
Find pair t_i, t_j, $i \neq j$ in C such that $\mathrm{GQL}(t_i, t_j)$ is the minimum between all pairs:
$(t_i, t_j) \leftarrow arg\min_{t_i, t_j \in C, i \neq j} \mathrm{GQL}(t_i, t_j)$
$\theta^c \leftarrow \mathrm{GQL}(t_i, t_j)$
while $\theta^c \leq \theta$ or $M > M_{Max}$ **do**
    $t \leftarrow \mathrm{MM}(t_i, t_j)$
    Replacing (t_i, t_j) by t in C :
    $C \leftarrow C \cup \{t\}$
    $C \leftarrow C \ \{t_i, t_j\}$
    $M \leftarrow M - 1$
    Find next pair and update θ^c :
    $(t_i, t_j) \leftarrow arg\min_{t_i, t_j \in C, i \neq j} \mathrm{GQL}(t_i, t_j)$
    $\theta^c \leftarrow \mathrm{GQL}(t_i, t_j)$
end while
return C, the reduced mixture of size M.

We also describe the moment matching algorithm on a pair of Gaussian that is used in the GMR algorithm in Algorithm 2.

4.3.2 The Decoder

The Gaussian mixture decoder works in a similar fashion as the decoder presented in section 4.2.1 with two main differences. The first is that all the messages passed between the nodes are pairs (m, v) representing a Gaussian pdf. The second is that the periodic extension from the check node is instead done at the variable node to allow the variable nodes to send Gaussian messages and not a periodic function.

As in the previous section, let $\mathbf{H}$ be the parity check for a LDLC and let $y = (y_1, \ldots, y_n)$ be the received codeword over a AWGN channel with noise variance σ^2. The variable

Algorithm 2 Moment Matching Algorithm

Input: Two (scaled) Gaussian $t_1 = (m_1, v_1, c_1)$ and $t_2 = (m_2, v_2, c_2)$.

Output: The (scaled) Gaussian that minimizes the KL divergence $t = (m, v, c) = \text{MM}(t_1, t_2)$.

1: Initializing and normalizing the coefficients:
2: $c \leftarrow c_1 + c_2$
3: $c_1' \leftarrow c_1/c$. $c_2' \leftarrow c_2/c$
4: Compute m and v:
5: $m \leftarrow c_1' m_1 + c_2' m_2$
6: $v \leftarrow c_1'(v_1 + m_1^2) + c_2'(v_2 + m_2^2) - m^2$
7: **return** The triple $t = (m, v, c)$.

nodes will be denoted by $x_1, \ldots, x_n$ and the check nodes by $c_1, \ldots, c_n$. Finally, we set $y_k(x) = N(x; y_k, \sigma^2)$.

The decoder is first initialized by having each variable nodes x_j send the message $y_j(x)$ to all their connected check nodes.

At the *check node update step*, each check node takes all its incoming messages and sends back a (possibly different) message to each of its connected variable node. For a given check node c_i, let $x_{i_j}, j = 1, \ldots, d$ be the nodes connected to it. For each each of these node x_{i_j}, let (m_j, v_j) describe the Gaussian received from it and h_j be the associated non-zero coefficient from $\mathbf{H}$.

From the check equations and because the convolution of Gaussian functions is also Gaussian, we obtain that the check node c_i sends the message $\tilde{p}_j = N(x; \tilde{m}_j, \tilde{v}_j)$ to the variable node x_{i_j} where

$$\tilde{m}_j = -\tfrac{1}{h_j} \textstyle\sum_{k=1, k\neq j}^{d} h_k m_k$$
$$\tilde{v}_j = \tfrac{1}{h_j^2} \textstyle\sum_{k=1, k\neq j}^{d} h_k^2 v_k.$$

For an efficient implementation of the check node update, forward-backward recursions can be used to compute all the messages output by a given check node. We present the check node update algorithm for a single check node in Algorithm 3.

We now look at the *variable node update step*. For any given variable node x_j, we denote by $c_{j_i}, i = 1, \ldots, d'$ the check nodes connected to x_j and let $\tilde{p}_i$ be the Gaussian received from that particular node. At the variable node x_j, the incoming messages first go through the periodic extension step that was previously skipped from the check node to obtain

$$p_i(x) = \sum_{l \in \mathbb{Z}} \tilde{p}_i(x + \frac{l}{h_j}) = \sum_{l \in \mathbb{Z}} N(x; \tilde{m}_j + \frac{l}{h_j}, \tilde{v}_j).$$

Then, for each i, the messages are multiplied together to obtain

$$\mu_i(x) = y_j(x) \prod_{k=1, k\neq i}^{d'} p_k(x).$$

Algorithm 3 Check Node Update

Input: Check node c_i, incoming messages (m_j, v_j) from x_{i_j} and the associated value h_j, for $j = 1, \ldots, d$

Output: Sends the message $(\tilde{m}_j, \tilde{v}_j)$ to node x_{i_j}, for all j.

Forward Recursion:
$A_0 \leftarrow 0$, $B_0 \leftarrow 0$
for $k = 1, \ldots, d$ **do**
    $A_k \leftarrow h_k m_k + A_{k-1}$
    $B_k \leftarrow h_k^2 v_k + B_{k-1}$
end for
Backward Recursion:
$a_{d+1} \leftarrow 0$, $b_{d+1} \leftarrow 0$
for $k = d, \ldots, 1$ **do**
    $a_k \leftarrow h_k m_k + a_{k+1}$
    $b_k \leftarrow h_k^2 v_k + b_{k+1}$
end for
for $j = 1, \ldots, d$ **do**
    $\tilde{m}_j \leftarrow -\frac{1}{h_j}(A_{j-1} + aj + 1)$
    $\tilde{v}_j \leftarrow \frac{1}{h_j^2}(B_{j-1} + bj + 1)$
    Send message $(\tilde{m}_j, \tilde{v}_j)$ to node x_{i_j}
end for

The node x_j then sends the moment matching Gaussian $f_i(x) = \mathrm{MM}(\mu_i(x))$ to the check node c_{j_i}.

As can be seen, the construction of $\mu_i(x)$ involves multiplying by the channel message Gaussian $y_i(x)$. This implies that the periodic Gaussians that are far from the channel message y_i will have negligible impact on the final value of $\mu_i(x)$. We can thus consider only the few section of $p_i(x)$ that are close to y_i. This allows us to approximate $p_i(x)$ with a small sum of Gaussian instead of an infinite sum.

In [HA16], the authors looked at the performance of the decoder when using a various number of Gaussians and arrived at the conclusion that representing $p_i(x)$ using only 2 or 3 Gaussians offers a good balance of decoder performance and computational efficiency. Accuracy for the decoder will increase as more Gaussians are kept, but the complexity grows accordingly.

In the case where 3 Gaussians are used for the periodic extension of $\tilde{p}_i(x)$, we set $B = \{b-1, b, b+1\}$ and we get $p_i(x) = \sum_{l \in B} N(x; \tilde{m}_j + \frac{l}{h_j}, \tilde{v}_j)$. The value b is the integer that minimizes the distance $|\tilde{m}_j + \frac{l}{h_j} - y_i|$; that is, $b = \lfloor h_i(\tilde{m}_j - y_i) \rceil$. In the case where we wish to use 2 Gaussians, we instead set $B = \{b, b+1\}$ with $b = \lfloor h_i(\tilde{m}_j - y_i) \rfloor$.

Like for the check node, the computations for the variable nodes can be done with a forward-backward recursion and by using the GMR algorithm after every multiplication. We present the variable node update algorithm in Algorithm 4. Multiplication of Gaussian is done as in proposition 4.3.2.

Both the forward and backward recursion are initialized with the pdf $\sqrt{y_i(x)}$, which is a Gaussian with mean y_i and variance $2\sigma^2$. The complexity of this algorithm is bounded by the size of the Gaussian mixtures, with each multiplication involving at most $|B| \times M_{Max}$ multiplication of Gaussian, where $|B|$ is the number of Gaussian chosen in the periodic extension and M_{Max} is the maximal number of Gaussian allowed in the reduced mixture for the GMR algorithm. It also follows that the biggest mixtures to reduce through GMR will also have at most $|B| \times M_{Max}$ Gaussian.

Finally, to prevent any instabilities in the decoding algorithm, a minimal variance γ should be selected for the outgoing messages $\mathrm{f}_i(x)$. This is done by replacing de variance v_i of $f_i(x)$ by $v = \max(v_i, \gamma)$. In [KD10], they chose γ to be in the range of 10^{-4} to 10^{-3}.

Algorithm 4 Variable Node Update

Input: Variable node x_j, incoming messages $\tilde{p}_i(x)$ from x_{j_i}, for $i = 1, \ldots, d'$
Output: Sends the Gaussian message $\mu_i(x)$ to node x_{j_i}, for all i.

1: **for** each incoming message $\tilde{p}_i(x)$ **do**
2: Compute the (limited) periodic extension $p_i(x)$ of $\tilde{p}_i(x)$ around the channel value y_j
3: **end for**
4: Forward Recursion:
5: $\alpha_0(x) \leftarrow \sqrt{y_j(x)} = N(x; y_i, 2\sigma^2)$
6: **for** $k = 1, \ldots, d'$ **do**
7: $\alpha_k(x) \leftarrow \mathrm{GMR}(\alpha_{k-1}(x) \cdot p_k(x))$
8: **end for**
9: Backward Recursion:
10: $\beta_{d'}(x) \leftarrow \sqrt{y_j(x)} = N(x; y_i, 2\sigma^2)$
11: **for** $k = d' - 1, \ldots, 1$ **do**
12: $\beta_k(x) \leftarrow \mathrm{GMR}(\beta_{k+1}(x) \cdot p_k(x))$
13: **end for**
14: Sending Messages:
15: **for** each connected check node x_{j_i} **do**
16: $\mathrm{f}_i(x) \leftarrow \mathrm{MM}(\alpha_{i-1}(x) \cdot \beta_i(x))$
17: The message $\mathrm{f}_i(x)$ is sent to node x_{j_i}
18: **end for**

At the final iteration of the decoder, the variable node x_j will instead output a single pdf $f_j = \mathrm{MM}(y_j(x) \prod_{k=1}^{d'} p_k(x))$. We then find the peak of the pdf to obtain the estimator $\hat{x}_j = arg\max_x f_j(x)$. Since $f_j(x)$ is Gaussian, it is maximized at its mean and we get $\hat{x}_j = m_j$

The decoding then terminates like the original scheme by computing $\hat{\mathbf{b}} = \lfloor \mathbf{H}\hat{\mathbf{x}} \rceil$, where $\lfloor \cdot \rceil$ is rounding each component to the nearest integer.

When applied to a Latin square LDLC, this decoder has a complexity of $O(n \cdot d \cdot t \cdot |B|^2 \cdot M^4)$, where t is the number of iteration in the decoder, $|B|$ is the number of Gaussians we use in the periodic extension and M is the maximal number of Gaussians allowed in the GMR algorithm. The memory requirement is $O(n \cdot d \cdot M)$ and all the messages are Gaussian represented as a pair of values.

4.4 Construction of Latin Square LDLC

We now present how to construct the parity check matrix for a Latin square LDLC so that the decoder can converge. The specific criterion for the decoder's convergence can be found in [SFS08]. The authors show that, in the case where $\mathbf{H}$ is a Latin square with generating sequence $h_1 > \cdots > h_d > 0$, all the necessary criterion are satisfied when:

1. $\det(\mathbf{H}) = 1$
2. $\gamma = \frac{\sum_{i=2}^{d} h_i^2}{h_1^2} < 1.$

If $\det(\mathbf{H}) = 1$, then the matrix $\mathbf{H}$ can be normalized by dividing it by $\sqrt[n]{\det(\mathbf{H})}$ for the code to fit in the LDLC definition. In practice, the normalization can be ignored if $\sqrt[n]{\det(\mathbf{H})} \approx 1$, as the effect on the decoder are negligible.

When choosing a generating sequence with $h_1 = 1$, $\gamma < 1$, a small d and a large n, the Latin square matrix $\mathbf{H}$ with randomized signs for its non-zero elements will be such that $\sqrt[n]{\det(\mathbf{H})} \approx 1$.

