Testing Fuzzy Extractors for Face Biometrics: Generating Deep Datasets

by

Alain Alimou Tambay

Thesis submitted to the University of Ottawa
In partial fulfillment of the requirements
For the M.C.S. degree in Computer Science

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

© Alain Alimou Tambay, Ottawa, Canada, 2020

Abstract

Biometrics can provide alternative methods for security than conventional authentication methods. There has been much research done in the field of biometrics, and efforts have been made to make them more easily usable in practice. The initial application for our work is a proof of concept for a system that would expedite some low-risk travellers' arrival into the country while preserving the user's privacy. This thesis focuses on the subset of problems related to the generation of cryptographic keys from noisy data, biometrics in our case.

This thesis was built in two parts. In the first, we implemented a key generating quantization-based fuzzy extractor scheme for facial feature biometrics based on the work by Dodis et al. and Sutcu, Li, and Memon. This scheme was modified to increased user privacy, address some implementation-based issues, and add testing-driven changes to tailor it towards its expected real-world usage. We show that our implementation does not significantly affect the scheme's performance, while providing additional protection against malicious actors that may gain access to the information stored on a server where biometric information is stored.

The second part consists of the creation of a process to automate the generation of deep datasets suitable for the testing of similar schemes. The process led to the creation of a larger dataset than those available for free online for minimal work, and showed that these datasets can be further expanded with only little additional effort. This larger dataset allowed for the creation of more representative recognition challenges. We were able to show that our implementation performed similarly to other non-commercial schemes. Further refinement will be necessary if this is to be compared to commercial applications.

Acknowledgements

I would like to thank Dr. Carlisle Adams for his supervision and support throughout my Masters, as well as my colleagues. I would also like to thank Dr. David Barrera and Dr. David Knox for their feedback and being on my defense committee. And a final thank you to CBSA for bringing us this project.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Biometrics

Biometric information is derived from a person's biological attributes. While well-known examples include fingerprints, eye scans, or voice samples, such information is not limited to the physical body alone: an individual's gait, or the length of pauses between keystrokes also provide valid biometrics. In recent years, the biometrics field has grown in popularity and their use for access to electronic systems has become commonplace. While such systems are indeed impressive, it would be difficult to understate how much research has been done on the use of new types of biometrics, increasing accuracy, and protecting individuals from misuse of their biometric information.

Biometrics can provide more security than conventional authentication methods [JRP04]; they can't be lost, lent, forgotten, or physically stolen. In most cases, biometrics are nearly ubiquitous and so provide an easily available method of identification. In addition, there is a great deal of possible varia-

tion between different biometric samples, more so than during conventional password use. And indeed, a large variety of biometric identifiers can be derived from a person. However, few of these identifiers are ever used to replace passwords for strict authentication purposes, due to a variety of difficulties with their usage.

Unlike passwords or passphrases, biometrics are difficult to reproduce exactly. People can easily replicate a word they have memorized, but even if one is an extremely gifted artist, replicating a prior photograph is nearly impossible. Similarly, two sets of fingerprints taken even consecutively will always differ if one examines them with enough precision. This may stem from slight differences in the position of the finger, to changes in the pressure applied by the individual on the scanner, to smudges on the image capturing surface, to even imperfections in the measuring instrument, among many other reasons. As such, there must be a way to account for the differences between samples stemming from the same source. These random and external differences are referred to as *noise*.

In addition, an individual's biometric information is directly tied to the person. Thus, there are many possible negative consequences to losing access control of biometric information: individuals do not want to be cross-matched between databases; they do not want to be tracked; they do not want people to steal or re-use their biometric information; and they don't want corporations or other entities to extract more information about them. Because of this interconnectedness, biometrics can't simply be reset or changed if or when their owner chooses; a compromised biometric is lost forever. People may therefore have reservations regarding what they feel public or private institutions should be allowed to do with their information, and are wary of laws relating to information security. Such issues curtail the adoption of widespread biometric authentication.

1.1.1 Biometric Authentication

It is important to note differences between the ways biometrics can be used for the recognition of persons. Biometric *identification* is the use of biometric samples to discern to which individual, from many, is the source of the sample. An example of such system would include an electronic home assistant which can recognize a household member by their voice. Biometric *authentication* is the use of biometric information to validate whether a person is indeed who they claim to be, by comparing biometric sample to information provided previously. Figure 1.1 illustrates a rudimentary description of a biometric authentication scheme:

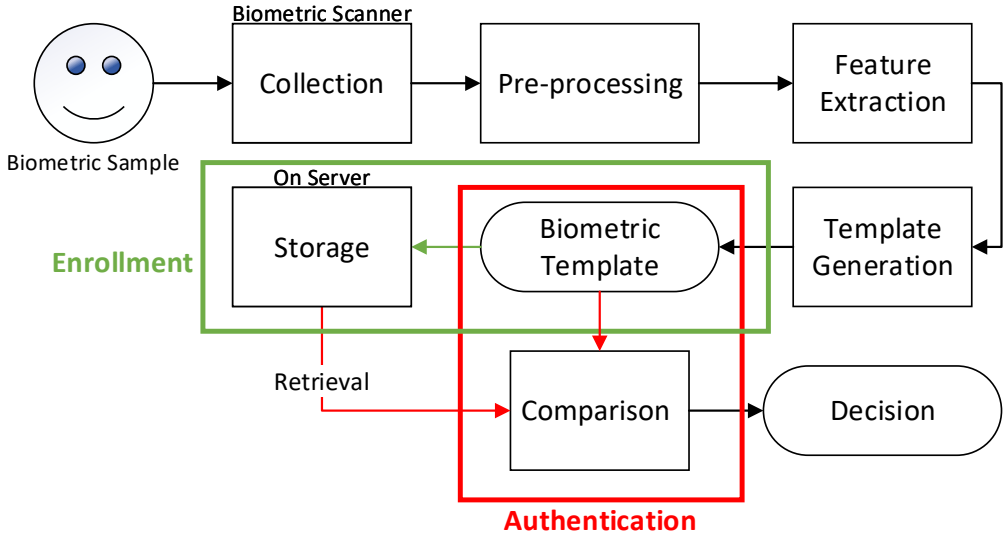


Figure 1.1: Rudimentary authentication scheme flowchart

Part 1: User Enrollment

1. **Collection:** The biometric information is first collected through a biometric scanner. These may be specialized, such as a fingerprint reader or a special keyboard, or they may be more generic, like a camera or a microphone.

2. **Preprocessing:** These "raw" biometric samples are taken through a series of *preprocessing* steps to reduce the presence of impertinent measured information, such as the lighting conditions, or the orientation of the finger on the fingerprint scanner. This serves to *denoise* and *normalize* the biometric sample. At this point, a fingerprint biometric would still resemble an image.

3. **Feature Extraction:** The processed biometric information is transformed into a more information-dense representation. This is done to conserve memory and make the biometric data easier to manipulate. Note that while this is not a lossless transformation; it is usually done in a way that the information that can most easily identify the user is retained. The transformed information is referred to as the *biometric features*. These could be the position of the ridge endings on a fingerprint, or the points where these ridges split in two. In the case of fingerprints, these are two types of fingerprint *minutiae*.

4. **Template Generation:** The biometric features may be manipulated further and are then converted into a format tailored to the application in which it will be used. This representation of the biometric information is the *biometric template*. A biometric template may consist of the set of points listing the minutiae on a finger and their position relative to one another.

5. **Storage:** The biometric template may then be stored on a server with a linked identity until needed for authentication.

Part 2: User Authentication

First, the user claims an identity by providing a name, a username, a previously received token, a unique ID, etc. The system can then attempt to authenticate the user. The first few steps of biometric authentication are similar to those for enrollment. Steps 1 through 4 are as above in *Part 1: User Enrollment* with the user providing a new biometric sample, this samples is preprocessed, the biometric features are extracted, and a new template is constructed. Then:

5. **Template Retrieval:** The system retrieves the enrollment template on the server belonging to the claimed identity.
6. **Comparison:** The retrieved enrollment template and the newly generated authentication template are compared using a *similarity metric*. This could entail creating a correspondence graph between the features in each template and calculating the hamming distance between those components.
7. **Decision:** Based on the similarity between the templates and the authentication *threshold* set, the user's authentication is either validated or rejected.

In this example, the existence of a version of the biometric information stored on a server gives cause for concern. If the database is hacked, or if a public or private institution improperly disseminates the information or uses it for

nefarious purposes, then the users may be at risk. Lost biometric information can jeopardize the privacy of users, including (but not limited to) users who are in witness protection programs, with clear links to their prior identity, or by groups trying to identify minorities at scale. Such problems of biometric privacy protection are a growing concern.

1.2 Project Background

We were approached by a branch of the Canadian government working on a system to expedite the processing of travellers arriving to the country. The system would use multiple identifiers to link travellers, including biometrics, electronic passports, registered accounts, and their mobile phone.

The system would work as follows:

- Trusted airport travellers would enroll on the mobile application before departure, which would require providing some personal information, including a biometric sample in the form of pictures of their face taken by their phone - a selfie.
- When departing, the system at the Canadian airport would be notified of the traveller's expected arrival, and the preliminary steps of the arrival process may be initiated.
- Travellers would use the mobile application to fill out and their travel documentation, and could submit the documentation electronically once landed.
- At the airport, low risk travellers would be prompted to use their mobile phones or on-site kiosks for the expedited process.

- The traveller would provide new biometric samples which would be evaluated for liveness and matched to the passport image before authentication.
- If successfully authenticated, travellers can swiftly and efficiently cross the border.
- In case of anomalies or failure during these steps, travellers can be asked to follow the conventional arrival process.

1.2.1 Preserving Biometric Privacy

For this project, we were to provide a modified biometric authentication component to the system as a proof of concept for further development. The modified component would serve to protect the travellers' biometric information. We believe that there are many benefits: increased trust from passengers; increased security for users, their data, and the provider. Specifically, we would be exploring the use of an alternate authentication scheme to preserve the users' privacy. Historically, this has been done in two ways.

The first method uses *Feature Transformation* schemes, which modify the biometric template stored in databases, and thereby minimize risk to the user. By using a unique irreversible transformation on the biometric template, compromised templates can't be linked to their pre-transformed versions, nor to other transformed templates belonging to the same user. Feature Transformation schemes solve another issue with the limited nature of biometrics, such as having a fixed number of fingers, or the inability to spontaneously alter one's gait. Compromised templates can simply be revoked and new templates with a different transformation can be issued. As such, the user can reset their template, just like they could with a password.

The second category of biometric privacy protection schemes fall under the umbrella of *Biometric Cryptosystem* schemes. These involve the use of a more traditional cryptographic key, linked to an underlying biometric in a particular way. This allows for the use of biometrics in any situation where a cryptographic key could be used. A biometric can then be used as a password without a fundamental change to the underlying algorithm, so long as the biometric is also used to get the key first.

This latter category of biometric cryptosystems can further be split into two subsets. The first subset are *key binding* schemes, such as the fuzzy commitment [JS06] and fuzzy vault schemes [JW99] (both of these are described in more detail below). In key binding schemes, the cryptographic key is generated and encoded in such a way that the biometric is required for its retrieval. In such schemes, the database only needs to store the key, which is not linked to the biometric, and therefore does not provide information about the user to anyone, even with access to the key.

The second type of biometric cryptosystems are the *key generating* ones. With these schemes, the biometric is used as an outside source of entropy to generate the cryptographic key, itself. As biometrics are inherently noisy, new measurements are seldom identical, despite their similarity to previous readings. Therefore, the ability to tolerate, or correct, some error is required. The *Fuzzy Extractors* and *Secure Sketches* primitives are used to generate a reproducible, uniformly random string from the noisy non-uniform input. These two-part error correcting primitives were introduced by Dodis et al. in [DRS04] and serve as the privacy preserving component to our implementation.

1.2.2 Dataset Restrictions

Due to the context of the project, the biometric sample images provided to the authentication system would have the following characteristics:

1. The images are taken from mobile phones or kiosks, which both limits and increases variability of quality.
2. The images would be similar in pose to passport images, to allow prior matching to the passport image.
3. Would be in a non-controlled setting (e.g. outside, inside the aircraft, in the airport), which leads to variation in many environmental factors.
4. As a workaround for issues found in later testing, the images would be taken as the user spoke while facing the camera.

As biometric authentication systems are usually tested using datasets of collected biometric samples, datasets providing samples allowing for representative testing of expected usage would be limited. Furthermore, the fuzzy extractor scheme requires multiple samples for enrollment, further restricting the available datasets. We found the pool of datasets matching our requirements to be small.

The datasets used to test fuzzy extractor schemes in practice were generated in a tightly-controlled laboratory settings. Such datasets often strictly control several factors such as lighting, background, camera quality, and facial expression. This is markedly different from the images used in the government system, which will feature images taken by individuals in various environments by phone or kiosk. It was important to emulate these variable conditions during testing. As such, a significant part of this project also

consisted of developing a method to test the system in a way that emulated expected usage.