The authors of [SFS08] propose two generating sequences that give good results when using the decoder. The first one is based on taking the first d reciprocals of primes and adding some dither to them as it ended up improving performance over the pure reciprocals. This approach revolves around maximizing the least common multiple between the various periods at the periodic extension step of the decoder. This allows for most of the periodic extensions to be attenuated and only the main peak is kept. For $d = 7$ this yields the sequence $\{\frac{1}{2.31}, \frac{1}{3.17}, \frac{1}{5.11}, \frac{1}{7.33}, \frac{1}{11.71}, \frac{1}{13.11}, \frac{1}{17.55}\}$ and for $d < 7$ the first d elements are chosen.

The second generating sequence proposed is the sequence $\{1, \frac{1}{\sqrt{d}}, \ldots, \frac{1}{\sqrt{d}}\}$ where there are exactly $d - 1$ copies of $\frac{1}{\sqrt{d}}$. Both sequences offer good performance for the decoder for $3 \leq d \leq 7$, while improvements to the decoder were negligible when d was chosen to be bigger than 7.

A final criteria that allows the decoder to converge is that $\mathbf{H}$ should not have short cycles of length 2 or 4 in the Tanner graph. This is a required assumption for a belief propagation algorithm convergence. As such, the algorithm we will now present allows for the construction of Latin squares with no cycles of length 2 or 4.

The algorithm works as follow. It generates d permutations of length n, where permutation p_i encodes the position of h_i in every row. Then, the algorithm cycles through all the permutations to find any 2-cycles and 4-cycles. If a cycle is found, two values are swapped in one permutation to remove the loop. The search continues until no more loops are found.

Once no more loops are found, the Latin square can be built from the d permutations, the generating sequence and by randomly changing the signs of the non-zero values. For storage purposes, a given Latin square can be stored as the list of d permutations and a list of d vectors of ± 1 to encode the sign of each element.

Algorithm 5 Regular Latin square Matrix Generation

Input: Matrix size n, weight d and generating sequence $h_1 \geq \ldots \geq h_d > 0$
Output: Matrix $\mathbf{H}$ for a Latin square LDLC.

Generate d permutations over $\{1, \ldots, n\}$ arranged in a $d \times n$ matrix P.
Set index for loop and number of consecutive colums without loops:
$c \leftarrow 1$, $LooplessColumn \leftarrow 0$
Removing loops:
while $LooplessColumnss < n$ **do**
  $ChangedPermutation \leftarrow 0$
  if $P_{i,c} = P_{j,c}$ for some $i \neq j$ **then**
    A 2-loop was found at column c
    $ChangedPermutation \leftarrow c$
  else
    Look for 4-loops:
    for $c_0 = 1, \ldots, n$, $c_0 \neq c$ **do**
      if the columns $P_{:,c}$ and $P_{:,c_0}$ have ≥ 2 common elements **then**
        A 4-loop is found.
        Set i as the smallest row index where a common element appears in column c
        $ChangedPermutation \leftarrow i$
        Exit the loop.
      end if
    end for
  end if
  if $ChangedPermutation \neq 0$ **then**
    A loop was found and is to be removed:
    Randomly select an index j, $1 \leq j \leq n, j \neq c$
    $i \leftarrow ChangedPermutation$
    Swap the values $P_{i,c}$ and $P_{i,j}$
    $LooplessColumn \leftarrow 0$
  else
    No loop found at column c
    $looplessColumn \leftarrow looplessColumn + 1$
  end if
  Increment search index: $c \leftarrow c + 1$
  If $c > n$, $c \leftarrow 1$.
end while
Generating the Latin square matrix $\mathbf{H}$:
$\mathbf{H} \leftarrow \mathbf{0}_n$, the $n \times n$ zero matrix
for $j = 1, \ldots, n$ **do**
  for $i = 1, \ldots, d$ **do**
    $\mathbf{H}_{P_{i,j},j} = h_i \cdot RandomSign$
  end for
end for
return $\mathbf{H}$

Chapter 5

Fuzzy Extractor with LDLC

In this chapter, we propose a fuzzy extractor for faces using LDLC as the quantizer. The scheme is based on our improved version of Sutcu et al.'s scheme, but we replace the per-component quantizer with a LDLC quantizer. The idea is to use lattice codes to work directly in the Euclidean space while doing the quantization step.

LDLC were chosen as the quantizer for two reasons. First, LDLC are capacity achieving codes over $\mathbb{R}^n$ that can be efficiently decoded. They are also a practical construction of a lattice code. The second reason is that a fuzzy extractor using LDLC was proposed in [PvdG16] as an instantiation of the continuous-source fuzzy extractor defined in [BDHV07b]. This made LDLC a good candidate for a quantizer in a fuzzy extractor.

The use-case of the fuzzy extractor explored by the authors of [PvdG16] was information consolidation between two devices that receive the same input to their sensors. In one of their example scenarios, one could take two devices, like cellphones, and have them close together before making an audio recording simultaneously with both devices, or one could be holding the two cellphones in one hand and shaking them for a few seconds while recording the motions. The two devices can then use a LDLC fuzzy extractor and a wireless communication channel to extract and reconcile a secret key from the given input. Since the input recorded by the two devices will be very similar, the protocol should yield the same key for both devices.

Our approach uses the LDLC fuzzy extractor in a very different application : a biometric authentication setting. We build a practical fuzzy extractor for faces that is used in a standard key generation setting by combining elements from [SLM09] and [PvdG16]. In this setting, a user is enrolled and generates a key from some biometric inputs. The user can then authenticate themselves by using a new biometric measurement at a later date to regenerate their key. Fuzzy extractors using LDLC have yet, to our knowledge, to be tried or tested in such a way for authentication purposes using facial biometrics.

The goal of this authentication system is to increase the privacy of biometric schemes by changing the problem of securely storing biometrics to the better known and studied problem of storing cryptographic keys. Further more, the scheme is built in such a way that leaks will not compromise the biometrics, while also providing changeability and reusability to the biometrics.

A proposed application of this scheme is to generate a secure key using biometrics and using it with a symmetric cipher (like AES) to encrypt a token message or sign some electronic documents by using a MAC. If the user can regenerate the key to decrypt the token or verify the MAC with their biometrics, then we can authenticate the user as being who they claim to be. This system only stores seeds and random values instead of storing biometrics like a classical biometric authentication system.

5.1 Background

As presented in Chapter 2, a fuzzy extractor can be seen as a pair of procedures that generates and regenerates a (cryptographic) key from noisy data. They were originally defined over discrete space by Dodis et al. in [DORS08] and then extended to continuous inputs in [BDHV07b].

This extension of fuzzy extractors to the continuous domain is done by discretizing the global distribution of inputs, χ_g, to obtain a discrete distribution D_g from which the min-entropy $H_\infty(D_g) = -\log(\max_x(Pr[D_g = x]))$ can be computed. The min-entropy can be understood as the number of nearly uniform bits that can be extracted by the fuzzy extractor.

In [PvdG16], a particular instantiation of continuous source fuzzy extractor is presented using lattice codes and the code offset method. They also show that using LDLC as a quantizer still make for a secure fuzzy extractor. Their fuzzy extractor instantiation can be defined as follows.

Definition 5.1.1. [Lattice Fuzzy extractor from [PvdG16]] Let χ_g be the global distribution of the features, χ_a be the distribution of features of a user and $\mathcal{L}$ be a lattice with generator matrix $\mathbf{G}$. Furthermore, let D_g be the distribution χ_g quantized by $\mathcal{L}$ with $H_\infty(D_g) = m$. The procedures **Gen** and **Rep** are defined as follows:

1. **Gen** takes χ_a as input and outputs a pair of strings (k, h), where $k \in \{0,1\}^l$ is the private extracted string and $h \in \mathbb{R}^n$ is a public vector.
   The string k is the string representation of a random point $m \in \mathbb{Z}^n$, and set $p = \mathbf{G}m$ to be the associated point of the lattice.
   The public value h is obtained from $h = p + x_a$, with x_a being a feature sampled from χ_a
   For the fuzzy extractor to be secure, it must hold that, given knowledge of h, k should be indistinguishable from a uniformly sampled string k'.
2. **Reg** takes a measurement y sampled from χ_a and $h \in \mathbb{R}^n$ as inputs and outputs the correct string k with probability equal to the detection probability $p_d = 1 - e_{FRR}$. This is done by decoding $h - y$ to its nearest lattice point in $\mathcal{L}$, represented by $p' \in \mathbb{Z}^n$. k is recovered as the string representing p'.

This fuzzy extractor is shown to be secure and the strength of the key depends, as is expected, on the global distribution of inputs and the density of the lattice used. There is

thus a compromise between the noise correction allowed by the lattice and the security of the key.

We recall that an n-dimensional LDLC is a lattice code that has a sparse parity check matrix $\mathbf{H}$ of size n. The generator matrix for the lattice used in the code is $\mathbf{G} = \mathbf{H}^{-1}$. In LDLC, the messages are integer vectors $m \in \mathbb{Z}^n$ and the codewords are the lattice points $c = \mathbf{G}m$.

The decoding of an LDLC is done through an iterative belief propagation algorithm, and we will denote by $decode(x)$ the message obtained by decoding the vector $x \in \mathbb{R}^n$. The details on the construction and the decoding algorithm for an LDLC can be found in Chapter 4.

Since we will be using LDLC as quantizer of the fuzzy extractor, we will denote the quantizer associated with an LDLC by Q. This quantizer gives the closest lattice point to a given input and it will be defined as $Q(x) = \mathbf{G} \cdot decode(x)$ for $x \in \mathbb{R}^n$.

5.2 The Fuzzy Extractor

The scheme we propose will follow the approach of Sutcu et al. of giving each user their own quantizer based around the distribution of their own biometric. This can be seen as each user building their own fuzzy extractor where the quantization is done through a LDLC with parity check matrix $\mathbf{H}$.

The general construction of our fuzzy extractor, for a given user, is as follow. Suppose that the features $x = (x_1, \ldots, x_n)$ of a user follow a distribution such that $x_i \sim N(\mu_i, \sigma^2)$ for some unique variance σ^2. We can denote the mean feature as $\mu = (\mu_1, \ldots, \mu_n)$. We can then say that this user's features are of the form $\mu + z$, with z being a vector of white Gaussian noise with variance σ^2. That is, $z_i \sim N(0, \sigma^2)$.

This user can then build a parity check matrix $\mathbf{H}$ for a LDLC code over a channel with noise σ^2 for the fuzzy extractor quantizer. If μ is a lattice point and $y = \mu + z$ is a biometric from the user, then $Q(y)$ should be μ with low error probability and the secret key k can be generated from μ (for example by hashing μ).

It is unlikely that μ will be a lattice point, as lattices are always centered at the origin. As such, we can set $h = Q(\mu) - \mu = \mathbf{G} \cdot decode(\mu) - \mu$ as helper data. The secret key k is still generated from μ, for example by taking a cryptographic hash of μ. The helper data h will work as a translation vector to align μ, and thus the user's distribution, with the lattice points.

To regenerate k from a new feature y and the public data h, we compute $x' = \mathbf{G} \cdot Q(y - h) + h$. If $y = \mu + z$, then, with small error probability, x' will be equal to μ and the secret key k can be rebuilt. If y is not sampled from the user's distribution, then there are no guarantee that the secret can be correctly rebuilt.

In an application, it is generally impossible to know the true distributions of features for any given user. Furthermore, the variance might not behave like white noise over each

component. As such, we will approximate the distribution from a sample of features. A user will provide N features $x_1, \ldots, x_N$ at the enrolment phase and we compute the sample mean $\bar{x} = \frac{1}{N}\sum x_j$ and the sample variance v_i for each component $i = 1, \ldots, n$. We use $\bar{x}$ as the approximation of μ. We then choose a noise variance for the lattice code, for example we can choose $\sigma^2 = \max v_i$. We can also further scale the noise variance σ^2 by a factor of $\beta > 0$ to better control the strictness of the scheme : a higher noise tolerance (i.e. with $\beta > 1$) will increase both the true positive rate (TPR) and FPR, while smaller noise tolerance (i.e. with $\beta < 1$) will reduce those TPR and FPR.

To provide additional security in the case of a leaked codebook, more randomness to the generated key, and more reusability for the biometrics, we add a random codeword shift to the data before computing the key k. Instead of building k directly from $\bar{x}$ or μ, we first sample a random lattice point p. This can be done by uniformly sampling a message ρ from $\{0, \pm 1, \ldots, \pm\lambda\}$ for some integer λ. We then use m to find the corresponding lattice point $p = \mathbf{G}\rho$. Then, instead of using μ as the secret we can compute the secret point as

$$s = p + \mu = p + Q(\mu) + h = \mathbf{G}m + \mathbf{G} \cdot decode(\mu) - h.$$

This construction of s provides the advantage that it prevents brute force searches to find s if the lattice $\mathbf{G}$ is leaked. Another option is to keep h as a private value and instead set

$$s = p + Q(\mu) + r,$$

with $r \in \mathbb{R}^n$ being a random vector that will be the public helper data.

The addition of the random codeword shift has the additional advantage of increasing the final entropy of the key, provided the shift is kept secret. In particular, this can prevent using general knowledge about the information contained in a biometric to guess or recover the key from a token encrypted with it. This can allow for the generation of cryptographically secure keys from inputs or biometrics that may not naturally have a high amount of information in them.

5.3 Description of the Scheme

We present the scheme using PCARP, but PCA can easily be substituted by any other feature extractor. The RP part of PCARP is there for randomizing the features and to provide both changeability and privacy in the case that the randomized features are obtained by an attacker. PCARP could, for example, be replaced with a different RP extractor for each user. Although the scaling parameter β is given as a global parameter, it can easily be made to be a user-specific parameter.

As we will see in a later section, the RP extractor appears to be a necessity for the scheme to work well in a facial image setting when using a PCA feature extractor. This is due to the fact that RP extractors tend to spread the variance of the data over all components, while an extractor like PCA concentrates the data variance in the first few components of the features.

Figures 5.3.2 and 5.3.3 contains diagrams for the enrolment and authentication process using this LDLC scheme.

5.3.1 The setup phase

The setup phase consists of training the PCA extractor and setting the global parameters of the system. Let $n_2 \geq 2m$ be the target dimension for PCA and let $T = \{b_1, \ldots, b_l\} \subset \mathbb{R}^n$ with $l > n_2$ be a set of training biometrics for PCA.

1. Train a PCA feature extractor $f_{PCA} \leftarrow \text{TrainPCA}(T, n_2)$.
2. Select threshold parameter t for the RP extractor.
3. Choose values for scaling parameter β for the system.
4. Set N to be the number of images required to enroll a user.
5. Set λ as the security parameter for the codebook shift.
6. Choose a generating sequence for a Latin square LDLC $\{h_1, \ldots, h_d\}$.

Store f_{PCA}, β, N and the LDLC generating sequence as server parameters.

We note that generating many Latin square LDLC can be computationally expensive. It can be practical to pre-generate many Latin square and store their representations for later uses. Generating a user LDLC can then be done by simply selecting a pre-generated Latin square at random.

5.3.2 The enrollment phase

In the enrollment phase, a user U_i is registered into the system. It requires a set of enrollment images $E_i = \{b_1, \ldots, b_N\} \subset \mathbb{R}^n$ and three seeds, or keys, k_1, k_2, k_3 to be used in randomization.

1. Use the keys k_1, k_2 to generate a user specific RP extractor $f_{RP,i} \leftarrow \text{RPGen}(n_2, m, k_1, k_2, t)$.
2. Obtain the user-specific PCARP extractor $f_i(x) = f_{RP,i}(f_{PCA}(x))$.
3. Extract the features of the biometrics and obtain $\{x_1, \ldots, x_N\}$, with $x_j = f_i(b_j)$.
4. From $\{x_1, \ldots, x_N\}$, compute $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_m) = \frac{1}{N}\sum_i x_i$, the mean value of each component.
5. Compute the sample variance for each component : $v_j^2 = \frac{1}{N-1}\sum_i (x_{i,j} - \overline{x}_j)^2$ for $j = 1, \ldots, m$. Set $\sigma^2 = \max_i(v_j^2)$.
6. Compute the scaling factor $\alpha = \frac{1}{\sqrt{2\pi e \beta \sigma^2}}$ for $\mathbf{H}$.
7. Generate the user's LDLC matrix $\mathbf{H}$ with the generating sequence $\{\alpha h_1, \ldots, \alpha h_d\}$. The user quantizer Q^i is the decoding of an input under the LDLC with the parity check matrix $\mathbf{H}$.
8. Decode $\overline{x}$ to obtain $q = Q^i(\overline{x}) = \mathbf{G}\dot{decode}(\overline{x})$, the nearest lattice point to $\overline{x}$.

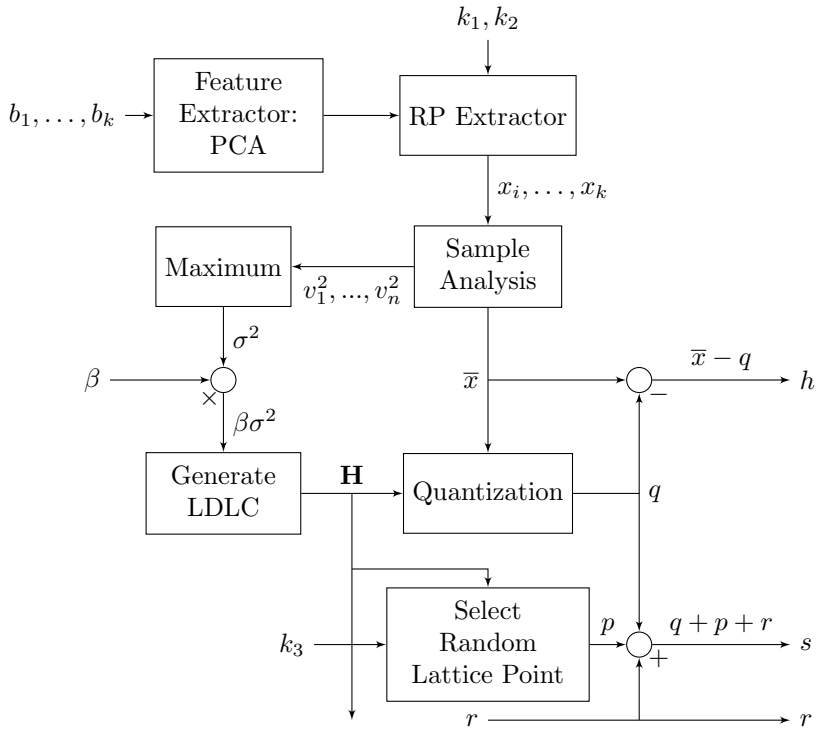

Figure 5.1: Enrolment process for a user U_i in a LDLC fuzzy extractor scheme. The β parameter can be system or user specific. The user LDLC parity check matrix $\mathbf{H}$ and the keys (or seeds) k_1, k_2, k_3 used in the randomization procedures are stored securely on the system. The public helper data r is a random vector given to the user, while $s = q + p + r$ is the secret. The translation data h can be kept secret or made public. The biometrics $b_1, \ldots, b_k$ are discarded after the enrolment is complete.

9. Compute the helper data $h = \overline{x} - l$, randomly sample a vector $r \leftarrow \mathbb{R}^m$, and use the key k_3 to choose a random lattice point p.
10. Compute the secret $s = p + q + r$.

The user's LDLC matrix $\mathbf{H}$ and the keys k_1, k_2, k_3 are securely stored in the system. The vector r is given to the user, and s can be used to generate an encryption key and encrypt a predetermined message M as $c_i = \mathrm{Enc}_s(M)$; this ciphertext is stored for future authentication of the user. The helper data h can be either stored securely or given to the user.

If the scheme is used as a secure sketch scheme where $\overline{x}$ is the secret that needs to be reconstructed, use $p = 0$ and $r = -h$ such that $s = q - h = \overline{x}$. It is also possible to use $r = h$ in the scheme.

5.3.3 Authentication phase

In the authentication phase, a user U' claims to be a user U_i by providing a biometric b' and vector r. The system accesses the stored data for user U_i : the three keys k_1, k_2, k_3,

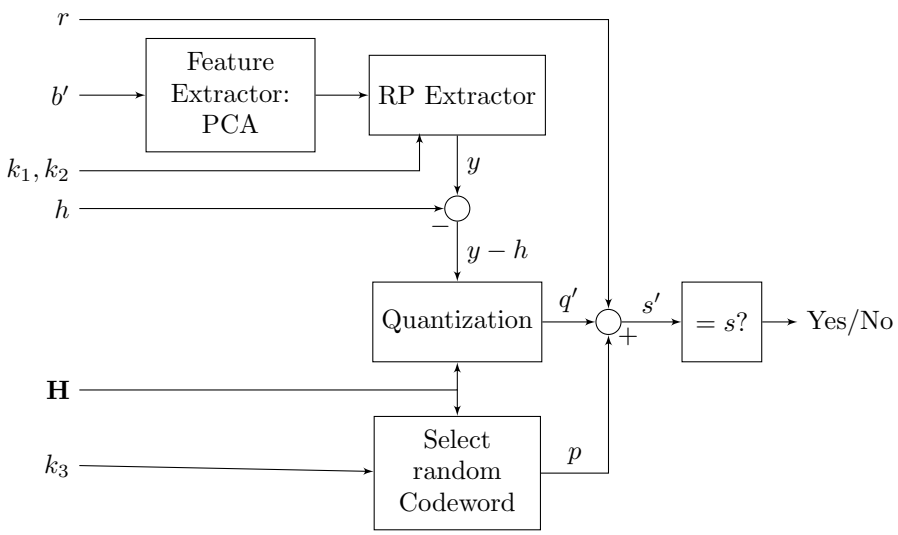

Figure 5.2: Authentication process for a user U_i in a LDLC fuzzy extractor scheme. The new biometric b' and the helper data r are provided by the user trying to authenticate. The randomization keys (or seeds) k_1, k_2, k_3 and the user's LDLC matrix $\mathbf{H}$ are taken from the system's secure storage. The translation data h will either come from the user if it was given, or from the secure storage.

the user LDLC matrix $\mathbf{H}$, the helper data h and the ciphertext c_i.

1. Use the keys k_1, k_2 to generate the user specific RP extractor $f_{RP,i} \leftarrow \text{RPGen}(n_2, m, k_1, k_2, t)$.
2. Obtain the PCARP extractor $f_i(x) = f_{RP,i}(f_{PCA}(x))$.
3. Extract the features of the biometric and obtain $x' = f_i(b')$.
4. Use the key k_3 to recover the lattice point p.
5. Compute $s' = Q^i(x' - h) + p + r$.
6. Verify if $\text{Dec}_{s'}(c_i) = M$. If the equality holds, authenticate U' as user U_i. Otherwise, reject the authentication.

5.4 Privacy Analysis

We now look at the privacy and security of the scheme when faced with an attacker following the attacker model from Section 2.2.1. That is, we consider a semi-passive attacker that

1. has full knowledge of the scheme,
2. has access to all the public data,
3. can get access through leaks, partially or totally, to the data that is stored or kept secret,
4. *cannot* influence or modify the scheme or any messages between entities, and
5. has *not* compromised any part of the system like the sensors.

An attacker is successful if they can either recover a user's biometric or if they can authenticate as the user. The scheme is considered more secure and private if the best angles of attack are general attacks that can also work on any other biometric authentication systems.

Suppose a user is enrolled in the system and the secret s is used to generate a key through a cryptographic hash function. That key is then used to generate an encrypted token T using AES, for example. We will assume that both T and the random vector r are public (or unsecured) and that the encryption scheme and the cryptographic hash function used are known. The privately stored data on the system are:

1. the translation vector h,
2. the lattice code, with matrices $\mathbf{G}$ and $\mathbf{H}$,
3. the codeword shift vector p obtained from the seed k_3
4. the RP extractor for the user, obtained from the seeds k_1, k_2
5. the global feature extractor, like PCA.

We first look at the case where the attacker only has access to T and r. To succeed in attacking the scheme, the attacker will need to break AES to retrieve the key. If the key is obtained, the secret s can only be recovered by correctly inverting the cryptographic hash function. In this situation, the security of both s and the key depend entirely on the encryption scheme and the hash function used.

If the encryption scheme and hash function are compromised, s could be recovered. Even then, the user's biometric cannot be rebuilt without additional knowledge. As such, if no leaks occurred, the scheme is both more private and secure than classical biometric authentication, with the security of the key and secret relying on the security of the symmetric cipher and the hash function used.

If only h is leaked or made public, an attacker does not gain an advantage on guessing s or recovering a biometric, as h is an offset from the lattice. If s is somehow obtained while the rest of the data is secure, the attacker can compute $s + h - r = p + \bar{x}$. This does not reveal any information on the biometrics, since both p is secret and the lattice from which p is sampled are unknown.