1.3 Work Done

Project Roadmap

The initial roadmap was as follows:

1. Implement an initial biometric-based fuzzy extractor authentication scheme.
2. Validate its use in a practical context. We were initially unsure that the scheme could be used with a mobile phone component, for reasons relating to storage, processing performance, and data transmission requirements. Note that this was successful and is not part of the thesis.
3. Test the addition of fuzzy extractor schemes as a wrapper to an existing feature extractor system, for both authentication accuracy and runtime efficiency.
4. Provide an implementation that fulfilled the requirements for integration with the service being developed by the government agency. This includes the development of many additional components such as a basic application programming interface (API). This was also successful, and is not part of the thesis.
5. Link our scheme to the greater travel system, including integration with the existing travel app, and access to the real data. This would allow the final testing of the proof of concept.

We found the existing biometric datasets unable to meet our testing requirements. At integration, as the project was in its early phases, the amount of real data available for our testing was limited. This necessitated the following additional steps:

6. Create a method by which we could generate data that resembled the expected data more closely than that available publicly.
7. Test the system using data emulating real world data.

Research Questions

This project provided the opportunity to explore the following research questions:

- Can fuzzy extractor based schemes be used in a real world contexts?
- How can a deep dataset of face images set to emulate semi-constrained contexts be generated with minimal effort?

Contributions

To this end, we:

- Modified a fuzzy extractor based biometric authentication scheme to simplify the codebook representation; and to allow the separation of the global codebook generation and user enrollment.
- Determined the available deep datasets of face biometrics.

- Created a process that generates datasets from YouTube videos.
- Tested the implemented scheme on existing and generated datasets.

1.4 Thesis Outline

In Chapter 2, we outline first the related work in the areas of privacy preserving biometric validation and error correcting code based commitment schemes. We then describe the existing datasets of face images and their ability to fulfill our requirements for a testing set.

In Chapter 3, we explain the fuzzy extractor schemes in detail, with emphasis on the construction presented by Sutcu et al. [SLM07, SLM09]. We then detail how we fixed the codeword alignment, improved the codebook representation, and added a scaling parameter as in [BTA]. Finally, we describe the details relating to the implementation and the testing.

In Chapter 4, we offer details on the creation of our own dataset from online resources. We discuss why the laboratory-created datasets did not match our real-world use case, and why it was necessary to create a new dataset. Our goal here was to create a process that can generate a large deep dataset, with subject identification and with minimal human involvement and effort.

In Chapter 5, we present the results of the dataset creation as well as the results of the testing with the algorithm. Here we are looking to understand if our changes to the implementation may have had unintended trade-offs. We compare our results to other research that have tested the available datasets in similar contexts. We then compare the performance of our algorithm on public datasets and on our generated one to determine if our implemented

algorithm could function in a real-world application that matches our use case.

In Chapter 6, we state our conclusions, based on the analysis of Chapter 5. Further research questions are presented and we explain the next steps that would be necessary to make the application of the algorithm feasible at a commercial level.

Chapter 2

Related Work

2.1 Existing Privacy Protection Schemes

2.1.1 Early Work

Davida et al. [DFM98] state that an increase in the number of applications requiring identification through physical user characteristics has led to a growing need for secure and accurate identification systems. They proposed in 1998 a series of schemes that allow for biometric authentication without compromising the security of the biometric template. In addition, their schemes allow for the correction of some of the variation in biometrics stemming from the fact that humans change over time and that they rarely mimic their past actions perfectly. To do so, they first collect multiple sets of biometric information at enrollment time and create a biometric template using a majority decoder. Then using an error correcting code they generate check digits which are stored with a hash of the template. At authentica-

tion time, a new set of the user's biometric information is captured, passed through the majority decoder and the output is then corrected using the stored check digits. The output is then hashed and compared to the stored value for authentication. Since the biometric templates themselves are not stored, there is little risk to the user's privacy though the leakage of the biometric template. Despite describing how the system would work with iris scans, [JW99] and [JS06] state that the schemes lack the robustness for real-world use. The error correcting capabilities are insufficient for an amount of noise common to most biometrics. Also, while they assume that subsequent readings of biometric information would be independent, noise stemming from how an individual might misuse a scanner (like incorrect placement) or introduced at the sensor (such as dust on a lens) will often affect multiple readings. However, this paper pioneered the use of error correcting codes for biometric template security and Davida et al.'s work has influenced much of the privacy preserving biometric field.

Following the work in [DFM98] by Davida et al., [JW99] by Juels and Wattenberg in 1999 intended to provide a more practical error tolerant scheme for biometric authentication. Their fuzzy commitment scheme allows the sending of a string in a secure manner. This scheme can be used for authentication in situations where an authorization officer acts as the sender and a biometric scanner as the receiver. In this case, the enrollment biometrics is used to encode a message that is transmitted. If the message is decoded, this must mean that the authentication biometric is close to the enrollment one, and the person is authenticated. Here, the fuzziness stems from using error correction to recreate a message exactly, when receiving a string close enough to the original one that was committed. Their construction requires the existence of a set of pre-generated codewords that contain built in redundancy to allow for their later reconstruction, which are the strings that

can be sent. To commit a particular string, one would find the codeword nearest to the message by hamming distance. The codeword is then XORed with the message to calculate the difference, and the difference is then stored along with a hash of the codeword. The receiver would then scan a new biometric, map it to the nearest codeword in the set, and then XOR the codeword with the stored difference. If the new biometric was close enough to the original, it would map to the same codeword, and the XORing the new codeword and the difference would grant the original message. Like in [DFM98], the message is not kept, and only the set of public codewords, the difference, and a hash of the codeword are stored. Similarly to [DFM98], this paper influenced many that came after. It is the first to coin the term *fuzzy* to describe error tolerant codes for biometrics. [JW99] does succeed in its goal to be a more robust scheme as it allows for more error-tolerance with provable security guarantees [JS06]. However, [JW99]'s scheme requires a uniform distribution of messages, which does limit its usage.

Juels and Sudan in 2002 published *A Fuzzy Vault Scheme* [JS06]. It follows the same concepts as [DFM98] and [JW99] but does so in a novel way. Juels and Sudan introduce the idea of a *fuzzy vault*, in which a sender can store information that can only be retrieved by a receiver with a password close enough to the sender's. Unlike previous papers, they use set difference rather than hamming distance to the compare passwords, which are represented by sets of bit strings. To send a message m, they first generate a polynomial p that contains m, such as by having m embedded in the coefficients. Then for each value a in the set A that represents the password, they compute the points $(a, p(a))$. They then generate a large number of chaff points not on the polynomial p. These two groups of points make up the *vault* and the message is now committed. To extract the message from the vault, a new password set B of the same size as A is required. For each b in B, if a

point (b, y), no matter the value of y, exists in the vault, then the point is assumed to be $(b, p(b))$. If B is similar enough to A, there is enough overlap for the receiver to re-derive the polynomial from the set of points $(b, p(b))$. Of note, this means that the order of appearance of the elements in the sets is irrelevant. This scheme can be adapted for biometric authentication if the password set A is the biometric generated at enrollment time and the set B is the biometric template captured at authentication time. This scheme is more memory intensive than [DFM98] and [JW99] and probabilistic due to the random chaff points added. [DRS04] points out that while [JS06] seems to be similar to their construction, it leaks information about the secret despite providing a large number of "provably valid" possible values. The scheme also requires a significant amount of memory and computation, which is lowered in subsequent schemes explored in this thesis.

Ratha et al.'s paper [RCB01] is widely regarded as one of the first to introduce the idea of cancelable biometrics. Building on the idea that biometrics have significant advantages over conventional passwords for identification purposes, the authors state that one of biometrics' biggest strength, their propensity towards staying mostly stable over long periods of time, was also one of their greatest weaknesses preventing their widespread adoption. Their temporal stability and their relatively large entropy allows for a secure password replacement. However, if they happen to be compromised, their "tied to owner" nature results in a host of problems. Particularly, a biometric that is leaked not only can't be reused, but is also costly to replace. Humans have a limited set of biometric information (e.g. ten fingers, one face) and so new biometrics can't be generated repeatedly at no cost. Ratha's solution involved using a one way transformation on the biometric. The reusable transformation would be applied either on the original biometric before template creation or on the template, and would be uniquely tied to a

specific enrollment. At authentication time, the same transformation would be applied to the biometric before comparison. If the transformed biometric was compromised (for example following and unauthorized access to the database) the current authentication credentials are revoked, and new credentials with a new transformation can be re-issued to the user. This allows for the re-use of a given biometric if its transformed form is compromised and also prevents leaks from invalidating other templates stemming from the same base biometric. This seems to have been the first non-invertible feature transformation proposal and provided the baseline for many other papers. However, Ratha et al.'s work requires transformations that are non-trivial to construct and that are unique for each user and so would not scale according to Sutcu et al. in [SSM05]. In addition, they state that other needs, such as a larger possibility for error correction and quantized outputs are required for a secure algorithm. Sutcu et al. instead built their own one-way function based cancelable biometric scheme using combinations of overlapping Gaussian functions and the use of a hash function on the output of said function to generate a cryptographic key. They go on to test their construction on the ORL face database [SH].

2.1.2 Secure Sketch and Fuzzy Extractor Based Schemes

Dodis et al.'s Work

The 2004 paper [DRS04] by Dodis et al. describes a novel way to generate cryptographic keys from noisy information using two new cryptographic primitives, the *secure sketch*and the *fuzzy extractor*. These both consist of a pair of functions used in sequence where the first generates an input for the second.

The first of the new primitives, the secure sketch, generates information allowing for the recreation of its original input, with its *Generate Sketch* (`GS`) and *Recover* (`Rec`) procedures. `GS`'s input is used to create a *sketch*, which is the string output of the procedure. The `Rec` procedure is provided with the *sketch* and a noisy version of `GS`'s original input. If `Rec`'s noisy version of the input is similar enough to the first input given to `GS`, then `Rec`'s output will be the original input. Otherwise, there is no guarantee about the output, which is dependent on the actual construction, but is likely to resemble a random possible input to `GS`. The *sketch* string does not provide meaningful information about the original input string and is said to be safely public. As such, if the input to the secure sketch is a biometric template, the existence and storage of the *sketch* does not violate the biometric privacy of the template's owner.

The other primitive, the fuzzy extractor is constructed from a secure sketch and a strong randomness extractor, such as a hash function. It generates a nearly uniformly random key from its initial input with its *Generate* (`Gen`) and *Reproduce* (`Rep`) procedures. `Gen` requires a source of entropy as input (such as a biometric template) and provides two outputs, a random string which can be used as a key, and a *helper data* string, which is provided to `Rep`. `Rep`'s inputs include the *helper data* string from `Gen` and a noisy version of `Gen`'s input. `Rep` will then reproduce the initial random *output* that `Gen` had created, assuming that the noisy input is similar enough to the original one. Otherwise, there is no guarantee on the output, which is dependent on the randomness extractor. If this is a hash function, the likelihood that `Rep` produces the same output is the hash function's collision probability. Again, the existence of the *helper data* does not provide any meaningful information about the input or output to these procedures, and *helper data*can be stored publicly.

The authors do not assume any particular notion of closeness between the two strings used as input to either of the functions for the secure sketch or the fuzzy extractor. Instead, they posit that the strings belong to a given metric space and that they are no more than a certain distance away in that space. In their paper, they state that the metric spaces will be finite and their distance function will only take integers. With this, they provide constructions for three such similarity metrics: hamming distance, set difference and edit distance. With the fuzzy extractor primitive, it is possible to generate a cryptographic key from biometric information, say for symmetric encryption, and then discard both the key and the biometric. The encrypted information can later be decrypted with a key reproduced from a new, similar biometric as long as the *helper data* is still available. In this case the *helper data* does not need to be kept private, and the new biometric can be captured from its owner at decryption time.

It is important to note that `Gen`'s random output string is not completely uniformly random given the existence of *helper data*. This is because there is some leakage of information given the existence of *helper data*, and an adversary *is* more likely to guess `Gen`'s output if they have access to *helper data* than without. Similarly, this happens with `GS`'s input and *sketch*. One can calculate the entropy of the key generated by the fuzzy extractor based on the existence of *helper data* and they show that they can make the entropy loss exponentially small.

Li, Sutcu, and Memon's Work (also see section 3.2.2)

In Li, et al's Secure Sketch for Biometric Templates [LSM06], the authors propose an improvement on the previous author's paper. Their work centers mostly around the assumption in Dodis's paper [DRS04] that the biometric

templates can be represented as integers in metric spaces. This paper finds three problems with this. Firstly, converting real numbers to discrete ones, often by rounding them to the nearest decimal fraction, but sometimes by separating the measurement axis into intervals as in [BDHV07], causes large loss in entropy compared to the original sample. Secondly, in some cases, it is impractical to represent the template in a metric space. The example they give is iris scans which, due to their large size are hard to use with error correcting codes. As such, it is more efficient to use a two level technique. Due to the change in similarity measure, this is no longer a metric space. Thirdly, in some cases it can be hard to represent the biometrics in a metric space. In fingerprint templates, the minutiae, which depicts certain points within patterns on the fingers, are the features. Fingerprint comparison can be done by comparing the distance between select minutiae. Due to errors introduced at the time of scanning, it is possible that multiple readings of the same unchanged finger will not have the same set of minutiae. Evaluating the similarity between two fingerprints involves evaluating the difference between the sets of minutiae and the Euclidean distance between them. This also does not fit with Dodis et al.'s fuzzy extractor representation.