The next situation we consider is the one where the lattice code is leaked, but not h. In this case, the attacker could use the lattice to find s through a search over all the lattice points. Indeed, the attacker can search over $m' \in \mathbb{Z}^n$ and compute $s' = \mathbf{G}m' + r$ then hash s' until the correct key to decrypt T is found. Such an attack is made prohibitively hard by the random codeword shift that can increase the search space to by at least $(2\lambda)^n$ possibilities, where n is the size of the templates. By choosing a large enough λ, the search space can be made larger than that of the encryption key itself.

If both the lattice code and h are obtained, then the attacker can now build a list of possible vectors for $\overline{x}$ by computing $\bar{x}' = \mathbf{G}m' + h$. The real value of the randomized biometric $\bar{x}$ used at enrolment cannot be confirmed without any additional knowledge. Furthermore, this does not provide any new information about s, since only the quantization of $\overline{x}$ is used in the construction of s.

We do note that the list of possible values for $Q(\overline{x})$ appears to be affected by the feature extractor used. For example, when using PCARP, the user's biometrics in our tests were centered around the origin, with their closest lattice points being $\mathbf{G}m$ for m with components mostly in $\{0, \pm 1, \ldots, \pm 5\}$. RP-RP on the other hand saw components for each user varying over a bigger range with components of m each varying between -1000 and $+1000$. As such, knowledge of the feature extractor can be used to limit the number of potential candidates for $Q(\overline{x})$. In particular, the list is significantly smaller when using PCARP than when using RP-RP.

This information can be used by the attacker if, along with the lattice code, the codeword shift p is also leaked. This can greatly reduce the search space for s and make a brute force search more viable, as trying all the potential candidates first will likely lead to the reconstruction of s. This can be an advantage if the list of potential candidates is small, like when PCARP is used.

If s is recovered, and both p and the lattice are known, then the attacker can find $Q(\overline{x})$ the lattice point closest to $\overline{x}$ by computing $q = s - r - p$. If h is also known, then they can recover $\overline{x}$ as $\bar{x} = q - h$.

In the case where $\overline{x}$ is recovered, then we are in the same situation as if one randomized biometric is leaked in a classical authentication system. Using RP, as shown in [WP10], provides privacy and re-usability to the biometrics even if the RP matrix is leaked. With sufficient reduction to the dimension of at least half with random projections, the original biometrics cannot be fully reconstructed from the randomized features.

Consequently, we can see that this scheme offers more privacy and security than a classical biometric authentication scheme. Indeed, a partial leak of data with this scheme may not allow an attacker to recover any biometric features, while a full leak provides a randomized biometric for a user. Comparatively, a leak in a classical biometric system directly provide (randomized) biometrics features.

If any private data is leaked, a user can simply re-enroll with a new set of biometrics and new user parameters. In the case of any sign of compromised data, users can be protected by, for example, revoking the validity of any token currently in circulation.

As previously mentioned, a fuzzy extractor scheme used in an authentication setting essentially convert the problem of securely storing biometric features to that of securely storing cryptographic keys. In this scheme, the keys needing to be stored are the three seeds k_1, k_2, k_3 and the pointer for which LDLC matrix $\mathbf{H}$ is used by the user.

5.5 Experimental Results on the AR Database

In this section, we present experimentation results of this LDLC fuzzy extractor scheme using the same subset of the AR database [MB98] used in the previous part of this work and we compare this fuzzy extractor scheme to our improved scheme from Chapter 3.

The image set used consists of two different sets : a training set for PCA and a testing set for the fuzzy extractor. The training set contains 67 different persons each with 5 front

facing images with different settings (neutral expression, angry expression, neutral with left lighting, neutral with right lighting, neutral with both lighting). The testing set contains 26 new persons with 7 images each (the same 5, with additional neutral and angry taken 2 weeks later).

The only pre-processing done to the images was to convert them to gray-scale and crop them into a portrait ratio of 768×576 to fit the ratio of passport pictures. More details on the the subset of the AR database used, and the reason for those choices can be found in section 3.4.1 of this work and the list of images can be found in Appendix A.

In the following tests, we compared a few variables: the feature extractor used (PCARP and RP-RP), the final number of components of the extractors (and thus the size of the LDLC used), the set of enrolment images, and the effect of the scaling parameter β. Finally, we will compare the results with those obtained from the scheme in Chapter 3.

The code for the experiments was written in Python using the NumPy [Oli06] and OpenCV [Bra00] libraries. OpenCV was used to import and work with the images, as well as provide an open source implementation of PCA. The rest of the code was original, including the complete implementation of LDLC, from the generation of the matrix to the decoder.

5.5.1 Test Methodology and Implementation Details

The tests were in a similar fashion as the one from Chapter 3 and they were done as follows. If PCA was used as the feature extractor, PCA was first trained on the set of 67 users. Since training PCA is deterministic, the PCA extractors were saved for ease of use. Then, we separate the images for each user of the testing set into two disjoint sets : the enrolment images and the authentication image(s).

Each test was done by first selecting:

1. the final size of the templates, or the number of components,
2. the enrolment and authentication image sets, and
3. the feature extractor used (PCA, PCARP, RP-RP),
4. the scaling value β.

Once the parameters are chosen, the users were then enrolled and we tried to authenticate each user using the authentication images of all users. We note that when RP is used, each user gets assigned their own random RP extractor.

The result of each authentication is saved along with the "true" expected result. From these results we can compute various statistics using the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Unlike the component-based scheme from the previous chapter, we do not have access to a score that we can use to compute the equal error rate (EER) due to the nature of the lattice decoder.

Furthermore, the (Euclidean) distance between a biometric and the "correct" lattice point is not a good indicator of where a value will be quantized. Indeed, we know information about the volume of the area that will quantize to a lattice point, but this area will

not be in the shape of a sphere. As such, quantizing to a given point depends both on the distance from a lattice point an input is and in which direction that particular input lies with regard to the lattice point.

We chose to do the tests and comparisons for a final component size of 20, 40, 50 and 100. When using RP as a randomization step, we chose to reduce by half of the first extractor, as was done in [WP10] to compare PCA and PCARP. This means that in the case of a PCARP or RP-RP, the target size of the first extractor is, respectively 40, 80, 100 and 200.

With regards to the images used in the test, each of the 26 users have seven (7) images. Each of those images was labelled from 0 to 6, and either one or two images were chosen for the authentication set, while the rest were used in the enrolment procedure. We choose the same image label for enrolment and authentication sets across all users of a test.

Each user generates a personal LDLC matrix $\mathbf{H}$ at enrolment. Since we are using a relatively small size of features, we use the proposed parameters from [SFS08] of $d = 3$ and the generating sequence of $(1, 0.57735, 0.57735)$, with $\frac{1}{\sqrt{3}} \approx 0.57735$, for all users. We fixed the number of iterations of the decoder at 100 iterations.

Since generating many suitable Latin square matrices for an LDLC can be time consuming, we first generated a series of such Latin squares using seeded random number generators and using Algorithm 5. These Latin squares were stored as a pair of lists, one is a series of permutations and the other a series of sign vectors. Using these two lists, the algorithm can easily be completed at a later time to easily construct a Latin square with any generating sequence of the right size. An advantage of this approach is that the permutation representation of a Latin square is easier to store than a full matrix.

A user being enrolled in the system would then randomly select one of the seeds used to generate the LDLC and use that particular pre-generated lattice code, scaling the values appropriately using the parameter α.

Finally, to evaluate the performance of these tests we look at the system outcomes (TP, FN, FP, and TN) and the following derived statistics:

1. the true positive rate (TPR), given by $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$,
2. the false positive rate (FPR), given by $\text{FPR} = \frac{\text{FP}}{\text{TN+FP}}$,
3. the accuracy (ACC), given by $\text{ACC} = \frac{\text{TP+TN}}{\text{Total Observations}}$,
4. the positive predictive value (PPV), given by $\text{PPV} = \frac{\text{TP}}{\text{TP+FP}}$.

5.5.2 Experimental Results

Enrolment Sets

We begin by looking at the effect of the enrolment set on the performance of the extractor where we use either 5 or 6 of the 7 images for enrolment and while using PCARP as a feature extractor. Similarly to our improved scheme from Chapter 3, the choice of the enrolment and authentication images predictably affects the outcomes of the scheme.

When considering an enrolment set of 6 images, there is a lot of differences in the outcomes of the scheme. For example, depending on the set, we could obtain various combination of Low to High TPR, Low to High FPR, and relatively varying PPV. Some overview of test values can be found in table 5.1, showing how the values can vary depending on which image is chosen. The results contain a similar variation of outcomes when the enrolment set is of size 5.

Image #	TPR	FPR	ACC	PPV
0	0.846	0.360	0.648	0.086
1	0.423	0.185	0.800	0.084
2	0.462	0.155	0.830	0.106
3	0.808	0.235	0.766	0.121
4	0.885	0.245	0.760	0.126
5	0.731	0.152	0.843	0.161
6	0.308	0.095	0.882	0.114

Table 5.1: Overview of test results for different enrolment sets of 6 images, using PCARP with feature size 20, and $\beta = 1$. The sets are defined by the authentication image. This table contains one outcome out of many for each of the different authentication image sets. Other outcomes were very similar to those presented, with some having slightly higher/lower values for all the different statistics. This shows that the enrolment and authentication sets are an important factor to the performance of the scheme.

Such differences arising from the choice of enrolment and authentication sets was already expected, as we had observed the same results previously in Chapter 3. A study of what types and amount of images (poses, expression, etc) is best for such a system would be a boon, but it is out of the scope of this work. As such, the rest of the tests will be done using one set of enrolment of 6 images.

The Effect of β

β	TP	FN	FP	TN	β	TP	FN	FP	TN
0.1	0	26	1	649	3	20	6	368	282
0.5	11	15	51	599	5	23	3	503	147
0.75	11	15	57	593	10	23	3	509	141
1	19	7	126	524	15	24	2	613	37
1.5	22	4	193	457	20	26	0	626	24
2	22	4	280	370	50	26	0	650	0

Table 5.2: Test results for different values of β, using PCARP with feature size 20. The β value causes the expected behaviour of controlling the strictness of the scheme. As β scales the channel variance the LDLC is designed for, it ends up scaling the volume of the lattice. This explains the non-linear effect of the β values.

Next, we look at the effect of the β parameter. Increasing the value of β will generally decrease strictness of the scheme, allowing for more TP and FP, while decreasing its value

increases the strictness of the scheme. We do note that, due to the nature of the soft decoder of LDLC and the different RP extractors for each user and test, relatively small changes in the value of β might not be directly apparent in the scheme's outcomes. An example on the effect of the value of β can be found in Table 5.2.

As can be seen in the table, the effect of β was as expected for a scaling parameter. Recall that β acts by scaling the channel noise for which the lattice codes are designed. This, in effect, means that β is scaling the volume of the lattice. This explains the non-linear effect that β has on the outcomes of the scheme.

Feature Extractor and Number of Components

The next parameters we look at are the choice of feature extractor and the number of components. We tested each combination multiple times and Tables 5.3 and 5.4 present some of the test outcomes using PCARP and RP-RP as extractors for feature sizes $n \in \{20, 40, 50, 100\}$. Some of the variance caused by generating RP extractors and the random selection of an LDLC matrix for each user can be seen from these results.

n	TP	FN	FP	TN	TPR	FPR	ACC	PPV
20	18	8	150	500	0.692	0.231	0.766	0.107
20	18	8	99	551	0.692	0.152	0.842	0.154
40	17	9	82	568	0.654	0.126	0.865	0.172
40	18	8	125	525	0.692	0.192	0.803	0.126
50	15	11	48	602	0.577	0.074	0.913	0.238
50	20	6	101	549	0.769	0.155	0.842	0.165
100	16	10	75	575	0.615	0.115	0.874	0.176
100	11	15	33	617	0.423	0.051	0.929	0.250

Table 5.3: Test results for using PCARP with different feature sizes n, with $\beta = 1$. For a fixed value of β, the LDLC scheme performs similarly with regards to the derived statistics (TPR, FPR, ACC, PPV) across all the tested templates size n. Although the statistics are similar, the number of TP and FP appears to reduce as n increases.

From our tests, the two feature extractors offer a similar performance, with PCARP showing more consistency in the outcomes due to the fact that the PCA extractor is entirely determined by the training set used. The choice in the number of final components n also does not appear to greatly affect the derived statistics for a fixed value of β, but it does appear to affect the total numbers of positive and negative observations. Indeed, although there is some variation, ACC and PPV are generally similar across all choices of n. On the other hand, we can observe a slight drop in TPR and FPR as n increases, which is more observable for larger n. This means that we get around the same ratio of TP to FP, regardless of the choice of n, even if the amount of TP and FP are lower at higher values of n.

We have also tested the scheme using only PCA and no randomization through RP. Some outcomes using features of size 20 can be found in Table 5.5. A first observation is

n	TP	FN	FP	TN	TPR	FPR	ACC	PPV
20	21	5	147	503	0.808	0.226	0.775	0.125
20	16	10	90	560	0.615	0.138	0.852	0.151
40	12	14	24	626	0.462	0.037	0.944	0.333
40	21	5	95	555	0.808	0.146	0.852	0.181
50	13	13	85	565	0.500	0.131	0.855	0.133
50	19	7	98	552	0.731	0.151	0.845	0.162
100	12	14	47	603	0.462	0.072	0.910	0.203
100	18	8	61	589	0.692	0.094	0.898	0.228

Table 5.4: Test results for using RP-RP with different feature sizes n, with $\beta = 1$. The results using RP-RP behave similarly to when using PCARP (see Table 5.3), but there is a lot more variance caused by the RP extractor that replaced PCA. The derived statistics appear to be very similar regardless of the number of componets n, with the number of TP and FP being slightly lower at higher number of components.

that when using pure PCA, we need to use a smaller value of β values to obtain meaningful observations that do not authenticate the majority of the inputs. This is easily explainable by the nature of PCA, where most of the data variance is encoded in the first few components, with less and less variance being in the later components of the extracted features.

We can then observe that at a given TPR, the PCA extractor provides performance that is similar to those using a PCARP extractor with regards to the ACC and PPV. Furthermore, we can conclude that using RP to randomize and protect the templates in case of a leak does not appear to negatively affect the performance of the scheme.

β	TP	FN	FP	TN	TPR	FPR	ACC	PPV
0.25	2	24	12	638	0.077	0.018	0.947	0.143
0.25	3	23	9	641	0.115	0.014	0.953	0.250
0.5	14	12	62	588	0.538	0.095	0.891	0.184
0.5	13	13	57	593	0.500	0.088	0.896	0.186
0.75	21	5	153	497	0.808	0.235	0.766	0.121
0.75	24	2	140	510	0.923	0.215	0.790	0.146

Table 5.5: Test results for using PCA and no randomization, for $n = 20$ and different values of β. Using PCA presents similar performance as using PCARP when looking at similar TPR values. This tell us that using RP to protect templates does not appear to negatively affect the performance of the scheme.

Comparing with the Component-based Scheme

Finally, we compare these results with our improved scheme from Chapter 3. To do this, we tested the scheme using PCARP, the same enrolment set and the same number of final components. We then looked at the statistics when it has a similar TPR as the LDLC

scheme using PCARP. These results are found in Table 5.6, and we will compare them to the LDLC results from Table 5.3.

As can be seen, the LDLC scheme perform worse than the per-component scheme. At equivalent TPR, the LDLC scheme has a FPR varying between 0.07 and 0.23, while our component-based scheme has an FPR of less than 0.03 at these same TPR values. In a similar fashion, the PPV values of the LDLC scheme, hovering between 0.10 to 0.25, are very low compared to the improved Sutcu scheme that has PPV values ranging from 0.50 to 0.85.

n	TP	FN	FP	TN	TPR	FPR	ACC	PPV
20	18	8	15	635	0.692	0.023	0.966	0.545
20	19	7	19	631	0.731	0.029	0.962	0.500
40	17	9	5	645	0.654	0.008	0.979	0.773
40	18	8	12	638	0.692	0.018	0.970	0.600
50	15	11	3	647	0.577	0.005	0.979	0.833
50	20	6	18	632	0.769	0.028	0.964	0.526
100	16	10	3	647	0.615	0.005	0.981	0.842
100	23	3	10	640	0.885	0.015	0.981	0.697

Table 5.6: Test results for the Chapter 3 scheme using PCARP. At equivalent TPR, the component based scheme performs better than the LDLC scheme by having significantly higher PPV and lower FPR.

The generally poor performance of the LDLC scheme on this database can be explained by our assumption on the distribution of the noise. Indeed, a LDLC is a code designed to correct additive white Gaussian noise. Since the biometric variance of a user's feature using PCA, PCARP or RP-RP does not appear to be like white noise, the LDLC scheme performs poorly compared to the component based scheme.

Although RP help by spreading the noise across all the components, the resulting distribution is still not white. This is highlighted by analyzing the variance of each components for any given users, where we get that the highest variance is often close to a hundred times larger than the smallest variance.

Because Lattice codes are designed to correct white Gaussian noise, the performance of the quantizer, and thus the fuzzy extractor, takes a hit. As such, for this LDLC feature extractor to be suitable for facial biometrics, some transformations should be added to the templates to spread the noise and make it more akin to a AWGN channel.

5.6 Tests using AWGN

As seen in the previous section, the low performance of the LDLC fuzzy extractor on our data set appears to be linked to the nature of the noise. More specifically, when using a feature extractor like PCA, PCARP or RP-RP, the biometric variance of the features is not similar to white Gaussian noise. In this section, we look at how our LDLC fuzzy extractor would behave if the user's noise was like that of an AWGN channel.

5.6.1 Test Methodology

Our approach for these tests will be the same as previously, with the exception that we will use simulated data derived from the biometrics as our data inputs. In addition, we will only look at the PCARP extractor for the feature data, since we know the general effect of the extractors on the scheme.

The simulated data will be generated as follows. First, using PCARP, we extract the features of each users U_i and we compute the average image $\overline{x}_i$ as well as the sample variances for each components for that user by using all seven images available in our dataset. Then, we pick a noise variance σ_i^2 that will represent the simulated feature noise for this user. The value chosen for σ_i^2 will be a parameter in the tests. This will act as the data obtained from the enrolment set.

The inputs used to test authentication will then be $\overline{x}_j + n_j$, with n_j being a vector of i.i.d. Gaussian noise sampled from $N(0, \sigma_j^2)$. The rest of the fuzzy extractor scheme remains the same.

This simulation will allow us to see how the scheme performs when the features of each user have a variance similar to white Gaussian noise, while also looking at data that has the same global distribution of user in the feature space as our facial image data set. In our tests, we look at three different variances for the noise : the smallest observed variance across the components (the *min variance*), the biggest observed variance (the *max variance*), and the middle variance (the *mid variance*). These three variances will respectively represent a small, large and medium amount of overlap of the user's feature distributions.

These variances are computed as follows. If $v_{i,1}, \ldots, v_{i,n}$ are the observed variance for each component for user i, then we define the *min variance* as $\sigma^2_{min,i} = \min_{k=1,n} v_{i,k}$, the *max variance* as $\sigma^2_{max,i} = \max_{k=1,n} v_{i,k}$, and the *mid variance* as $\sigma^2_{mid,i} = (\sigma^2_{min,i} + \sigma^2_{max,i})/2$.

Finally, in any given test, we generate a new noisy template for a user whenever we try to authenticate using that particular user. In the following test, we will do 10 authentication attempts per user. This yields a total of 6760 attempts per test, 260 of which have a positive condition.

5.6.2 Experimental Results

In this section, we will look at how the scheme performs when using the simulated data we just defined. We will first look at the effect of individual RP extractors on the outcomes. Then we look at how the number of components and the chosen data variance affect the system, before finally comparing the LDLC scheme with the component based scheme on AWGN-like inputs.

The Feature Extractor

We begin by looking at how the different RP extractors used for randomizing each user's templates affects the results of the scheme. The regular approach is that each user has

their own PCARP extractor. Table 5.7 contains the outcomes of five different tests using the simulated data and using the scheme as designed, with $n = 20$ components and using the mid variance. It can be seen that the outcomes are generally consistent.

On the other hand, Table 5.8 presents some results for five tests where all the users had the same PCARP extractors, where we used the mid variance as noise and with $\beta = 1$. As can be seen and was expected, the RP extractor adds some variation on the final outcome of the test. The outcomes are also similar to when each user has their own RP extractor, with the difference that using a unique RP extractor leads to more variation in the number of TP and FP.

Test nb	TP	FN	FP	TN	TPR	FPR	PPV
1	25	235	35	6465	0.096	0.005	0.417
2	37	223	51	6449	0.142	0.008	0.420
3	38	222	60	6440	0.146	0.009	0.388
4	37	223	38	6462	0.142	0.006	0.493
5	34	226	63	6437	0.131	0.010	0.351

Table 5.7: Tests using a different PCARP extractor for all users. Using the mid variance for noise, 20 components and $\beta = 1$. The final outcomes appear to be consistent throughout the different tests.

Test nb	TP	FN	FP	TN	TPR	FPR	PPV
1	40	220	70	6430	0.154	0.011	0.364
2	41	219	69	6431	0.158	0.011	0.373
3	51	209	37	6463	0.196	0.006	0.580
4	35	225	37	6463	0.135	0.006	0.486
5	32	228	14	6486	0.123	0.002	0.696

Table 5.8: Tests using a single PCARP extractor for all users. Using the mid variance for noise, 20 components and $\beta = 1$. The results across the different tests are similar to the results when each user has a different RP extractor, but they have a bit more variance in their final outcomes.

These results hold true for both cases even when using different values for β. This particular observation essentially allows us to look at the effect of the other parameters (β, n and the noise variance) while using a single PCARP extractor for all users.

Number of Components

For the next series of tests, we used the same PCARP extractor for all users and choice of parameters to better see their effects. We begin by looking at the effect of the number of components. Table 5.9, 5.10 and 5.11 respectively contain the outcomes for various values of beta for $n = 20$, 50 and 100. The first observation confirms that β has the desired effect of controlling how strict the scheme is, with higher values of β increasing the FPR and TPR by authenticating inputs more often. The effect of β is also consistent with the

previous observations from using the AR data set in our tests, as the degree at which β affects the results depends on the number of components, with small changes in its value being less felt when n is larger.

β	TP	FN	FP	TN	TPR	FPR	PPV
0.5	1	259	2	6498	0.004	0.000	0.333
1	41	219	69	6431	0.158	0.011	0.373
1.5	105	155	488	6012	0.404	0.075	0.177
2	145	115	942	5558	0.558	0.145	0.133
6	214	46	4399	2101	0.823	0.677	0.046
10	238	22	5507	993	0.915	0.847	0.041

Table 5.9: Tests using one PCARP extractor for all users. Using the mid variance for noise and 20 components. When compared with the results for 50 and 100 components, the scheme seems to perform better at a lower number of components.

The second observation is that the scheme performs better with a smaller number of components. Indeed, at a similar TPR, the FPR for $n = 20$ is lower than for $n = 50$ and $n = 100$. For example with 20 components and a TPR of 0.404 we have an FPR of 0.075. In comparison, a similar TPR of about 0.40 gives an FPR of 0.250 when $n = 50$ and an FPR of 0.370 when $n = 100$. Similarly, the PPV at that TPR for $n = 20$ is comparatively higher at 0.177 versus 0.061 for $n = 50$ and 0.041 for $n = 100$. We can conclude from this that a lower number of components appears to yield better results for this fuzzy extractor, especially if the input data has AWGN-like variation.

β	TP	FN	FP	TN	TPR	FPR	PPV
0.5	0	260	0	6500	0.000	0.000	N/A
1	1	259	0	6500	0.004	0.000	1.000
2	58	202	100	6400	0.223	0.015	0.367
6	106	154	1627	4873	0.408	0.250	0.061
10	132	128	2253	4247	0.508	0.347	0.055

Table 5.10: Tests using one PCARP extractor for all users. Using the mid variance for noise and 50 components.

β	TP	FN	FP	TN	TPR	FPR	PPV
1	0	260	0	6500	0.000	0.000	N/A
2	13	247	7	6493	0.050	0.001	0.650
6	26	234	373	6127	0.100	0.057	0.065
10	36	224	631	5869	0.138	0.097	0.054
20	64	196	1371	5129	0.246	0.211	0.045
40	102	158	2405	4095	0.392	0.370	0.041

Table 5.11: Tests using one PCARP extractor for all users. Using the mid variance for noise and 100 components.

Noise Variance

We now look at how much the noise variance affects the final results. Tables 5.12 and 5.13 contain test outcomes using the min and max variances, respectively, for 20 components and with the same PCARP extractor as the test presented in Table 5.9. These three tests using min, mid and max variance simulate situations where the user's feature distributions have a (relatively) low, medium and large amount of overlap in the feature space.

As we can see, the fuzzy extractor appears to perform well when the user's feature distributions have a minimal overlap in the space. This is an expected result, since the more overlap there is in the input data, harder it is to differentiate the users. Indeed, the higher the amount of overlap is, the worse the performance of any classification scheme will be.