The papers by Li, Sutcu, and Memon [LSM06, SLM07] propose a general method for the use of a secure sketch based algorithm in a continuous domain. This involves quantizing the biometric template into a discrete representation and the method should do so in a way that minimizes the entropy loss according to a new metric they present, which is to be more suited to evaluate it in this kind of situation. Also of note, they bring attention that, in certain biometric representations, noise can shift points a certain distance or cause points to be replaced with others. They name the first of these errors white noise and the second replacement noise. They also state that they are focused on white noise and that other error correcting techniques

can be used for replacement noise, such as a secure sketch based technique for set difference.

Additional Work

We used the above schemes as the baseline for the fuzzy extractor project. Since Dodis's original construction requires the use of discrete values, much work has been done to address the inability to use fuzzy extractors schemes with continuous biometrics. Buhan et al.'s work[BDHV07], which provided a method for quantization, was published in 2007. However, since it only provides a method of quantization, using Li, Sutcu and Memon's [LSM06, SLM07, SLM09] for the described schemes seemed preferable.

In addition to this, there has been much work involving fuzzy extractors, but most of it was tangential to our work. There has been the usage of alternate means of quantization that are more complex, such as [PvdG16]'s work with low density lattice codes. There have been many general protocols that do not specify a particular biometric, like in [ZZZ15, LNB+18, LGM+17], the latter of which implements and tests a scheme, but using randomly generated simulated data. Fuzzy extractors have been included in authentication systems for Internet of Things devices in [LNB+18, Das17, MS17].

There has been work on different biometrics, such as fingerprints [MY19, GE16], and iris scans [FAD06, lHH12]. A few implementations of schemes are tested with face biometrics, such as the face image and iris scan dual biometric authentication system proposed in [EM17], which is tested using the Essex Faces94 Database [Spa07]. We have, however, been unable to find any papers testing fuzzy extractor schemes on facial biometrics meant to emulate real world usage.

2.2 Evaluation of Available Datasets of Face Images

We categorize datasets based on the number of unique samples per subject, as in [SJ19]. Datasets with a large number of samples per subject are denoted as *Deep Datasets* and those with few samples are said to be *Shallow Datasets*. In cases where the number of images per person is variable, we used the mean number of images per subject. In addition, because we need these images as a set for enrollment, we need to be able to identify the person in the dataset. This requires either labels or for the subject's pictures to be grouped. *Secure sketch* based algorithms need a set of images for enrollment rather than a single biometric sample. We used 8 images for enrollment, and as such required datasets with at least 8 images per subject.

Dataset construction seems to have changed over time. Earlier datasets such as [SH, MB98, GBK97, Spa07] were populated by images taken for the dataset with live participants. In these, increasing the number of images is easily done by increasing the number of images per participant. Such datasets were often deep, but of smaller size, due to a limited number of subjects. Later, more datasets were created from images available online. This leads to the creation of much larger datasets, both in terms of number of images and number of subjects although the subjects are rarely identified. The constituent images are also more varied, as the dataset creator could not dictate how the images were taken. This can be useful for applications that require more "in the wild" images, but may also cause issues if the images are too abnormal. Many of these newer datasets therefore use various methods to select only images that fit certain criteria, or process images to fit certain criteria after the fact.

Dataset generation often involves a human verification component. The early datasets had people placing the subjects and physically taking the images while some of the later ones either had poor graduate students annotating and labelling the image manually or provided unlabelled images that were later edited when mistakes were found and shared with the creators. Large companies with private datasets such as *Google* [SKP15] or *Flickr* (see [KSSMB16]) can crowdsource the labelling by having people provide labels during normal application use. Our goal was to create a process that can generate a large deep dataset, with subject identification and with minimal human involvement and effort.

Here we describe some existing face image datasets that currently exist. Some of these were used by us at early stages in the project, before we created our own method of dataset generation, while others were considered but ultimately did not meet our requirements. These datasets are grouped according to their *depth*.

2.2.1 Deep Datasets

These tend to be older datasets, and were made of photos taken of people in lab settings. We used these during the early portions of the project, before live testing.

AT&T Database of Faces, 1994

The *AT&T Database of Faces*, formerly *The ORL Database of Faces* [SH], consists of ten images of each of the 40 subjects. The images are 92 by 112 pixels and are in 8 bit grayscale. It was used in [SLM09] to evaluate

the performance of their algorithm. We also used it at first to test our implementations, as it was easy to use, as it is quite consistent, there is almost no variation in lightning, slight variation in expression, the images are cropped around the face with little variation for the location of key features, such as the eyes or mouth. This dataset is good for evaluating algorithms, but the set of images very much have the rigidness of lab taken images . As such, when the project started involving more testing with images taken by real people, we looked for a new source of face images.

AR Face Database, 1998

The *AR Face Database* [MB98] by Martinez and Benavente consists of 126 subjects and contains over 4000 images. As with the other deep datasets, these images were laboratory generated. The images vary in terms of expression, illumination (center, left, right), and in inclusion of different accessories. These coloured images are 768 by 576 pixels, and feature a blank background. The images for each person were taken in two sessions that were at least two weeks apart. We created a subset of the dataset for our testing purposes. The subset consists of 5 of the 13 images per capture session for a total of 10 images per subject. The sides of the images were also cropped to more closely match passport picture dimensions. We excluded the images that differed significantly from the neutral image, such as those with wild facial expressions, or those that contained accessories. This left us with images with the labels 1, 3, 5, 6, 7, 14, 16, 18, 19, and 20.

Essex Faces94 Database, 1994

The *Essex Faces94* is the largest of the deep dataset we used, with 153 subjects and 20 images each. It is freely available online and has been used as a benchmarking tool for testing some face recognition algorithms. Notably, Sutcu uses it in [SLM07]. It consists of head shots closely cropped around the head, where at closest some of the hair is cropped out, and at furthest some of the upper shoulder is visible. The subjects are mostly undergraduate students in the 18 to 20 year range (113 males, 20 females) and 20 staff members (all male). The dataset contains glasses as the only accessories and each subject's images were taken in a single session. The position of the face is constant with little variation, but there is variation in the expressions. There is no variation in lighting and the photos were taken into front of a green background. There are also the *Essex Faces95* [Spa18] and *Essex Faces96* [Spa94] databases of images with a large number of images for additional subjects, but these introduce variation by changing the distance between the camera and lighting to the subject.

Yale Face Database, 2001

The series of Yale Face Databases, starting with [GBK97] in 1997 and culminating with the *Extended Yale Face Database B (B+)* [GBK01], was published in 2001. The first release, *Yale Face Database* contained 15 subjects, and the subsequent ones had only 10. Each database had multiple images per individual, 576 in the *B* database. This dataset has a low number of subjects, and most variation in between images stems from differences in lighting (64 variations + 1 ambient light one). This is the only deep dataset that we did not use, as the number of subjects was much too small.

2.2.2 Shallow Datasets

We did not use any of these. They tend to be much larger than the Deep Datasets above. We attempted to use these as is, or generate a deep "sub" dataset from some of these larger ones (notably the *Labeled Faces in the Wild* dataset [HMBLM08]), but neither approach yielded results. In these cases, we were unable to reconcile the differences between these datasets' sample images and the series of consecutive, forward-facing images the authentication system was to receive. These tend to be used for training and testing face detection algorithms as opposed to face recognition.

Labeled Faces in the Wild, 2008

Huang et al.'s *Labeled Faces in the Wild* dataset [HMBLM08] aimed to provide a large set of unconstrained and labelled face images. It contains 13,233 images of 5749 individuals. The dataset contains images that were detected by the Viola-Jones detector [VJ04] used on a database of images. The images were resized, cropped and padded to have the same size, 250 by 250 pixels, and false face detections and images containing individuals who could not be identified were removed manually. The "in the Wild" stems from the fact that these images were taken for other purposes and are therefore unconstrained. This leads to variation in many characteristics, such as lighting conditions, backgrounds, pose, expression, articles, image quality, colour saturation. The images may contain many faces, and the faces are not completely centered, but the center pixel is located within the region of the face of the individual the image is labelled as. We did not use the Labeled Faces in the Wild dataset.

The MegaFace Benchmark, 2016

The dataset provided in [KSSMB16], *The MegaFace Benchmark: 1 Million Faces for Recognition at Scale* is large, with 1,027,060 images, of over 690,000 people. These images come from Yahoo's Flickr photos [TSF+15] with 100 million Creative Commons photos. The goal of the creation of *The MegaFace Benchmark* is to provide challenges to test face recognition algorithms. The challenges include the identification and verification of *probe* individuals from among a set of *distractors*, which are individuals with only a single picture making up the MegaFace dataset. The individuals to be identified and recognized come from two *probe sets*: an 80 person subset of the FaceScrub dataset [NW14] and the FG-NET aging dataset [CL08]. To construct the dataset, they optimized their sampling for the number of unique identities, and they only kept faces larger than 50 by 50 pixels. They used the HeadHunter algorithm by Mathias et al. [MBPVG14] to detect images of faces from the original larger data source. The finished set, however, has no constraints on the pose, expression, lighting, or exposure, and does not contain multiple images per person. Because of this, it did not satisfy our criteria.

MS-Celeb-1M, 2016

Guo et al.'s *MS-Celeb-1M* dataset [GZH+16] contains images from Google's freebase data dump. It is intended to serve as a benchmark for facial recognition at large scale and is populated by images of celebrities. As such it is quite large and is split according to expected usage. They provide a "training" set consisting of 100 images from search engines for the most popular 100,000 celebrities, but these images are not strongly filtered for accuracy. As an example they show that the images for Steve Jobs include an image

which is split lengthwise and contains the photo of both Steve Jobs and Ashton Kutcher who plays him in a movie. The other *Labeled* set is around 30,000 manually labelled images from which they created two subsets of different difficulty. The *Random* set consists of a random image chosen from queried images available online, while the *Hard* set consists of the images from those 30 that are most different from any in the training set. Since there is no temporal link between the photos, this is simply an aggregation of labelled images. Since our use case consists of consecutive images taken in one enrollment session, this dataset did not seem suitable for our needs.

Diversity in Faces, 2019

The Diversity in Faces [MRFS19] dataset, was released in early 2019. It consists of nearly 1 million images. Interestingly, its method for extracting face images from pictures is quite similar to the method we used in our dataset generation. Their motivation was to provide a diverse dataset to be used for training machine learning models. They posit that increased diversity in available datasets should reduce the number of problems related to narrow datasets. Their dataset is constructed from the images in the YFCC-100M dataset, which stems from the Flickr image hosting service, as with [KSSMB16] above. It contains 10 facial coding schemes that identify information about the images, such as [Far94]'s Craniofacial Distances. While this dataset is impressive, it is shallow, and came out well after we were deep in the project, and as such we couldn't use it.

YouTube Based Datasets:

In addition, many datasets have been created from [YT]. One of these in particular contains pictures of people: the YouTube Faces Database [WHM11]. The dataset was created from the individuals included in the [HMBLM08] database. Their names were searched on YouTube, and the six highest results were used for image extraction. These videos were then scoured for series of consecutive frames with *stable detection*. They define these as sets of at least 48 consecutive frames containing Viola-Jones [VJ04] detected faces and where the inter-frame distance between faces was low. The videos were then manually verified for correct labelling, uniqueness, and to ensure there were no other errors. They release a fairly large dataset, within which the images are cropped around the faces. From an initial set of 18,899 videos of 3345 subjects, they were able to extract 3425 videos of 1595 subjects. They also use *Matched Background Similarity* to detect whether a face appearing in two different videos is of the same subject, based on the similarity of the video's background. While their research is interesting, and parts of it could be used to bolster our own, they also note that:

> Videos found in on-line repositories are very different in nature. Many of these videos are produced by amateurs, typically under poor lighting conditions, difficult poses, and are often corrupted by motion blur. In addition, bandwidth and storage limitations may result in compression artifacts, making video analysis even harder.

And unfortunately, the quality of the extracted images is too low for our use.

Notably, however, they use [HMBLM08] as a source of subjects to search. This could be an area of improvement for our created process, as our channel selection is manual. This brings attention to other (non facial image) datasets

generated from YouTube which we could possibly use as well. The YouTube Personality Dataset [BGP13] has audio and video based behavioural features from vloggers. Similarly, it would be possible to get channels from the *Trending YouTube Video Statistics* dataset [Jol] on Kaggle [Kag], and Google's *YouTube-8M* dataset [YT8] .

A summary of the datasets examined and their characteristics are listed in Table 2.1

Table 2.1: Datasets Details

	Images		**Image per Subject**		
Dataset	**"Passport"**	**Varied**	**In Sets**	**Identified**	**Linked**
AT&T	✓	✓	✓	✓	✓
AR	✓	✓	✓	✓	✓
Essex94	✓	✓	✓	✓	✓
Yale[1]	✓		✓	✓	✓
LFiW		✓			
MegaFace		✓		✓	
MS-Celeb[2]		✓		✓	
DiF		✓			
YT Faces[3]		✓	✓		✓

Notes:

1. The variation mostly stems from lighting.
2. No similarity between images in the set - these were aggregated from Google Images.
3. Low quality images.

Chapter 3

Scheme Implementation

3.1 System Process Flow

We first review the system process flow and identify the components for which we were responsible. This provides context to the rest of the chapter:

Travel Facilitation System — Pre-Travel and Enrollment

At the time when the traveller intends to travel, they would download the application, and sign up to the program. Their account is linked to their identity by providing their passport and travel information. At any time before their travel date, the traveller would enroll onto our biometric authentication component: (our responsibilities are in **bold**)

1. The traveller provides a series of face images as biometric samples, which is preprocessed and verified to ensure liveness.

2. These samples are compared to the traveller's passport photo to ensure the traveller matches their passport identity.

3. A token consisting of the personal and travel information is generated.

4. **The biometric samples would be used to generate a cryptographic key and helper data.**

5. The cryptographic key would be used to symmetrically encrypt the token, binding it to the traveller, the generated key is not saved.

6. The encrypted token is uploaded to a server.

7. The helper data would be stored on the traveller's mobile device.

8. The traveller, now enrolled, is free to travel to Canada.

Travel Facilitation System — Arrival and Authentication

On arrival, if deemed "low risk", the traveller would be prompted to follow the expedited procedure by their mobile application: (our responsibilities are in **bold**)

1. The traveller provides a new facial image biometric sample, which is preprocessed and verified to ensure liveness.

2. The encrypted token is downloaded from the server.

3. **The cryptographic key is regenerated using the new biometric sample and the helper data.**

4. The recreated key is used to decrypt the encrypted token.

5. If the token is successfully decrypted, then the authentication is successful, and the traveller proceeds to Canada.

6. If the decryption fails, then the key was not successfully regenerated, the authentication is unsuccessful, and the traveller reverts to the regular arrival procedures

The user's experience with the system is represented in Figure 3.1.

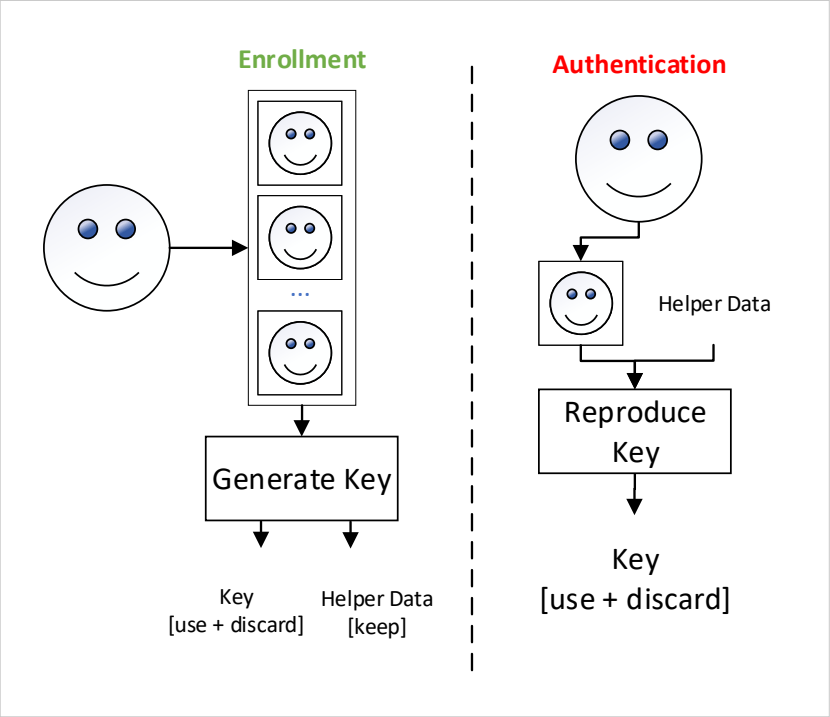


Figure 3.1: User Enrollment and Authentication.

3.2 Existing Privacy Preserving Schemes

The following sections will start with an explanation of an application of the fuzzy extractor schemes proposed by Dodis et al. in *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data* [DRS04] and the later quantization additions by Li, Sutcu, and Memon in a series of papers [LSM06, SLM07, SLM09] including *Secure Sketch for Biometric Templates* and *Design and Analysis of Fuzzy Extractors for Faces*.

3.2.1 Dodis's Fuzzy Extractor

As stated in the previous chapter, [DRS04]'s fuzzy extractor can be used to generate cryptographic keys from noisy information, which in our case would be biometric samples. The fuzzy extractor has a secure sketch, also described in the paper, and a strong randomness extractor component, such as a cryptographic hash function. More formally, the [DRS04] paper presents a general definition for the secure sketch and the fuzzy extractor primitives as follows.

Secure Sketch

Recall that the secure sketch primitive consists of a pair of procedures which we named *generate sketch* (`GS`) and *recover* (`Rec`). For a metric space M with a distance function `dis` and a threshold t, they define an (M, t)-*secure sketch* as a pair of randomized functions (here `GS` and `Rec` as in Figure 3.2) with the following properties:

1. The procedure `GS` on the initial input $in_0 \epsilon M$ produces a bit string

sketch ϵ $\{0,1\}^*$.

2. The procedure `Rec`'s inputs are an element $in_i \epsilon M$ and a *sketch* ϵ $\{0,1\}^*$. If $\texttt{dis}(in_0, in_i) \leq t$, then $\texttt{Rec}(in_i, \texttt{GS}(in_0)) = in_0$. If $\texttt{dis}(in_0, in_i) > t$, then no guarantee is provided for `Rec`'s output.

Figure 3.2: `GS` and `Rec` procedures.

While the secure sketch is error correcting and allows the recreation of the input string, it can't be used to generate cryptographic keys from biometrics. This requires additional components, as seen in with the fuzzy extractor.

Fuzzy Extractor

An $(M, t, l)-$*fuzzy extractor* consists of a pair of procedures *generate* (`Gen`) and *reproduce* (`Rep`) with the following properties:

1. The procedure `Gen` on the initial input $in_0 \epsilon M$ will produce two outputs: the extracted bit string $R \epsilon \{0,1\}^l$ and the *helper_data* ϵ $\{0,1\}^*$.

2. The procedure `Rep`'s inputs are an element $in_i \epsilon M$ and *helper_data* ϵ $\{0,1\}^*$. If $\texttt{dis}(in_0, in_i) \leq t$, and both R and *helper_data* were the outputs $(R, \textit{helper\_data})$ of $\texttt{Gen}(in_0)$, then $\texttt{Rep}(in_i, \textit{helper\_data}) = R$. If $\texttt{dis}(in_0, in_i) > t$, then no guarantee is provided for `Rep`'s output.

As with the secure sketch, there is an additional property about the entropy loss of the extracted string R given the existence of the *helper_data*.

Independently of the metric space, the usage of a secure sketch scheme would be as shown in figure 3.3:

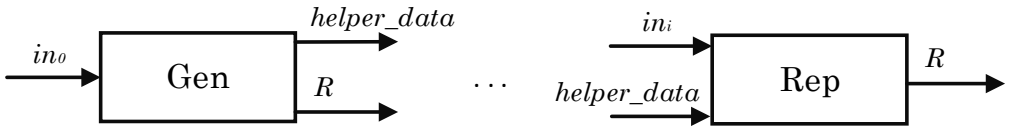


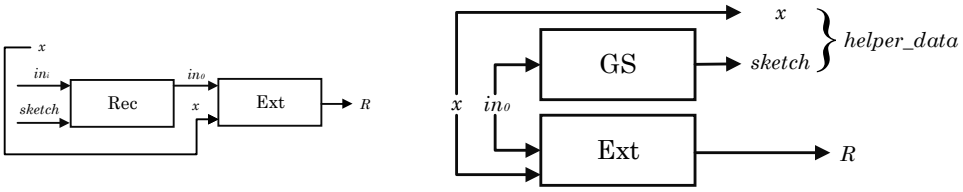
Figure 3.3: `Gen` and `Rep` procedures.

The creation of a fuzzy extractor requires both a secure sketch and a randomness extractor, such as a cryptographic hash function (`Ext` below). The construction is done as follows:

Assuming (`GS`, `Rec`) is an $(M, t)-$*secure sketch* and that `Ext` is $(x, l)-$*strong extractor* with the initialization parameters (if any) x and with an output of length l, then we can define the $(M, t, l)-$*fuzzy extractor* (`Gen`, `Rep`) as follows:

1. $\texttt{Gen}(in_0, x)$: then $\textit{helper\_data} = (\texttt{GS}(in_0), x)$, $R = \texttt{Ext}(in_0, x)$, and the output is $(R, \textit{helper\_data})$.

2. $\texttt{Rep}(in_i, (\textit{sketch}, x))$: then $in_0 = \texttt{Rec}(in_i, \textit{sketch})$, and the output is $R = \texttt{Ext}(in_0, x)$.



(a) Inner workings of *generate*

(b) Inner workings of *reproduce*

Figure 3.4: Full Fuzzy Extractor Primitive

3.2.2 Sutcu's Quantization Scheme

This section outlines the original discretization facial biometric authentication algorithm as described by Sutcu et al. in [SLM07, SLM09].

Template Representation

Assume we represent a feature vector of size n extracted from user i's biometric sample as

$$B_i = [b_{i1}\ b_{i2}\ \cdots\ b_{in}]$$

where each component $b_{ij} \in R$ is a real number and n is the total number of features in the template. There is variation in the possible coefficients for the components b_{ij}. This variation can be estimated using a set of multiple biometric features belonging to the same user. Using these features, for each component j, one can calculate the minimum and maximum values mn_{ij} and mx_{ij}, respectively. With these, it is possible to calculate the midpoints $\bar{b}_{ij} = (mx_{ij} + mn_{ij})/2$ and amplitude $p_{ij} = (mx_{ij} - mn_{ij})/2$ for each feature. At this step, for the i-th user, an authentic biometric feature $Bi = [b_{i1}\ b_{i2}\ \cdots\ b_{in}]$

would fall between

$$\bar{b}_{ij} - p_{ij} \leq b_{ij} \leq \bar{b}_{ij} + p_{ij}$$

for all $j = 1, 2, \ldots, n$.

Randomization

Following this, a user-specific random mapping is applied to increase noise tolerance, add cancelability, and increase diversity. That is, each user's extracted features are multiplied by Ri, a personalized square $n \times n$ matrix whose components are uniformly and randomly distributed in $[-1, 1]$. The post randomization mid-point vector $\bar{X}_i$ used in the secure sketch algorithm is determined by $\bar{X}_i = R_i B_i$. Similarly, a randomized amplitude vector δ_i is calculated as above from the randomized extracted features. At this step, for the i-th user, an authentic randomized biometric feature $Xi = RiBi = [x_{i1}\ x_{i2}\ \cdots\ x_{in}]$ would fall between

$$\bar{x}_{ij} - \delta_{ij} \leq x_{ij} \leq \bar{x}_{ij} + \delta_{ij}$$

for all $j = 1, 2, \ldots, n$. They note that step exists to provide cancelability and diversity, but does not provide dimensionality reduction or increase security by introducing non-invertibility as is often the case for random projection. This step is not present in [SLM07] as it does not affect the base functioning of the scheme. It can be seen as an optional improvement.

Quantization and Codebook

The following step consists of the use of a scalar quantizer to map the components of the feature vector Xi to a discrete domain. This is needed to apply

Sutcu' et al.'s secure sketch scheme for discrete domains. It also requires the existence of a *global codebook* C from which the user's individual quantizer is derived. This is done at enrollment time, where the many required sets of biometric samples are available. The creation of the global codebook is described here as the previous steps help its general understanding, due to their similarity. However, as described in the paper, this would happen at user enrollment.

Global Codebook Setup

The global codebook C consists of discrete domains C_j for each feature component as follows: $C = [\, C_1 \; C_2 \; \cdots \; C_n \,]$. C_j is derived from the global amplitudes. This is the global minimum $MN_j = \underset{i}{min}(x_{ij})$ and $MX_j = \underset{i}{max}(x_{ij})$ for the set of features extracted from the biometric samples available, using the same extractor as in 3.2.2. The discrete domain C_j for component j is calculated by quantizing the global amplitude by a quantization step δ_j. Essentially,

$$C_j = \{MN_j - r_j,\; MN_j - r_j + \delta_j,\; MN_j - r_j + 2\delta_j, \ldots,\; MN_j - r_j + L_j\delta_j\}$$

where r_j is a random number chosen to obfuscate the real component MN_j, and L_j is an integer chosen so that $MN_j - r_j + L_j\delta_j \geq MX_j$. These components are the codewords of the global codebook.