β	TP	FN	FP	TN	TPR	FPR	PPV
0.5	4	256	0	6500	0.015	0.000	1.000
1	32	228	0	6500	0.123	0.000	1.000
2	159	101	2	6498	0.612	0.000	0.988
6	177	83	44	6456	0.681	0.007	0.801
10	183	77	250	6250	0.704	0.038	0.423

Table 5.12: Tests using one PCARP extractor for all users. Using the *min* variance for noise and 20 components. As can be seen, when the amount of overlap between the users is low, the scheme can perform extremely well.

β	TP	FN	FP	TN	TPR	FPR	PPV
0.5	4	256	68	6432	0.015	0.010	0.056
1	33	227	567	5933	0.127	0.087	0.055
2	162	98	2026	4474	0.623	0.312	0.074
6	241	19	5532	968	0.927	0.851	0.042
10	255	5	6134	366	0.981	0.944	0.040

Table 5.13: Tests using one PCARP extractor for all users. Using the *max* variance for noise and 20 components. Concurently, when the amount of overlap is high, the performance of the scheme is greatly limited.

Comparing to the Component-based Scheme

Finally, we look at how our scheme from Chapter 3 performs when the variance on the user's features is white Gaussian noise. We will compare these results to the performance of the LDLC fuzzy extractor. This comparison of the schemes will use the same unique PCARP extractor for each user, and we will look at how the scheme from Chapter 3 performs at the same non-zero FPR as the LDLC scheme in this simulation.

Tables 5.14, 5.15 and 5.16 contain the resulting data for the scheme from Chapter 3 at identical FPR as the data in the previous tables for the LDLC scheme (respectively Tables 5.9, 5.12 and 5.13), using the three different noise variance.

As can be seen, the LDLC scheme performs better when a low FPR is required (relatively to the input distributions). It provides a higher TPR and PPV than the Ch. 3 scheme, with the differences slowly shrinking as the FPR increases. At some point the differences flips for large enough FPR, with the LDLC scheme having smaller TPR and PPV than the Ch. 3 scheme. The differences in performance between the two scheme appear to be less pronounced when the input distributions have more overlap like in the case where we used the max variance for the noise.

We also observed the same result when looking at fixed TPR, where LDLCs had smaller FPR with at a low TPR and higher FPR at a high TPR value.

	Ch. 3 scheme			LDLC Scheme		
FP (FPR)	TP	TPR	PPV	TP	TPR	PPV
2 (0.000)	0	0.000	0.000	1	0.004	0.333
69 (0.011)	14	0.054	0.169	41	0.158	0.373
488 (0.075)	76	0.292	0.135	105	0.404	0.177
942 (0.145)	125	0.481	0.117	145	0.558	0.133
4399 (0.677)	256	0.985	0.055	214	0.823	0.046
5507 (0.847)	260	1.000	0.045	238	0.915	0.041

Table 5.14: Comparison of the two fuzzy extractor at fixed FPR, for 20 components using the mid variance. The LDLC scheme performs significantly better when a low FPR is required by having both higher TPR and PPV than the component-based scheme.

	Ch. 3 scheme			LDLC Scheme		
FP (FPR)	TP	TPR	PPV	TP	TPR	PPV
2 (0.000)	102	0.392	0.981	159	0.612	0.988
44 (0.007)	170	0.654	0.794	177	0.681	0.801
250 (0.038)	204	0.785	0.449	183	0.704	0.423

Table 5.15: Comparison of the two fuzzy extractor at fixed FPR, for 20 components using the min variance. Again, the LDLC scheme performs better at low FPR.

	Ch. 3 scheme			LDLC Scheme		
FP (FPR)	TP	TPR	PPV	TP	TPR	PPV
68 (0.010)	0	0.000	0.000	4	0.015	0.056
567 (0.087)	22	0.085	0.037	33	0.127	0.055
2026 (0.312)	151	0.581	0.069	162	0.623	0.074
5532 (0.851)	259	0.996	0.045	241	0.927	0.042
6134 (0.944)	260	1.000	0.041	255	0.981	0.040

Table 5.16: Comparison of the two fuzzy extractor at fixed FPR, for 20 components using the max variance. Although the LDLC scheme still performs better in this situation at low FPR, the difference is relatively small. This is due to the high overlap of the features making classification harder in general.

Finally, we can see from these various tests that the LDLC scheme performs better than the component-based scheme from Ch. 3 when the biometric variance of the users is AWGN-like in most situations, regardless of the noise variance. The LDLC scheme does perform worse when we are aiming for high FPR or TPR values, but applications requiring such things are rare in an authentication setting. Further more, with our particular test data, the scheme seems to perform better at lower number of components (like 20) compared to higher numbers (like 100). The accuracy of the scheme is highly dependent on the distribution of the inputs and how much overlap there is between the feature distribution of the users.

This can mean that in applications where failures to authenticate are more costly than false positive, a component-based fuzzy extractor (or another authentication method entirely) might be preferable.

5.7 Conclusion

In this chapter we presented a construction for a fuzzy extractor using LDLC as the quantizer and work directly in the Euclidean space. This extractor is also resistant to various attacks and partial leaks of stored information like random seeds. We then tested this scheme on facial biometrics and compared it with the scheme from Chapter 3.

The results of these tests show that the LDLC scheme performed worse than the improved Sutcu et al.'s scheme on the same data set of images. This is in huge part due to the variance on the biometric templates not being like white Gaussian noise. The number of components used in the scheme does not appear to greatly affect the performance using this particular data set, but additional tests using inputs with AWGN point to lowered performance as the number of components increases. This means that good performance can be attained at a lower decoding complexity.

We then looked at the performance of this LDLC scheme with features having a white Gaussian noise and compared how the two schemes performed using such inputs. Under these circumstances of AWGN-like user feature distributions, we see that not only is the LDLC scheme performing well, it also performs better than the Ch. 3 scheme when a low FPR is desired. The scheme's performance is, as expected, highly linked to the amount of overlap in the feature space that the user's distributions have, with more overlap visibly degrading the performance. We did not compare the results from section 5.5 to those of section 5.6, as they involve inputs of different natures.

It also performs better in situations where the need for a low FPR is preferred than situations where a high TPR or FPR is desired. For example, an LDLC scheme is well suited for the original setting of this research that is authentication in a border crossing situation. In such a setting, a very low FPR is more desirable than a high TPR, as any false positive can have grave consequences while a false negative can be easily dealt with through human interaction.

This LDLC scheme is thus more suited to biometrics, or any other form of noisy inputs, that have distributions closer to white Gaussian noise or distributions that can be modelled

or approximated using white Gaussian noise. Indeed, as constructed, this fuzzy extractor can work directly with any other continuous-source inputs. It is not limited to facial images and, with the nature of LDLC allowing the decoding of large values, it may be of interest to see how it performs with different inputs like sound, speech or any data captured over a certain period of time.

If the scheme is to be used on facial biometrics to achieve better performance than a component-based quantization scheme, additional work should be done to transform or normalize the templates to better fit an AWGN model. Furthermore, this transformation should, if at all possible, try to minimize the amount of overlap of the user features to improve the performance.

Further Discussion

We note that spreading the noise of an input to make it appear more Gaussian or white is not trivial and it is something we lacked the time to explore deeply in this work. In particular, simply spreading the noise and making it white may likely remove information from the biometrics that is used in classification and recognition. A technique to normalize or make a user's biometrics noise more white, would also need to conserve, to some extent, the original information.

In a different direction to making the noise more white, we briefly explored the thought of normalizing a user's biometrics before quantization. Recall that we compute the sample variance v_i of each component. Instead of taking the maximum as the channel noise, σ^2, we considered using the v_i values to normalize the distribution of each component and the inputs before quantization. This didn't yield any notable results. Furthermore, it would require us to store both the v_i and the mean vector to properly normalize the inputs, which would defeat the point of using a fuzzy extractor for facial authentication.

A last approach we considered briefly that didn't yield results was to use various different LDLC decoders in parallel. The idea was to, for example split the features into two vectors of smaller size: one containing the half of the component with the highest variance and the other half the components with the smaller variances. This could be done by reordering the components and storing that permutation. In this situation, the decoding step would be "split" into two by using two (smaller) LDLC, one for each half of the features. Our preliminary test using this approach did not provide interesting results, as in authentication attempt failed to correctly recreate the key. We did not explore this approach in more depth due to a lack of time, but we believe it may be worth further exploration.

Chapter 6

Conclusion

In this thesis, we presented two fuzzy extractors for continuous features and we applied them to face biometrics. The first one is an improved version of the scheme given by Sutcu et al in Chapter 3, while the other one is a construction using LDLC as a quantizer in Chapter 5. Both were tested on the same face image dataset which was a subset of the AR database. We also compare them using simulated data with additive white Gaussian noise.

The new results presented in the first part of this work were the improvements to the fuzzy extractor for faces from [SLM09]. Through an implementation of their scheme, we found areas where the security and privacy could be improved. Due to the randomization method used, their scheme also performed poorly when tested on the AR database, not authenticating any user regardless of the choice of parameters.

The improvements we propose make the fuzzy extractor more secure against partial leaks of data and brute force attacks, as well as make the scheme more practical from an implementation perspective. The scheme also offers good performance with only minimal preprocessing. The increase in security was achieved by changing how the codebooks were constructed as well as by using random projections as the randomization tool instead of random square matrices.

The addition of a random codeword shift also has the second advantage to add more entropy to the key generated by the fuzzy extractor. This can, if the shift remains securely stored, alleviate the low information value of facial images shown in [YA10] or in any other low information inputs.

The implementation improvements were also achieved with a change in the codebook construction. Although some prior empirical knowledge is needed at the setup phase, the construction of the global codebook is now independent from the users' data. This allows the system to be set up without needing the users to be already enrolled. This also allows for enrolment of a user to happen at any time. The final improvement we introduce is a user-based scaling parameter that allows fine tuning the strictness of the authentication to a desired FAR/FRR ratio depending on the system requirements. The choice of the parameters can benefit from additional knowledge about the inputs to be used, and the

careful selection of the scaling parameter may also benefit from using a knowledge-based system, for example.

Even when using a small set of mostly unprocessed images, this improved scheme still gave very acceptable performance. Due to various resource and time constraints, as well as a lack of free or open source code for extensive preprocessing of facial images, we only did minimal preprocessing to the images. It is of note that preprocessing techniques for facial recognition are something that has been well studied and many techniques are generally implemented in facial recognition softwares. As such, any additional preprocessing should greatly improve the performance of the scheme.

The scheme can also benefit from additional knowledge on how to better select the enrolment and authentication images, as it is a major factor in the accuracy and performance of the scheme. We also note that in the context of this work and the project from which it derived, facial recognition can be quite challenging. Indeed, in a border crossing setting, the capture of the biometrics at enrolment and at authentication is not done in a controlled environments. Furthermore, there might be a large time differential of weeks or months between the two phases, in which the person might change various things about themselves from using (different) make-up or not, to changing their hairstyle or the accessories (like glasses) they wear. These are not factors we investigated in detail in this thesis due to the image database we used, but these should be taken into consideration for such an application of biometrics in a real situation.

In the second part of the thesis, we constructed a fuzzy extractor in a similar fashion as the first scheme presented, but where we used LDLC as the quantizer instead of using a range-based per-component quantizer. We first presented how to construct and decode LDLC using their iterative decoders, then we gave the construction for our scheme. Our particular construction has the same security advantages as the improved scheme from the previous section, in that it is resistant to leaks and brute force attacks while also having the ability to be set up without needing users to be already enrolled.

We then implemented the scheme and tested it to see the effects of the parameters and compared it to the first scheme of the work. The performance of the LDLC scheme was generally stable when looking at the various parameters, but comparatively to our improved scheme, it did not perform as well. The main cause for this low performance is the nature of the noise in the biometric templates; approximating it as a white Gaussian noise increased the error rates. Since lattice codes are designed to correct white Gaussian noise, it is natural that their performance suffered on this data set. This was further confirmed by our simulations using features with additive white Gaussian noise distribution, where the LDLC scheme generally performed better than our improved component-based scheme, especially when a low false positive error rate is desired.

The application considered in this thesis for the fuzzy extractors was as a privacy-preserving facial authentication tool as a way to facilitate the border crossing of low risk travellers in airports who wished to opt-in. The keys generated would be used to help authenticate the traveller and allow them to "sign" travel documents by linking them to their biometrics. Fuzzy extractors allowed such an application of biometrics while never having the need to store biometrics.

As such, they can be used in various applications where one would be interested in generating keys or authenticators using biometrics without storing biometrics or where privacy is desired. They can also be of interest to small or medium enterprises who wish to use biometrics authentication but may not have the resources and money to comply with all the policies and restrictions, like the European GDPR, related to privacy and the use of biometrics.

Future Research

Following this work, a few possible avenue of research are available to further improve on these feature extractors.

The first avenue is to improve knowledge on how facial biometrics vary for each individual user. This additional knowledge can be used to answer various questions such as : "how many images should be taken for the enrolment of users" and "what pose or expression should the images have". Having guidelines for the enrolment could allow for a more precise generation of the codebooks to improve the performance of the schemes, and such knowledge can allow for better characterization of a user's biometric through a smaller number of images. It would also reduce the burden on the user at enrolment, as the number of required images could be reduced.

This would likely involve the creation of a new database and it can be a costly endeavour. Such research could directly improve the performance of our schemes, as well as guide the enrolment procedure of a real world application of this fuzzy extractor, or of any other facial recognition scheme. A good enough database may also be useful to see how facial biometrics can vary through time, as well as take into consideration many of the important factors in facial recognition such as illumination, poses, expression and other general changes in appearances of the users.

We note that some applications of a fuzzy extractor, like the border crossing scenario we considered in this thesis, can allow for controlled enrolment sets and authentication images (for example by screening the inputs at the source using various software). Such situations can greatly benefit from these guidelines.

A second possibility is to look at how various forms of additional preprocessing on the images will affect the performance of the schemes, as well as the use of more modern or commercial feature extractors. Such work can also benefit from a larger and more extensive database of facial images.

Other possible directions revolve around the LDLC scheme. One being looking at other types of biometrics with continuous features and exploring its performance on these other biometrics. Such types of inputs could be, for example, voice, sounds, or inputs that can vary through time like videos or sequences of movements. The LDLC extractor can be directly used regardless of the type of inputs, as long as the data can be meaningfully represented as vectors.

Another possibility is developing or applying a transformation on the templates that would align the noise to what lattice codes are better suited to correct. This could be

approached by trying to spread the biometric noise of the templates and make it like a AWGN channel to fully utilize the strength of a lattice code quantizer, or normalize the inputs without removing too much information. A final possibility would be to, instead of modifying the inputs, design specific types of lattice codes that could work better for the particular noise distribution of facial biometrics.

APPENDICES

Appendix A

Dataset Used

Here we list the subset of images from the AR dataset we used in our experiments.

A.1 The PCA Training Set

The set used to train PCA is the set of images with the following names:

m-028-1.png, m-028-3.png, m-028-5.png, m-028-6.png, m-028-7.png,
m-029-1.png, m-029-3.png, m-029-5.png, m-029-6.png, m-029-7.png,
m-035-1.png, m-035-3.png, m-035-5.png, m-035-6.png, m-035-7.png,
m-041-14.png, m-041-16.png, m-041-18.png, m-041-19.png, m-041-20.png,
m-042-14.png, m-042-16.png, m-042-18.png, m-042-19.png, m-042-20.png,
m-043-14.png, m-043-16.png, m-043-18.png, m-043-19.png, m-043-20.png,
m-044-14.png, m-044-16.png, m-044-18.png, m-044-19.png, m-044-20.png,
m-045-14.png, m-045-16.png, m-045-18.png, m-045-19.png, m-045-20.png,
m-046-14.png, m-046-16.png, m-046-18.png, m-046-19.png, m-046-20.png,
m-047-14.png, m-047-16.png, m-047-18.png, m-047-19.png, m-047-20.png,
m-048-14.png, m-048-16.png, m-048-18.png, m-048-19.png, m-048-20.png,
m-049-14.png, m-049-16.png, m-049-18.png, m-049-19.png, m-049-20.png,
m-051-14.png, m-051-16.png, m-051-18.png, m-051-19.png, m-051-20.png,
m-052-14.png, m-052-16.png, m-052-18.png, m-052-19.png, m-052-20.png,
m-053-14.png, m-053-16.png, m-053-18.png, m-053-19.png, m-053-20.png,
m-054-14.png, m-054-16.png, m-054-18.png, m-054-19.png, m-054-20.png,
m-055-14.png, m-055-16.png, m-055-18.png, m-055-19.png, m-055-20.png,
m-056-14.png, m-056-16.png, m-056-18.png, m-056-19.png, m-056-20.png,
m-058-14.png, m-058-16.png, m-058-18.png, m-058-19.png, m-058-20.png,
m-059-14.png, m-059-16.png, m-059-18.png, m-059-19.png, m-059-20.png,
m-060-14.png, m-060-16.png, m-060-18.png, m-060-19.png, m-060-20.png,
m-061-14.png, m-061-16.png, m-061-18.png, m-061-19.png, m-061-20.png,
m-065-14.png, m-065-16.png, m-065-18.png, m-065-19.png, m-065-20.png,
m-066-14.png, m-066-16.png, m-066-18.png, m-066-19.png, m-066-20.png,

m-067-14.png, m-067-16.png, m-067-18.png, m-067-19.png, m-067-20.png,
m-069-14.png, m-069-16.png, m-069-18.png, m-069-19.png, m-069-20.png,
m-070-14.png, m-070-16.png, m-070-18.png, m-070-19.png, m-070-20.png,
m-071-14.png, m-071-16.png, m-071-18.png, m-071-19.png, m-071-20.png,
m-072-14.png, m-072-16.png, m-072-18.png, m-072-19.png, m-072-20.png,
m-073-14.png, m-073-16.png, m-073-18.png, m-073-19.png, m-073-20.png,
m-074-14.png, m-074-16.png, m-074-18.png, m-074-19.png, m-074-20.png,
w-001-1.png, w-001-3.png, w-001-5.png, w-001-6.png, w-001-7.png,
w-002-1.png, w-002-3.png, w-002-5.png, w-002-6.png, w-002-7.png,
w-003-1.png, w-003-3.png, w-003-5.png, w-003-6.png, w-003-7.png,
w-004-1.png, w-004-3.png, w-004-5.png, w-004-6.png, w-004-7.png,
w-005-1.png, w-005-3.png, w-005-5.png, w-005-6.png, w-005-7.png,
w-007-1.png, w-007-3.png, w-007-5.png, w-007-6.png, w-007-7.png,
w-008-1.png, w-008-3.png, w-008-5.png, w-008-6.png, w-008-7.png,
w-009-1.png, w-009-3.png, w-009-5.png, w-009-6.png, w-009-7.png,
w-010-1.png, w-010-3.png, w-010-5.png, w-010-6.png, w-010-7.png,
w-011-1.png, w-011-3.png, w-011-5.png, w-011-6.png, w-011-7.png,
w-012-1.png, w-012-3.png, w-012-5.png, w-012-6.png, w-012-7.png,
w-013-1.png, w-013-3.png, w-013-5.png, w-013-6.png, w-013-7.png,
w-014-1.png, w-014-3.png, w-014-5.png, w-014-6.png, w-014-7.png,
w-015-1.png, w-015-3.png, w-015-5.png, w-015-6.png, w-015-7.png,
w-016-1.png, w-016-3.png, w-016-5.png, w-016-6.png, w-016-7.png,
w-017-1.png, w-017-3.png, w-017-5.png, w-017-6.png, w-017-7.png,
w-018-1.png, w-018-3.png, w-018-5.png, w-018-6.png, w-018-7.png,
w-019-1.png, w-019-3.png, w-019-5.png, w-019-6.png, w-019-7.png,
w-021-1.png, w-021-3.png, w-021-5.png, w-021-6.png, w-021-7.png,
w-022-1.png, w-022-3.png, w-022-5.png, w-022-6.png, w-022-7.png,
w-029-14.png, w-029-16.png, w-029-18.png, w-029-19.png, w-029-20.png,
w-030-14.png, w-030-16.png, w-030-18.png, w-030-19.png, w-030-20.png,
w-031-14.png, w-031-16.png, w-031-18.png, w-031-19.png, w-031-20.png,
w-032-14.png, w-032-16.png, w-032-18.png, w-032-19.png, w-032-20.png,
w-033-14.png, w-033-16.png, w-033-18.png, w-033-19.png, w-033-20.png,
w-034-14.png, w-034-16.png, w-034-18.png, w-034-19.png, w-034-20.png,
w-035-14.png, w-035-16.png, w-035-18.png, w-035-19.png, w-035-20.png,
w-036-14.png, w-036-16.png, w-036-18.png, w-036-19.png, w-036-20.png,
w-037-14.png, w-037-16.png, w-037-18.png, w-037-19.png, w-037-20.png,
w-038-14.png, w-038-16.png, w-038-18.png, w-038-19.png, w-038-20.png,
w-039-14.png, w-039-16.png, w-039-18.png, w-039-19.png, w-039-20.png

A.2 The Testing Set

The set of images used to test the fuzzy extractors is the set of images with the following names:

m-008-1.png,␣m-008-14.png,␣m-008-16.png,␣m-008-3.png,
␣m-008-5.png,␣m-008-6.png,␣m-008-7.png,
m-020-1.png,␣m-020-14.png,␣m-020-16.png,␣m-020-3.png,
␣m-020-5.png,␣m-020-6.png,␣m-020-7.png,
m-025-1.png,␣m-025-14.png,␣m-025-16.png,␣m-025-3.png,
␣m-025-5.png,␣m-025-6.png,␣m-025-7.png,
m-026-1.png,␣m-026-14.png,␣m-026-16.png,␣m-026-3.png,
␣m-026-5.png,␣m-026-6.png,␣m-026-7.png,
m-027-1.png,␣m-027-14.png,␣m-027-16.png,␣m-027-3.png,
␣m-027-5.png,␣m-027-6.png,␣m-027-7.png,
m-030-1.png,␣m-030-14.png,␣m-030-16.png,␣m-030-3.png,
␣m-030-5.png,␣m-030-6.png,␣m-030-7.png,
m-031-1.png,␣m-031-14.png,␣m-031-16.png,␣m-031-3.png,
␣m-031-5.png,␣m-031-6.png,␣m-031-7.png,
m-032-1.png,␣m-032-14.png,␣m-032-16.png,␣m-032-3.png,
␣m-032-5.png,␣m-032-6.png,␣m-032-7.png,
m-033-1.png,␣m-033-14.png,␣m-033-16.png,␣m-033-3.png,
␣m-033-5.png,␣m-033-6.png,␣m-033-7.png,
m-036-1.png,␣m-036-14.png,␣m-036-16.png,␣m-036-3.png,
␣m-036-5.png,␣m-036-6.png,␣m-036-7.png,
m-037-1.png,␣m-037-14.png,␣m-037-16.png,␣m-037-3.png,
␣m-037-5.png,␣m-037-6.png,␣m-037-7.png,
m-038-1.png,␣m-038-14.png,␣m-038-16.png,␣m-038-3.png,
␣m-038-5.png,␣m-038-6.png,␣m-038-7.png,
m-039-1.png,␣m-039-14.png,␣m-039-16.png,␣m-039-3.png,
␣m-039-5.png,␣m-039-6.png,␣m-039-7.png,
m-040-1.png,␣m-040-14.png,␣m-040-16.png,␣m-040-3.png,
␣m-040-5.png,␣m-040-6.png,␣m-040-7.png,
m-075-1.png,␣m-075-14.png,␣m-075-16.png,␣m-075-3.png,
␣m-075-5.png,␣m-075-6.png,␣m-075-7.png,
m-076-1.png,␣m-076-14.png,␣m-076-16.png,␣m-076-3.png,
␣m-076-5.png,␣m-076-6.png,␣m-076-7.png,
w-020-1.png,␣w-020-14.png,␣w-020-16.png,␣w-020-3.png,
␣w-020-5.png,␣w-020-6.png,␣w-020-7.png,
w-023-1.png,␣w-023-14.png,␣w-023-16.png,␣w-023-3.png,
␣w-023-5.png,␣w-023-6.png,␣w-023-7.png,
w-024-1.png,␣w-024-14.png,␣w-024-16.png,␣w-024-3.png,
␣w-024-5.png,␣w-024-6.png,␣w-024-7.png,
w-025-1.png,␣w-025-14.png,␣w-025-16.png,␣w-025-3.png,
␣w-025-5.png,␣w-025-6.png,␣w-025-7.png,

w-026-1.png,␣w-026-14.png,␣w-026-16.png,␣w-026-3.png,
␣w-026-5.png,␣w-026-6.png,␣w-026-7.png,
w-028-1.png,␣w-028-14.png,␣w-028-16.png,␣w-028-3.png,
␣w-028-5.png,␣w-028-6.png,␣w-028-7.png,
w-057-1.png,␣w-057-14.png,␣w-057-16.png,␣w-057-3.png,
␣w-057-5.png,␣w-057-6.png,␣w-057-7.png,
w-058-1.png,␣w-058-14.png,␣w-058-16.png,␣w-058-3.png,
␣w-058-5.png,␣w-058-6.png,␣w-058-7.png,
w-059-1.png,␣w-059-14.png,␣w-059-16.png,␣w-059-3.png,
␣w-059-5.png,␣w-059-6.png,␣w-059-7.png,
w-060-1.png,␣w-060-14.png,␣w-060-16.png,␣w-060-3.png,
␣w-060-5.png,␣w-060-6.png,␣w-060-7.png

References

[Abi17] Aysajan Abidin. On privacy-preserving biometric authentication. In Kefei Chen, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology*, pages 169–186, Cham, 2017. Springer International Publishing.