Of note, the distance δ_j between consecutive global codebook codewords is determined as a function of the smallest minimum amplitude observed among the feature vectors for component j. That is,

$$\delta_j = \alpha \, min_i(\delta_{ij})$$

where α is a scaling parameter.

Enrollment

The components of the i-th user's midpoint vector and amplitudes consisting of $\bar{v}_i$ and δ_{ij} can be mapped onto C_j. The discrete midpoint is the quantization of $\bar{v}_{ij}$, the codeword in the global codebook closest to it. The discrete amplitude d_{ij} is calculated by

$$d_{ij} = \left\lceil \frac{\delta_{ij}}{\delta_j} \right\rceil$$

And so, user i's j-th component's codebook C_j^i consists of one in every $2d_{ij}+1$ consecutive points in C_j.

The quantization of the j-th coefficient x_{ij} onto user i's codebook is denoted by $Q_j^i(x_{ij})$. Similarly to the global codebook, i's *user codebook* $C^i = [\, C_1^i \; C_2^i \; \cdots \; C_n^i \,]$.

Sketch Generation and Template Reconstruction

At enrollment time, users provide their biometric information from which templates B_i are extracted and randomized X_i. The midpoint vector $\bar{X}_i$ and the set of amplitudes δ_{ij} are calculated from these features and, using the existing global codebook, user codebooks are constructed. These user codebooks consist of the quantized midpoints $Q^i(\bar{X}_i)$ and the codewords in the global codebook that are integer multiples of $(2d_{ij}+1)$ global codewords away from the quantized midpoints.

The sketch P_i for user i is a vector $P_i = [p_{i1}\ p_{i2}\ \ldots\ p_{in}]$ where the coefficients $p_{ij} = Q_j^i(\bar{x}_{ij}) - \bar{x}_{ij}$, that is, a vector where each coefficient is the difference between the codeword closest to the respective midpoint value and the original midpoint value.

At authentication time, the authenticating user provides a biometric sample from which a feature vector is extracted and randomized. This noisy feature vector $\widetilde{X}_i = [\tilde{x}_{i1}\ \tilde{x}_{i2}\ \cdots\ \tilde{x}_{in}]$ and sketch P_i are used to calculate the reconstructed original midpoint vector. This resembles $Q_j^i(\tilde{x}_{ij}) + p_{ij}$ for $j = 1, \ldots, n$. The result will be the original midpoint vector if

$$-d_{ij} \leq Q_j^i(\tilde{x}_{ij}) - Q_j^i(\bar{x}_{ij}) \leq d_{ij}$$

where d_{ij} is the user's error tolerance for component j.

3.3 Additions to Sutcu's Work

Our final implementation is different from the scheme described above. We first implemented a simple version of the algorithm, that was true to the original paper. As the project progressed, and our needs changed, we have departed from the implementation described. In this section we will detail differences between the algorithm described in the paper and our final implementation. We will also provide additional details about the implementation that do not modify the scheme. Finally, we will detail some of the image preprocessing we ended up doing.

3.3.1 Scheme Modifications

Codeword Alignment Change

During our testing, we noticed a type of error that seemed to stem from an oversight in the original algorithm, or a part that was not detailed in the paper describing it.

Specifically, section 3.2.2 describes that the user codewords should be chosen to be one of every $2d_{ij} + 1$ consecutive global codewords, but how to choose which is not defined. Naively, one could set the starting user codeword to the global codeword closest to the user's midpoint vector. This would however mean that an approximation of the user's features could be found by brute forcing the values of the user codebook. This defeats the purpose of not storing the biometric features and reduces the user's privacy. A malicious actor that gains access to this information such as by getting unauthorized access to the database on which this is stored need only the public sketch P to reconstruct the midpoint feature vector.

Alternatively, one could try to match the global codebook creation by including every $2d_{ij}+1$ consecutive points from the first one in the global codebook. However, this leads to errors in the case where the midpoint vector ended up near the halfway point between two user codewords, as only a small variation in one direction could be tolerated.

There are multiple ways to fix this, and we as in [LSM06], addressed this by doing a small modification to the algorithm and subtracting the sketch P_i from the extracted feature before quantization. This requires a minor change to the Secure Sketch's *recover* (`Rec`) function, contained within the Fuzzy Extractor's *reproduce* (`Rep`) function. Rather than providing the raw

extracted biometric sample to be quantized, we will first subtract the sketch from the extracted feature before quantizing the result. This introduces no further changes to the algorithm. The secret reconstructed from the quantized feature vector

$$Q_j^i(\tilde{x}_{ij}) + p_{ij}$$

would now look like

$$Q_j^i(\tilde{x}_{ij} - p_{ij}) + p_{ij}$$

for all $j = 1, 2, \ldots, n$.

Codebook Representation

Rather than storing a user codebook as an array of the codewords from the global codebook for each feature, we can instead store user i's component codebook C_j^i as the following tuples:

$$C_j^i = (\, g_{ij},\ s_{ij}\,)$$

For each user i and each feature j, the *generating codeword* g_{ij} is the codeword from which all others are derived and emphuser step s_{ij} is the distance between those codewords $s_{ij} = (2d_{ij} + 1) \times \delta_y$. The set of codewords for a particular user-feature pair would be reconstructed as follows:

$$\ldots,\ g_{ij} - 2s_{ij},\ g_{ij} - s_{ij},\ g_{ij},\ g_{ij} + s_{ij},\ g_{ij} + 2s_{ij},\ \ldots$$

In practice however, we would not need to store the set of codewords. Before this change, the noisy biometric feature $\tilde{x}_{ij}$'s quantization would require finding the user's codeword nearest to it for that feature. Optimally, this would be done with an equation that takes into consideration the lowest and

highest codeword in the codebook for that feature as well as the distance between consecutive codewords. We can use a similar equation, using only the generating codeword and the step s_{ij}. The quantization of the noisy vector $Q^i(\tilde{X}_i)$ is calculated as follows

$$Q_j^i(\tilde{x}_{ij}) = g_{ij} + s_{ij} \left\lfloor \frac{\tilde{x}_{ij} - p_{ij}}{s_{ij}} \right\rceil$$

where $\lfloor \cdot \rceil$ is the rounding function. With this construction, codebooks are no longer a list of consecutive codewords. The global codebook can be constructed in the same way, with a generating global codeword g_j and global step δ_j. The new global codebook C construction to replace the one in 3.2.2 is as follows:

$$\begin{aligned} C &= [\, C_1 \; C_2 \; \cdots \; C_j \; \cdots \; C_n \,], \text{ and} \\ C_j &= (\, g_j, \; \delta_j \,) \end{aligned}$$

Note that the generating codeword g_j can be any codeword in the global codebook. And so, we set

$$g_j \equiv MN_j - r_j \pmod{\delta_j}$$

so that g_j can be seen to be a random value $0 \leq g_j < \delta_j$. The storing of codebooks is now less memory intensive, as each component requires only a tuple of size two rather than an array of indeterminate size.

Increase in Privacy

The above also provides the potential for benefit to user privacy in some specific cases.

We consider an attacker that:

1. Does not have access to users' biometric samples.
2. Does have access to the helper data, which is not held securely.
3. Has gained total access to the server-side data, including the global codebook, all user codebooks, the feature extractor, the randomness extractor, and all user randomization matrices.

A successful attack would entail either of the following:

- The regeneration of the key generated by a user's biometric, without using the biometric samples belonging to that user.
- The de-anonymization of a user by retrieving an image similar to their enrollment biometric samples.

Using the user codebook stored on the server, the attacker could attempt a brute-force attack to regenerate the user's biometric sample.

For a given user i, for each component j, the addition of that component's sketch p_{ij} to each codeword in the associated codebook C_j^i provides an attempted guess of the real quantized component $Q_j^i(x_{ij})$.

Each set of reconstructed components from each codeword represents a possible processed and extracted set of midpoint vectors equivalent to the original $\bar{v}_i$.

The attacker could then use the randomness extractor to regenerate the original biometric key from this vector, without the biometric.

Similarly, the attacker could instead attempt to recover an image similar to the user's face biometric sample. A multiplication by the inverse of the user's randomization matrix Ri^{-1} provides the average values of the extracted enrollment-time feature components. At this point, a correctly guessed sample would be similar to the result of an intermediate step during enrollment. Specifically, they have access to the averaged features of the enrollment samples, rather than the initial samples.

At this step, using the feature extractors, the attacker can invert the feature extraction process. Since we used PCA, this would require this recreated average vector to be multiplied by the inverse of the eigenvectors used for extraction, and an addition of the mean faces used to generate the extractor. These would both be present on the server, as they are used for enrollment.

The result would be imperfect, as not all components were kept with PCA, and would be an image from which the averaged enrollment features would be extracted. The user is now de-anonymized, and this emulated sample can be used to authenticate on other facial biometric systems.

A given feature extractors may perform optimally with as few as 20 features. In addition, due to the random nature of biometrics, codebooks for a given feature component may be fairly small; in our testing, we have seen some with as little as three codewords.

Given access to all enrolled codebooks, the attacker could search for the users with the fewest total number of possible codeword permutations, which is the product of all codebook lengths for that user.

If the attacker could find, within the set of user codebooks, a codebook consisting of only three features per component for each component, this would result in almost 3.5 billion possible combinations of codewords to try.

This number is not enough to prevent brute-forcing.

However, since in our modified construction each codebook consists of pairs

$$C_j^i = (\, g_{ij},\ s_{ij} \,)$$

any integer k now satisfies the equation that defines the codewords of codebook C_i^j:

$$C_{ij} = g_{ij} + k \times s_{ij}$$

leading to, in theory, an unlimited number of possible codewords, per codebook. In practice, k is restricted by the implementation. For example, if k is limited to 32-bit numbers, then the number of possible codeword combinations is 2^{32n}, where n is the number of components in the user's biometric template. Brute-force attacks can be made computationally infeasible by increasing the possible values of k further, for example to 128 bits.

This is more important if the global component variation is small, which leads to small codebooks, or if the user variation is large, leading to sparse codebooks. Other solutions to the problem would involve setting a minimum number of codewords per codebook, or providing non-invertibility with the matrices used for randomization (as in [TN05]).

Beta Scaling Parameter

We added another scaling parameter β. The α used in 3.2.2 provides control over the granularity of the global codebook codewords (by changing the distance between global codewords) and by extension, the codebooks in general. The β is used to modify the size of the user's discrete amplitude in 3.2.2.

The discrete amplitude d_{ij} is now calculated by

$$d_{ij} = \left\lceil \frac{\beta \times \delta_{ij}}{\delta_j} \right\rceil$$

which allows for a modification to the *strictness* of the algorithm. By choosing a β larger than 1 we can allow for less strict authentication, while decreasing it below 1 increases strictness. While there are limits to how finely this can be manipulated as the user codewords are still integer multiples of the global codebook codewords, the size of the global codewords can also be modified by tuning α.

Because of the discrete nature of the quantizing and since every quantized feature needs to match to a specified codeword for each user, there is no information about how narrowly a particular sample was rejected. As such, we created a scoring metric described in 3.3.2 to be able to evaluate the performance of the algorithm.

The addition of the beta scaling parameter was introduced by Brien et al. in [BTA].

Global Codebook Generation

The algorithm described by Sutcu et al. generates the global codebook at enrollment time, since it uses the enrollment samples to calculate the global steps in 3.2.2. This was not an issue at first, when we were testing with public datasets, as the entire group was enrolled at once. When testing with real users, and setting up the system to be used organically, we were unable to initialize the global codebook since few of the total users were providing samples at the very beginning. There were two workarounds for this.

At first, when starting the non-dataset testing, the global codebook steps were set to scalars that were deemed *small enough*. We simply set these at an order of magnitude smaller than the smallest ones we found experimentally when testing with existing datasets. We wanted a more rigorous approach.

Following this, we kept testing samples and added them to the AR dataset [MB98], since it was the one that most resembled our testing samples (see Figure 3.5) and expected images. We used this as a training set which we split into parts used to generate our PCA Matrix used as a feature extractor and our global codebook. This would be done when the application is first launched, and the results can be stored for reuse. Users could then be enrolled as we collected their biometric samples.

(a) Test image (b) subject m-020, image 1

Figure 3.5: Test image and AR face database [MB98] sample

3.3.2 Implementation Details

Feature Extractor

The quantization based Fuzzy Extractor scheme can be seen as a wrapper added after the feature extraction. It presents a method of evaluating templates and authenticating users based on their enrollment information and their authentication sample. As such, it is fairly agnostic to the feature extraction method used, and can work on any method that produces a number of consistently ordered scalar values. There are many feature extraction algorithms available, we used PCA (also called Eigenfaces in this context, as in [TP91]) for ease of implementation and time reasons, due to our familiarity with the algorithm. It also provided a stable baseline as it was the feature extractors used by Sutcu et al. in their fuzzy extractor papers. Optimizing the process for results is something that could be done in a final, commercial, implementation. Our PCA implementation was provided by the *Scikit-learn* Python library [PVG+11].