[ABS15] Wilson Abel Alberto Torres, Nandita Bhattacharjee, and Balasubramaniam Srinivasan. Privacy-preserving biometrics authentication systems using fully homomorphic encryption. *International Journal of Pervasive Computing and Communications*, 11(2):151 – 168, 2015.

[Adl04] Andy Adler. Images can be regenerated from quantized biometric match score data. In *In Proceedings Canadian Conference on Electrical and Computer Engineering*, pages 469–472, 2004.

[Adl05] Andy Adler. Vulnerabilities in biometric encryption systems. In Takeo Kanade, Anil Jain, and Nalini K. Ratha, editors, *Audio- and Video-Based Biometric Person Authentication*, pages 1100–1109, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Adl08] Andy Adler. Biometric system security. In Anil K. Jain, Patrick Flynn, and Arun A. Ross, editors, *Handbook of Biometrics*, pages 381–402. Springer US, Boston, MA, 2008.

[AMU97] Y Adini, Y Moses, and S Ullman. Face recognition: the problem of compensating for changes in illumination direction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):721–732, 1997.

[BBD+10] J R Beveridge, D S Bolme, B A Draper, G H Givens, Yui Man Lui, and P J Phillips. Quantifying how lighting and focus affect face recognition performance. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 74–81. IEEE, 2010.

[BDHV07a] Ileana Buhan, J. Doumen, Pieter Hartel, and Raymond Veldhuis. Constructing practical fuzzy extractors using qim. 01 2007.

[BDHV07b] Ileana Buhan, Jeroen Doumen, Pieter Hartel, and Raymond Veldhuis. Fuzzy extractors for continuous distributions. In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS), Singapore*, pages 353–355. ACM, 2007.

[BDHV07c] I.R. Buhan, J.M. Doumen, Pieter H. Hartel, and Raymond N.J. Veldhuis. *Constructing practical Fuzzy Extractors using QIM.* Number LNCS4549/TR-CTIT-07-52 in CTIT Technical Report Series. Distributed and Embedded Security (DIES), 7 2007.

[BHK97] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.

[BM01] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 245–250, New York, NY, USA, 2001. ACM.

[Bow02] S.T. Bow. *Pattern Recognition and Image Preprocessing.* Signal Processing and Communications. CRC Press, 2002.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[Cam94] AT& T Laboratories Cambridge. The database of faces, 1994. Formerly: The ORL Database of Faces.

[CBOB14] A. Chouchane, M. Belahcene, A. Ouamane, and S. Bourennane. 3d face recognition based on histograms of local descriptors. In *2014 4th International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–5, 2014.

[CS98] J. Conway and N.J.A. Sloane. *Sphere Packings, Lattices and Groups.* Grundlehren der mathematischen Wissenschaften. Springer New York, 1998.

[CSC08] Ann Cavoukian, Alex Stoianov, and Fred Carter. Keynote paper: Biometric encryption: Technology for strong authentication, security and privacy. In Elisabeth de Leeuw, Simone Fischer-Hübner, Jimmy Tseng, and John Borking, editors, *Policies and Research in Identity Management*, pages 57–77, Boston, MA, 2008. Springer US.

[DFM98] George Davida, Yair Frankel, and Brian Matt. On enabling secure applications through off-line biometric identification. pages 148–157, 06 1998.

[DKM+07] S. C. Draper, A. Khisti, E. Martinian, A. Vetro, and J. S. Yedidia. Using distributed source coding to secure fingerprint biometrics. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 2, pages II–129–II–132, April 2007.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, March 2008.

[Dro15] Giulia Droandi. Non-interactive privacy preserving protocol for biometric recognition based on somewhat homomorphic encryption. volume 2015, 07 2015.

[Dun09] Ted. Dunstone. *Biometric System and Data Analysis Design, Evaluation, and Data Mining*. Springer US, New York, NY, 1st ed. 2009. edition, 2009.

[Gal63] R. G. Gallager. Low-density parity-check codes. Cambridge, MA: M.I.T. Press, 1963.

[GBN05] Navin Goel, George Bebis, and Ara Nefian. Face recognition experiments with random projection. *Proceedings of SPIE - The International Society for Optical Engineering*, 5776, 03 2005.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 169–178. ACM, New York, 2009.

[HA16] Parrao Hernandez and Ricardo Antonio. *Construction and Decoding of Low Density Lattice Codes*. PhD thesis, Japan Advanced Institute of Science and Technology, 2016.

[HH16] Changhee Hahn and Junbeom Hur. Efficient and privacy-preserving biometric identification in cloud. *ICT Express*, 2(3):135 – 139, 2016. Special Issue on ICT Convergence in the Internet of Things (IoT).

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[JL84] William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.

[JS06] Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Designs, Codes and Cryptography*, 38(2):237–257, Feb 2006.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, CCS '99, pages 28–36, New York, NY, USA, 1999. ACM.

[KD08] Brian Kurkoski and Justin Dauwels. Message-passing decoding of lattices using gaussian mixtures. In *Computing Research Repository - CORR*, pages 2489 – 2493, 08 2008.

[KD10] Brian Kurkoski and Justin Dauwels. Reduced-memory decoding of low-density lattice codes. *Communications Letters, IEEE*, 14:659 – 661, 08 2010.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[Lar06] D.T. Larose. *Data Mining Methods and Models*. Wiley, 2006.

[LBD+09] Yui Man Lui, D Bolme, B.A Draper, J.R Beveridge, G Givens, and P.J Phillips. A meta-analysis of face recognition covariates. In *2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–8. IEEE, 2009.

[LHK05] Kuang-Chih Lee, J Ho, and D.J Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, 2005.

[LHV+18] Shuiyin Liu, Yi Hong, Emanuele Viterbo, Alessia Marelli, and Rino Micheloni. Fast decoding of low density lattice codes, 06 2018.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60:1–23, 05 2010.

[LT03] Jean-Paul Linnartz and Pim Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In Josef Kittler and Mark S. Nixon, editors, *Audio- and Video-Based Biometric Person Authentication*, pages 393–402, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[MB98] A.M. Martinez and R. Benavente. The ar face database. CVC Tech. Report # 24, 1998.

[MMJP03] D. Maltoni, D. Maio, A.K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer Professional Computing. Springer New York, 2003.

[Moo05] T.K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley, 2005.

[MRW99] Fabian Monrose, Michael K. Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, CCS 99, page 7382, New York, NY, USA, 1999. Association for Computing Machinery.

[MTP11] Evangelia Micheli-Tzanakou and Konstantinos N. Plataniotis. *Biometrics: Terms and Definitions*, pages 142–147. Springer US, Boston, MA, 2011.

[Oli06] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006.

[PM17] Elena Pagnin and Aikaterini Mitrokotsa. Privacy-preserving biometric authentication: Challenges and directions. *Security and Communication Networks*, 2017:9, 2017.

[Pol94] G. Poltyrev. On coding without restrictions for the awgn channel. *IEEE Transactions on Information Theory*, 40(2):409–417, March 1994.

[PvdG16] Vladimir P. Parente and Jeroen van de Graaf. A practical fuzzy extractor for continuous features. In Anderson C.A. Nascimento and Paulo Barreto, editors, *Information Theoretic Security*, pages 241–258, Cham, 2016. Springer International Publishing.

[QYX13] J. Qian, J. Yang, and Y. Xu. Local structure-based image decomposition for feature extraction with applications to face recognition. *IEEE Transactions on Image Processing*, 22(9):3591–3603, Sep. 2013.

[RCCB07] Nalini K. Ratha, Sharat Chikkerur, Jonathan H. Connell, and Ruud M. Bolle. Generating cancelable fingerprint templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(4):561–572, April 2007.

[Reg06] Oded Regev. Lattice-based cryptography. In *Advances in cryptology—CRYPTO 2006*, volume 4117 of *Lecture Notes in Comput. Sci.*, pages 131–141. Springer, Berlin, 2006.

[RJY15] J. Ren, X. Jiang, and J. Yuan. A chi-squared-transformed subspace of lbp histogram for visual recognition. *IEEE Transactions on Image Processing*, 24(6):1893–1904, June 2015.

[RW05] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[SCS11] Fabio Scotti, Stelvio Cimato, and Roberto Sassi. Biometric privacy. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 101–104. Springer US, Boston, MA, 2011.

[Set06] Ishwar K Sethi. Biometrics. In *Privacy and Technologies of Identity: A Cross-Disciplinary Conversation*, pages 117–134. Springer US, Boston, MA, 2006.

[SFS08] Naftali Sommer, Meir Feder, and Ofir Shalvi. Low-density lattice codes. *Information Theory, IEEE Transactions on*, 54:1561 – 1585, 05 2008.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[SLM09] Yagiz Sutcu, Qiming Li, and Nasir Memon. Design and analysis of fuzzy extractors for faces. *Proceedings of SPIE - The International Society for Optical Engineering*, 7306, 05 2009.

[SLWT15] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. Deepid3: Face recognition with very deep neural networks. *CoRR*, abs/1502.00873, 2015.

[SRS+98] Colin Soutar, Danny Roberge, Alex Stoianov, Rene Gilroy, and Bhagavatula Vijaya Kumar. Biometric Encryption using image processing. In Rudolf L. van Renesse, editor, *Proceedings of the SPIE, Volume 3314, p. 178-188 (1998).*, volume 3314 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 178–188. 1998.

[SWY+06] Shiguang Shan, Wenchao Zhang, Yu Su, Xilin Chen, and Wen Gao. Ensemble of piecewise fda based on spatial histograms of local (gabor) binary patterns for face recognition. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 606–609, Aug 2006.

[TN05] Andrew Teoh and David Ngo. Cancellable biometrics featuring with tokenised random number. *Pattern Recognition Letters*, 26:1454–1460, 07 2005.

[TNG04] Andrew Teoh, David Ngo, and Alwyn Goh. Personalised cryptographic key generation based on facehashing. *Computers & Security*, 23:606–614, 10 2004.

[TP91] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, January 1991.

[TY07] A. B. J. Teoh and C. T. Yuang. Cancelable biometrics realization with multispace random projections. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(5):1096–1106, Oct 2007.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[VTO+10] E. A. Verbitskiy, P. Tuyls, C. Obi, B. Schoenmakers, and B. Skoric. Key extraction from general nondiscrete signals. *IEEE Transactions on Information Forensics and Security*, 5(2):269–279, 2010.

[WHD18] H. Wang, J. Hu, and W. Deng. Face feature extraction: A complete review. *IEEE Access*, 6:6001–6039, 2018.

[WP10] Y. Wang and K. N. Plataniotis. An analysis of random projection for changeable and privacy-preserving biometric verification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(5):1280–1293, Oct 2010.

[YA10] Richard Youmaran and Andy Adler. *Measuring Information Content in Biometric Features*, pages 579–597. IEEE, 2010.

[YF09] Yair Yona and Meir Feder. Efficient parametric decoder of low density lattice codes. *IEEE*, pages 744 – 748, 08 2009.

[YyY02] Jian Yang and Jing yu Yang. From image vector to matrix: a straightforward image projection techniqueimpca vs. pca. *Pattern Recognition*, 35(9):1997 – 1999, 2002.

[ZKC+98] Wenyi Zhao, Arvindh Krishnaswamy, Rama Chellappa, Daniel L. Swets, and John Weng. *Discriminant Analysis of Principal Components for Face Recognition*, pages 73–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.