Strong Randomness Extractor Choice - SHA256

The fuzzy extractor scheme includes a randomness extractor to generate cryptographic keys from the biometric. We used the SHA256 hash function as our randomness extractor, with the biometric template serving as input. During our proof of concept implementation we would use the digest to symmetrically encrypt a small token consisting of some information recovered from their e-passport. We would store the encrypted token on the server, and provide them with the helper data, which was saved on their phone. At authentication time, the user would provide a new biometric sample and

their helper data, and we would regenerate the key from their sample and use it to decrypt the token. If the decrypted token had the proper "shape" - i.e. was recognized as an object, we would provide the stored passport information to the application, which would do its own verification. Otherwise, we assumed this was not the proper user.

While this seems sufficient for the proof of concept, we would use a more robust randomness extractor for key generation if this were to be used in a commercial context. The key derivation functions PBKDF2 [MKR17] would require the addition of a nonce for each user to use as a salt, but would require little modification to be added to the existing implementation.

Floating Point Errors

With one implementation, we noticed occasional mistakes that we found to be caused by floating point errors. This is to be expected as many of the variables in the process were derived from biometric information (global codebook step, PCA matrix, extracted features) and we were doing a large number of calculations with them. Since, floating point addition and multiplication are commutative, but not always associative or distributive, we found that some results were sometimes incredibly close, but not exactly the same. Remember that our key generation from the biometric samples functioned as follows:

$$\text{Bio-Key}_i = H_{\text{SHA256}}(\bar{X}_i)$$

and the recreated key was calculated as follows:

$$\text{Bio-Key}'_i = H_{\text{SHA256}}(Q_i(\tilde{X}_i - P_i) + P_i)$$

These errors could be addressed by rounding the features before the hashing, but this would reduce the entropy of the resultant key. We instead modified our initial biometric key generation to be as follows:

$$\text{Bio-Key}_i = H_{\text{SHA256}}(Q_i(\bar{X}_i) + P_i)$$

This alternate calculation would introduce the floating point errors to the initial midpoint vector before the creation of the biometric key, if such errors exist. The errors present at key reconstruction time would be accounted for and the regenerated key would match the key from enrollment time.

Scoring Function

We evaluated the similarity between the original midpoint vector and the reconstructed vector using a scoring function. Because the full algorithm hashes the result, we only know a binary a *pass* or *fail* result. During the testing, we could limit ourselves to the comparison of the reconstructed feature to the original one. Since the addition of the sketch to the quantized vector has no effect on the likelihood of accepting the new sample, we concentrated on the pre-quantization distance. Specifically, a particular authentication template will regenerate the cryptographic key if its distance from the quantized original codeword is below a certain tolerance. From 3.2.2:

$$-d_{ij} \leq Q_j^i(\tilde{x}_{ij}) \ - Q_j^i(\bar{x}_{ij}) \leq d_{ij}$$

for component j. We can measure how close or far each noisy feature was to the quantized features, as a proportion of the threshold distance:

$$\text{distance}(\tilde{x}_{ij}) = \frac{|\tilde{x}_{ij} - Q_j^i(\bar{x}_{ij})|}{d_{ij}}$$

If this normalized feature distance is greater than 1, the feature will map to another codeword, and the biometric key reconstruction will fail. As such, we only need to keep the largest normalized feature distance, which we call the score.

$$\text{score}(\tilde{X}_i) = \max_j \left| \frac{\tilde{x}_{ij} - Q_j^i(\bar{x}_{ij})}{d_{ij}} \right|$$

This is the L^∞ norm of the distances. At baseline level, a biometric template with a score 1 or less will be successfully authenticated, and one with a score above 1 will be rejected. The addition of the β parameter allows changes to the acceptance threshold so that any arbitrary score can be set as a threshold.

3.4 Preprocessing Input Images

The initial implementation was tested with the datasets freely available online. We used [MB98, SH, Spa07] and expanded the project to real time usage to meet the requirements for integration into the travel facilitation system. In 3.3.1 we explain that we modified the algorithm so we could provide a set of biometrics for initial Global Codebook construction, and to generate our PCA feature extractor. We got best results using our subset of the AR database. This is likely to be because of the similarity between our initial testing samples and the content of the database. The AR dataset had the clearest background, and we took images that resembled them in posture -

our applications had an oval where the user was instructed to have their face in the photo, as to center them in relation to those in the training set. The images could then be cropped around the face to minimize the effect of the background variation.

Widening our testing pool to new devices - phone and webcams, new locations, and new people lead to an increase in variation in the location of the face in the image. We could no longer reliably crop images in a pre-specified particular location. The variation between images also increased as the levels of freedom of the capturing device increased. On laptops the pictures were fairly consistent, with variation mostly being present in the distance and angle between the camera and the subject. With the phone app there was the possibility of change in pitch, yaw, and roll, and variance increased further. We needed to add a preprocessing step that would allow us to increase the similarity level within our training set.

The main issues we identified and where preprocessing might work were: addressing variations in lighting, the centering of the image around the face, and cropping around the face. While the feature extractor averages the image-wide lighting intensity, this corrects only for overall image brightness, not direction-based differences. One commercial application involved in the project had an advanced image processing component, but we were unable to use it prior to full integration.

We added a preprocessing step to address some of these issues, this mostly consisted of properly aligning faces, and we were unable to address the lighting-based variation. The only method which would do this was a way to select images that were least different from one another, but this method does not address differences across different sets of samples. It was, also, only used during the testing of the generated dataset images, as the in per-

son testing authentication attempts did not provide enough image samples to discard problematic ones.

Face Cropping and Resizing

When we progressed to pictures taken on mobile phones and non-constrained datasets, we first used the Viola-Jones detector [VJ01] to locate the faces within the images. The detector provides a loose bounding box around the face which serves as a *region of interest* within the image. The detector's bounds were not very consistent and as such the cropping was not very tight. The facial features were approximately in the same location, and we could resize the images to normalize the size. This type of preprocessing can address face location-based issues.

Face Alignment

When testing with our generated dataset, since the images were not taken with the intent to be authenticated, they contained a large amount of variation, similar to the shallow datasets we tested (see 2.2.2). The above face cropping detector could detect faces, but did not meaningfully address the various face angles. We replaced our face cropping preprocessing with a face aligning transformation. Since this change was only used when testing our generated dataset, it is detailed in 4.4.

Chapter 4

Dataset Creation

4.1 Motivation

As previously stated, our first implementation of the privacy preserving algorithm was a simple re-creation of the *Secure Sketch in a Continuous Domain* algorithm as described by Sutcu et al. in their paper [SLM09]. To test our implementation we used the same dataset - the *Olivetti Face Database* [SH] which is now known as the *AT&T Database of Faces* (see 2.2.1). This served as a prototype from which we built the rest of the project.

We were then included in the proof of concept for a system intended to accelerate the passage of travellers through airports. Due to external image processing and verification, we were required to work with images that loosely resembles passport images. More specifically, they were to be front facing, with the subject looking straight ahead and were to be taken either by the user on their phone like a *selfie*, or from a kiosk they would be facing. The dataset used, the *Olivetti Face Database* was similar enough for the time

being. It allowed for theoretical testing, but was insufficiently similar to the images we expected to receive and as such would not provide a proper testing set as the project advanced. We used *Principal Component Analysis* (PCA) as our feature extractor, which means that differences between the set used for *training* and the one used for *testing* are likely to lead to errors. Similarly, we use a set of extracted features external to those used for enrollment to create the *Global Codebook*, and discrepancies between sets is again a source of errors.

In parallel to the development of the code, we similarly wanted to improve the dataset to get a larger testing set. After the AT&T dataset, we considered the Essex Faces94 database [Spa07] which was larger, and contained more individuals, specifically, 126 compared to ORL's 40. We eventually used a subset of the *AR Face Database* [MB98], with some images removed. The AR Face Database had two sets of images per user, which were taken 2 weeks apart, and more images per person. We excluded the images with accessories and those that were very different from the mean image, such as the one where the subject is screaming, and then combined the image sets to get a large enough number of samples per person. This modified dataset was smaller than the Essex dataset, but the images more closely matched our testing set.

The first part of our inclusion in the project consisted of showing that the addition of the a *Fuzzy Extractor* privacy preserving "wrapper" to a biometric authentication system did not prevent its use on a mobile device. We created a phone application that took the authentication pictures and worked in conjunction with a server to authenticate users. At this point we were generating our PCA model and the *global codebook* with a subset of images from the AR dataset that were cropped to match what we would be getting from the application. The testing was done with a different subset of the AR

dataset images and photos of people testing our application.

This testing exposed some of the issues that would stem from user-based testing. The most notable stemmed from a lack of variability. Recall that multiple biometric samples are required of each user to create their codebook. Some people, when enrolling, would take their multiple pictures by posing and tapping the *capture* button several times. In these cases, the pictures had an almost unchanging expression and the little variations were due to slight movements of the hand when pressing the button. This created very strict codebooks which required an adjustment of the strictness parameter β. To increase variability, we prompted people to say a small phrase or even sing while taking the pictures. This increased the variation in expression (mostly in the mouth and nose area).

In addition, we were now testing using images that were fairly conceptually removed from those used to train the algorithm. The training images were taken in a lab setting, with consistent lighting, consistent pose, and predetermined expression. We then shifted our attention from trying to improve the algorithm for our specific use-case to trying to find a dataset whose content either matched our expected images and the samples we were receiving, or could be modified to do so. This is also where we first explored the idea of creating our own dataset, with a small application.

The privacy preserving algorithm was then in part integrated in the proof of concept system. At this stage, we were receiving enrollment and authentication requests from the testing participants that were similar to expected usage. There were now, due to the integration in the system and due to the fuzzy extractor scheme, an increasing set of constraints on our images. Since the application validated the images by comparing them to the passport image, we would only receive images that would somewhat resemble

them. Of course, we could preprocess them to crop them after. We searched for datasets containing images that were more representative of this expected usage that still satisfied our early criteria (enough images per user to enroll, labelled images, speaking, etc.). A new dataset more closely resembling the enrollment images should both increase the efficacy of the algorithm, and allow for more "offline" testing: testing without requiring users to enroll and authenticate on the system.

4.2 Data Source

As previously mentioned, we considered making our own dataset to train and test the system. The main advantage to creating a dataset is the ability to tailor it for its specific use case. However, this is very expensive in terms of time and effort, as such it is often easier to use one that is already available. Given that we were unable to find a publicly available database of faces that matched all our requirements, nor were we able to modify one to do so, we explored whether making our own cheaply was feasible. At first, we simply wanted images that allowed for more offline enrollment and authentication testing. To this end, we intended to store the images of willing actors passing through our system and encouraging more people to participate in a trial. This proved to be fairly difficult as the system was location bound, and we weren't able to enroll users without having them go through the entire process, which was fairly involved. The total amount of time spent for a single sample was quite significant, and so we explored other avenues.

While trying to find new datasets earlier in the project, we looked at creating a set of images from online sources. The first implementation of our algorithm which authenticated people while they were using the application did so by

capturing frames from a webcam. This was what led to the idea, while trying to solve the issue of low variability between images, of recording a video of users reciting a predetermined phrase and extracting frames from said video. We weren't able to generate an appropriate amount of images that matched our use cases from sets of images online, but it seemed possible to do so with videos. This enabled us to emulate the process by which we acquired enrollment and authentication frames in the proof of concept application.

Force this, it can be useful to see the requirements as belonging to three categories: requirements for the source website, requirements for the contained videos, and requirements for the extracted images.

4.2.1 Image Requirements

The requirements for the image frames were quite simple, and closely matched those of the images given to the algorithm:

- The image had to be of a single person.
- The image's subject was forward facing. This matched the passport like nature of the input images.
- The face in the image should be mostly non-occluded: not enough to prevent face detection or the extraction of features, but some coverage such as by some hair, or common eyeglasses is acceptable.
- The face in the image should be centered enough so that the entirety of the face is inside the frame.
- The face within the image should be large enough. We tried with both 60 pixels wide by 60 pixels tall and 80 pixels for both. We used 80 as

one of our detectors has a minimum detection size. The detectors are discussed later.

The frames that satisfied these would be worth considering for our dataset.

These were the video requirements:

- The subject of the video should talk, sing, or do other similar actions. We needed a small amount of variation between images.
- The video should have a single person in it. This simplified the identification of the subject.

The second requirement proved to be challenging. We did no extra validation on the images once they were captured, and so a lone frame with a different subject will not be automatically detected. Our process contained measures to detect videos that seemed likely to contain more than one subject, but they are fallible. Videos that were likely to be caught by these measures were acceptable, as they would be discarded. This led to what we called the *same person problem*.

4.2.2 Same Person Problem

The *same person problem* is an issue that we encountered as we tried various data sources. Since authentication consists of correctly distinguishing matches from non-matches, testing requires labelled images indicating the identity of the subject. For some source websites (such as *archive.org* [arc]), our inability to identify the same person across multiple videos or different people within the same one rendered the source unusable. On others,

we could sometimes link people based on metadata, such as the uploader's username.

We distinguished this by calling the two sub-problems *intra-video linking* and *inter-video linking*. Intra-video linking is the linking of an individual within a video. Solving it completely would allow us to use any video and generate datasets on almost any data. However, this is basically the facial recognition problem. Realistically, we simply need to be able to ensure only a single person's image was extracted from each video. We could then use "video sets" of images for enrollment as one person. Inter-video linking, however, came with a different set of challenges. If we identified sets of images by video, we could only use one video per person. Otherwise, we might get *false* false positives if we attempted to authenticate a person with their own images from another video. This means that being able to link a given person across videos would both allow for the use of a set with more videos per user and allow the testing using authentication images belonging to the same user, but from a different video. This could give more meaningful results as, in the expected use-case, there would be a significant amount of time between the user's enrollment and their authentication.

4.2.3 Source Requirements (Cont'd)

For the source website, we needed a large repository of videos. First, this allows for a large resultant database of images. More importantly, this also allows us to be more strict with the provided samples. Because of this, we can trade off total dataset size or creation speed with higher quality images. In addition, we need the source to provide a way of inter-video and intra-video linking of persons.

4.2.4 Video Sources

Our first video source was the video posting and sharing service *Vine* [vin]. Navigating it automatically was fairly simple. The videos were short, up to 6 seconds in length, and the main page had some of the more popular videos separated by categories. Vine seemed promising since many videos, especially in the Comedy category, were short clips of a person doing something in front of a camera. It was easy to use a face detection algorithm to isolate the videos which we could use. With the assumption that each account was only operated by one user we could use the account name of the uploader to allow the possibility of inter video linking of people. This would allow the creation of groups of images taken by a single person at different times. However, since the site disabled uploads in late 2016, we could not easily grow the dataset by using the daily published content. We quickly exhausted the easily accessible search space. Navigating previously published content effectively was difficult, and we had not found a way to aggregate, identify, and link usable video sources.

We then moved on to *archive.org* [arc] which is an archive of broadcasts that aired on TV. We directed our focus on news segments in effort to extract the images of anchors or public persons while they were announcing. This too seemed promising, as the individuals spoke, were often centered, and faced the camera. This increased variability, provided consistent behaviour that facilitated our task, and meant that the images taken from the video were similar to passport images in pose. There were unfortunately two main problems with the site. For one, we couldn't easily identify the subject of the image, as each channel had multiple anchors and this information wasn't in the metadata. Secondly, there were often other people that were shown on screen, either when they were being interviewed, or when they were

being discussed. Because of this, different people would be labelled as the same subject. We were able to mildly attenuate the second issue, but not completely fix it. Because of this, we were unable to use *archive.org*.

Finally, YouTube [YT] had a much larger amount of content than the previous two sites, and a large user base which continuously uploads content. We needed to distinguish videos from which we may be able to easily extract face images and from the previous two attempts we identified some difficulties we would see.

4.2.5 Intra-linking People on YouTube

From our prior experience from working with Vine and archive.org, we knew that ideally, we would need to link persons within a video (intra-linking). In chapter 2 (section 2.2) we described the difference between what has been described as *deep datasets*, which contain multiple sets of images belonging to individuals, and *shallow datasets* that are comprised of images that, while potentially linkable, are simply collections of images of people. We needed a deep dataset to adequately test our project.

In regards to intra-linking, we need multiple images of the same person for enrollment and some additional ones for authentication. As such, the ability to distinguish between individuals in order to make sets of images is paramount. As stated before, this was almost equivalent to solving the problem for which we were doing the proof of concept. Instead, we limited the videos from which we could extract images to ones with a single person, and, like with Vine, assume the person in the video is always the same person. In addition, we implemented some basic checks to skip over videos that are unusable, such as ones with special guests.

From our time working with *archive.org* we had a few rudimentary ways to detect *guests* in videos - people that were not the owner of the channel. One such way was simply running a face detection algorithm on the entire frame, including the non central part we would normally exclude, and exclude the outside faces later during post-processing (see 4.4). The faces present near the edges of the frame were not included in the dataset since they were more likely to have a non front-facing angle or be partially off screen. However, by running the face detection algorithms on the entire frames as a test, we could detect videos where multiple people were present. Videos with multiple people would be removed from consideration during the image acquisition phase.

We also explored the possibility of tracking the position of faces over time. If we only detect a single face with sudden drastic changes in location, we may have a video that has multiple people staggered such that only one is in view at once, such as an interview. In practice, however, many channels used many *jump cuts* in their videos, and we had not found a simple method to differentiate between the two that didn't involve facial recognition. Consequently, we simply elected to avoid channels where we found such videos.

Passing the frames through an image similarity measure could be another possibility to address this issue. In this approach, we would extract a larger number of images, but only keep a set of most similar images. This is different from the face recognition as the similarity would be evaluated in a method that is agnostic to the presence of faces. The idea resembles the one above - images that are similar to each-other, whether by evaluating the entire image or a region containing a face, are more likely to contain the same person. To provide such similarity measure, we implemented an image hashing algorithm, these are a subset of perceptual hashing. This idea is somewhat reminiscent of [WHM11]'s use of the difference in image background to test

link people across videos.

4.2.6 Inter-linking People on YouTube

On *Vine* we could identify a video source from the uploader's user account identification string. Some accounts belonged to a single individual and we could assume that all videos uploaded there contained images of the same person. We wanted to do something similar with YouTube and found that User Channels were mostly equivalent. In terms of inter-linking, we use the uploader's channel as an identifier. We would assume all the images from videos uploaded to the same channel would contain the same person, while those from other channels would contain different people.

4.2.7 Channel Requirements

This, however, added restrictions to the channels we could use. The channels from which we could extract information were those that followed the following criteria:

- Have the same person in each video. Some channels had series of videos with different hosts, but each video would only feature one. This wasn't detectable with our algorithm.
- If the channel has frequent guests, the guest and channel owner need to be in frame together. Since one of the methods we use to ensure the video only has one person involves checking to see if multiple people appear in the same frame, we could accept channels with some (even

frequent!) videos that included multiple people, as long as they were in frame together enough for us to detect and discard the video.

- Need to have a large number of videos with a single person. While we could do some validation and discard some videos, some channels were so barren of useful videos that their inclusion increased false acceptance more than they contributed to usable content.

- The videos need to involve the proper type of action. Some channels fail to provide enough variation between facial images. Notably, one channel consists of a person smiling at a camera. These videos are about 4 hours long.

- We also excluded channels that had characteristics that were difficult to otherwise address. One such example is the inclusion of a microphone over the person's face. Microphones, when obstructing the lower part of the face, would rarely occlude the face to prevent a detection, and would often show up in our samples.

Due to this, we hand-picked the channels that would be included in the list to be evaluated. This was more involved than having a fully automatic process, but was much less effort per image than taking photos manually or asking people to enroll in the app. The YouTube channels we used were aggregated from the results when searching for keywords that were likely to fit our requirements such as – "vlog", "make-up", "review", and from channels provided to us by people we queried. The channels were then manually verified, including a sped-up play through their most recent videos to see if they seemed to match our requirements. We stored the channel URLs of those who did in a text file that was provided to the algorithm.

We expected the end result to be a dataset created from YouTube videos

where the channel is used as the person's identifier and the videos are used as the source of frames. This would allow us to use images from a video as a set for enrollment, and different images from other videos on the same channel to authenticate, as a way to represent changes due to the temporal difference between the two. We could also replicate other types of datasets with our extracted images. Using only a single video per person corresponds to existing manually constructed deep datasets with many images per person that have been taken in a single sitting. Using only one image per video for each person replicates some scraped shallow datasets with multiple images per person, from different sources.

4.3 Image Acquisition Process

The program first runs an initialization and maintenance sequence to detect the presence of a database and correct folder structure present from prior runs of the program. In that case, the results will be added to existing ones. Otherwise, the required directories are created. The dataset creation process is as follows:

4.3.1 Aggregation of Videos

First, the function will look at a text file provided containing URLs to the YouTube channels that can serve as video sources. YouTube channels can have URLs of the following three different shapes:[YTh]

`https://www.youtube.com/user/<user_name>`
`https://www.youtube.com/channel/<channel_id>`

`https://www.youtube.com/channel/<channel_name>`

The first is a legacy type of channel URL that exists for certain old accounts. The second uses the channel_id that is automatically assigned to every YouTube account at creation time. The channel_name URLs are custom URLs available to channels that reach a certain size.

The program extracts the variable part of the URL if it is of one of the above shapes, or the channel URL if another type of URL was given (like another page on the user's profile). The extracted string is appended to $'https://www.youtube.com/'$ to generate a general URL shape for the program.

For each of these, the channel's title is retrieved, which allows for human identification at a quick glance. With this, it also retrieves information on the channel's last 30 uploaded videos: their URL, title, and length. This is also stored in the database. Failures at channel acquisition are stored in a text file for manual review.

4.3.2 Frame Extraction

Strike System

The frame extraction process reads the video information in the dataset, and generates a list of those with a "scraped" tag set to *False*. Some parameters are loaded, such as *minimum amount of time between consecutive faces* and *video playback speed*.

For each video in the list, the video URL is modified to start at a particular timestamp (initially 30 seconds). This is done to skip introduction segments that may be repeated across multiple videos on the same channel. The video

is then opened in the browser, and the playback speed is increased using a JavaScript bookmarklet.

During the video runtime, the location containing the video is screened for faces using the detection function described below. Failure to detect a face has no effect. If a face is detected some information is saved. Then, a small delay, defined by the minimum amount of time between consecutive faces, is introduced before the procedure continues.

We employ a strike system to exclude videos that may contain more than one person. If the face detection function locates more than one face, the video receives a strike and a larger delay must pass before the next frame capture. Consecutive multi-faced frames increase the delays, while a frame with only one resets it. This serves to efficiently skip over sections of the video where more than one face may be detected. Strikes also remove the saved face that was found most recently. If the number of strikes reaches the threshold amount, the entire video is rejected and this procedure stops. The threshold is initially set at 5, but is increased by 1 for each 10 proper images detected in the video. Rejected videos are given a "bad" tag in the database, and their detected images are saved in a separate location to allow for human review.

As the strike system also removes the previous image when multiple faces are detected in a frame. In practice, as the strike system also removes the previous image when multiple faces are detected in a frame, we can extract the correct frames in some videos where other faces may be present. For example, while the images extracted from a proposal video (see Figure 4.1) feature only the owner of the YouTube channel, the video features multiple frames with at least two people (see Figure 4.4).

But, as previously noted, this system does serve to detect videos with jump cuts, such as seen in Figure 4.3. This series of images was the most poorly performing (in terms of strikes to final images) set of images accepted during our testing with the most lenient parameters, and led to the addition of the scaling strike system to allow further strictness without limiting the number of extracted images.

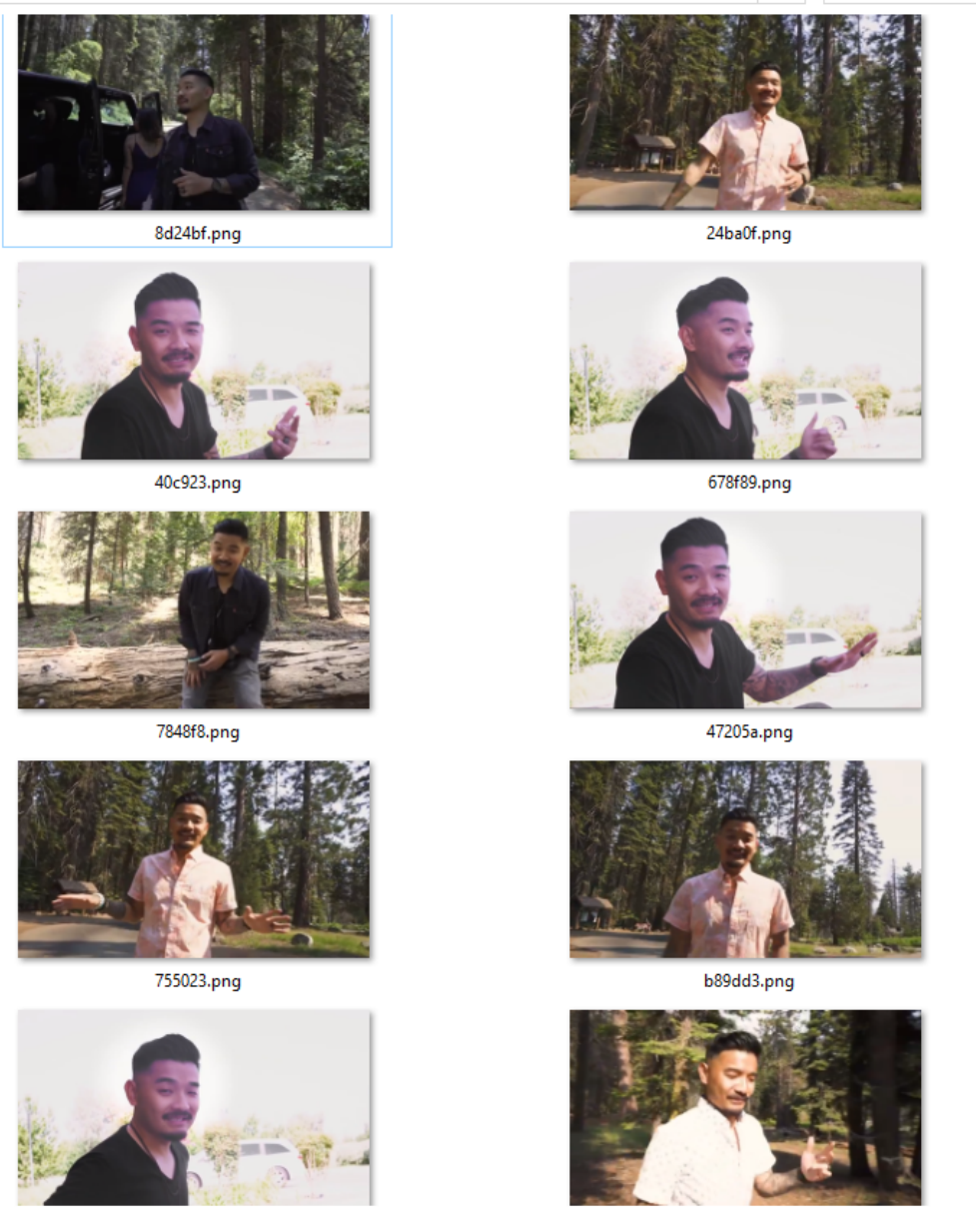


Figure 4.1: Frames extracted from proposal video

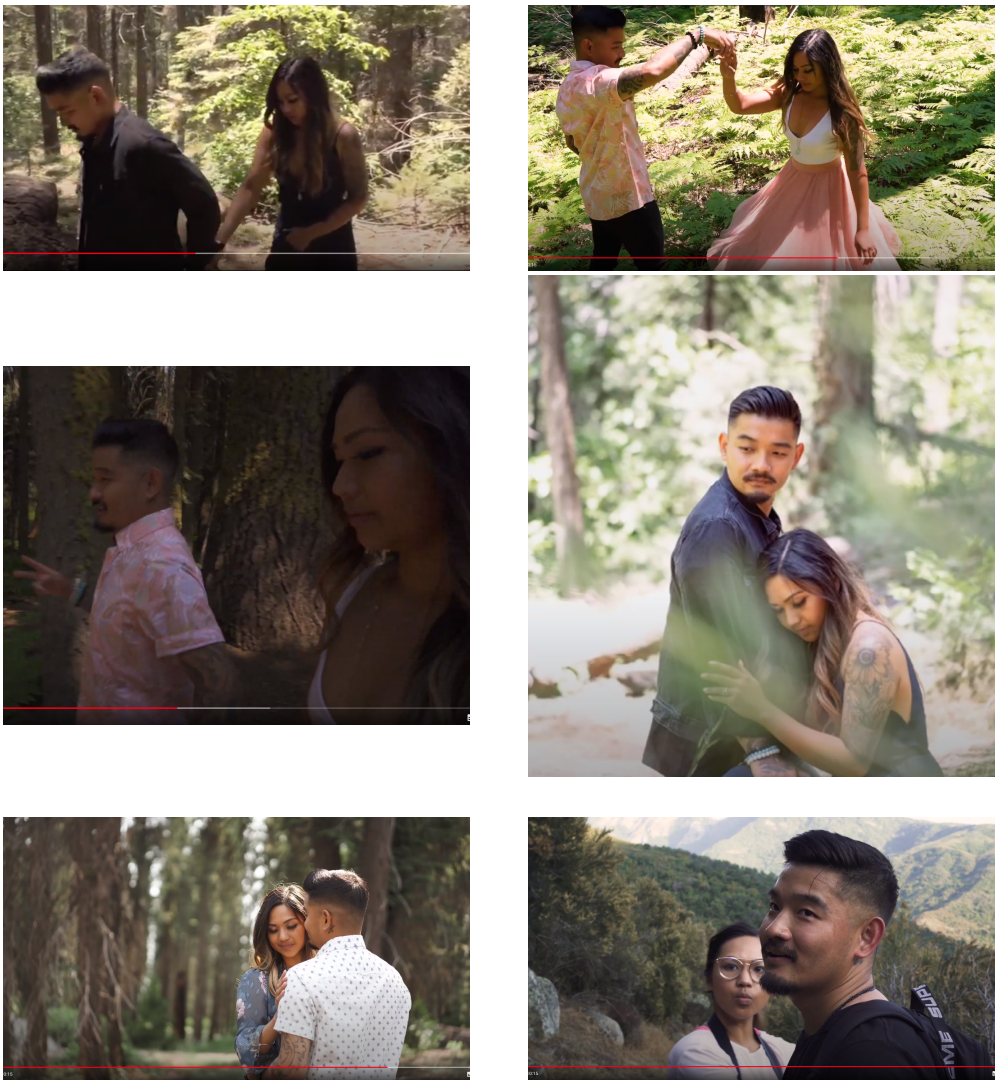


Figure 4.2: Frames showing the presence of second person

Figure 4.3: The camera position switches repeatedly during this game of Battleship

Face Detection

The face detection function is a two part process where two face detection algorithms are used in sequence. A *haar-like feature cascade classifier* [VJ04] is provided the image during the first pass, followed by a *Histogram of Oriented Gradients* [Kin09] (HOG) face detector, both of which are given a minimum and maximum face size to detect. The default values are set to 80 pixels by 80 pixels, as Dlib's HOG detector was trained on images of at least that size. The haar cascade detector is less accurate, but faster if the HOG is given an upscale parameter larger than 0, which allows the detection of smaller faces. Haar cascade classifier is used to filter out images. Frames on which the haar cascade detector identified no images are skipped over, while the others are passed onto the HOG based face detector:

If the HOG detector detects no face, it is assumed that a face was falsely identified by the less accurate haar cascade detector, and the frame is skipped as though no face was detected.

Frames on which the HOG detector detects more than one face provide a strike against the video, as detailed above. Both detectors are used in this case as the haar classifier is less accurate and sometimes falsely detects faces in random objects. The HOG detector is used as a trusted verifier to confirm the presence of more than one face in the frame.

If exactly one face is detected by each detector, it is assumed that the frame contains only that face. The face detection function returns the following information, which is saved: the region of interest containing the face, represented as a bounding box; the entire video frame, which may become an image in the final dataset; the timestamp within the video where this frame resides. There is then a small delay before attempting to evaluate the next

frame.

The process behaves as though the information provided by the HOG detector is correct. It is more accurate in both correctly locating faces, and in discriminating non faces from real faces. It is the more advanced detector. As such, it can detect partially occluded faces better than the haar cascade classifier, and accepts a larger range of face angles (more likely to detect face profiles), and partially cropped faces. By limiting the frames sent to the HOD detector to only those where the haar classifier also detected faces, we can leverage the haar detector's strictness, without being vulnerable to its propensity for false detections. Because we filter for less occluded, more forward facing faces, our images were more likely to resemble passport photos.

(a) Including screenshot of other video (b) Note the masks in the background

Figure 4.4: Samples with two faces detected

The image acquisition phase ends when the requisite number of faces has been extracted from the video; when the video has accumulated the threshold number of strikes; or when the time remaining in the video is considered too small (30 seconds), to skip a possible ending segment that may be repeated across videos.

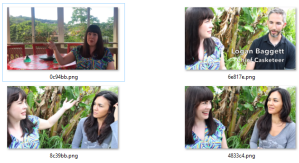
After this step, there is a final confirmation: if the number of frames containing faces detected is over a threshold amount, and if the video has not been

assigned too many strikes, then the video is accepted. The extracted frames are stored as images on disk, along with their source video, timestamp, region of interest, and file path in the database. If the video is rejected, the frames detected until its rejection are also saved, to a location dedicated to videos that were not accepted. This allows for later verification. The video's database entry is then edited to show it's been explored.

Figure 4.5 shows videos that were rejected due to large number strikes. The reason for the first video's rejection was not apparent, as there were no frames with multiple faces in the rejection folder. Further inspection showed that all frames containing more than one person were removed by later strikes.

(a) Saved rejected image

(b) Purged by later strike



(c) Guest on channel

Figure 4.5: Videos rejected by strike rules

4.4 Dataset Image Processing

Since the generated dataset is populated by images extracted from videos that were not intended to be used for authentication, there is a great deal of variation between images. It is possible to attenuate this variation using clever processing of the extracted images. We have several possible preprocessing transformations available. We have generated multiple versions of the datasets with different image processing techniques that can be used. The

following transformations are available to us:

- Face cropping
- Facial landmark alignment
- Facial landmark detection
- Perceptual hashing-based group image-similarity ordering

First, as mentioned previously, a detector can be used to locate faces within an image. Using the coordinates of a bounding box, images can be cropped around the face. The images can also be resized to regularize the size of the face within. This also serves as a rudimentary way to align features, as faces are somewhat centered after cropping. We have generated datasets where the faces are cropped and resized to 80 by 80 pixels. We also created datasets where the bounding box is increased by 20% and 50%.

The authors of the *Dlib* library [Kin09] have released a shape predictor for faces that detects 68 landmarks. These landmarks correspond to specific locations surrounding the face and outlining the eyes, nose, and lips. Dlib also contains methods that apply a linear transformation to the image to align these landmarks with default positions based on a forward facing face. These can be used to align them based on the position of eye and nose landmarks, or based on the lower mouth area.

These landmarks were also used to further filter the images. While post-processing the images of the dataset, we would exclude any image on which these facial landmarks could not be detected. Figure 4.6 shows images rejected by this step.

Figure 4.6: Images rejected using facial landmarks.

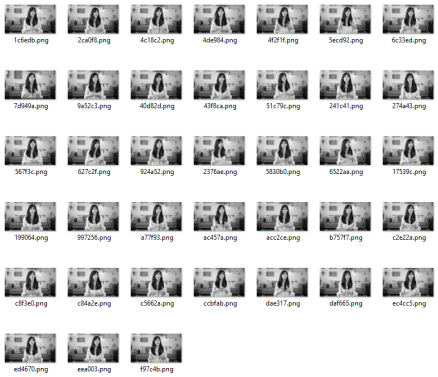
There is also the possibility of using group transformation. In our case, these would take the set of images from a single video as input to generate a subset for inclusion in the dataset. Using a type of image hashing, we can group images by similarity. Very simply, we can generate a subset of images that look the most like the average image in the set. This often removes outliers from the set, which helps reduce the number of unwanted images that may not have been caught. In practice, this served to filter out images from certain channels that occasionally used stock reaction images for humorous intent. We created tighter versions of the dataset that contained up to 12 of the most similar images per video.

We implemented a fairly simple perceptual hashing algorithm. For a grayscale image, the image-wide average pixel intensity (brightness) is calculated. The image is split into an 8 by 8 grid. The average intensity is calculated for each grid square, and the square is assigned a binary value indicating whether it has higher than average intensity for the image. With these 64 values used

as a bit string, the hamming distance between two image strings can be used as an approximation of their dissimilarity.

Figure 4.7 shows the results of the post-processing steps on example images.

(a) Sample frame



(b) Collected samples

(c) Cropped and resized

(d) Aligned based on eye position

(e) Sorted by similarity

(f) 12 most similar images

Figure 4.7: From video to processed image set.

Chapter 5

Results and Analysis

5.1 Dataset Generation

5.1.1 Increasing Dataset Size

The image acquisition process is automatic, but requires an initial list of YouTube channels to search. The process's final parameters were determined by repeated testing and manual evaluation of the dataset. This consisted of separately saving the images that were rejected along with the reason for the rejection, and manually inspecting the images. Several combinations were tested, and three of these created datasets are explored below. The testing described in this thesis was done on further post-processed datasets, meaning there was additional manipulation done on the image frames extracted from the videos. These post-processed datasets stem from the *Strict* dataset below.

The first dataset was generated from a curated list of 23 YouTube channels.

It served as the baseline dataset to first test the feasibility of the dataset generation algorithm. These initial results are contained in Table 5.1.

Table 5.1: Original dataset details

Initial Number of Channels	23
Videos Reviewed	672
Contributing Number of Channels	23
Contributing Number of Videos	263
Contributing Videos Percentage	39.1%
Number of Images	9,544
Non Contributing Channels	0

To test the scalability of the generation method, another dataset generation was attempted, but with a larger list of 71 YouTube channels chosen in a less strict manner. The number of strikes allowed per video was increased from 6 to 10. This *Expanded* dataset's initial results are contained in Table 5.2. The percentage of videos that contributed to the dataset is about the same, 39% for the curated list of channels to 40% for the expanded one. This indicates that the more lenient channel choices and the lower strictness (number of allowed strikes) nearly canceled out in terms of likelihood of video rejection.

Table 5.2: Expanded dataset details

Initial Number of Channels	71
Videos Reviewed	2,064
Contributing Number of Channels	69
Contributing Number of Videos	823
Contributing Videos Percentage	39.9%
Videos Rejected Stritly by Strikes	118%
Videos Striked Out	118%
Number of Images	26,189
Non Contributing Channels	2

For the Expanded dataset, 428 of the 823 contributing videos have reached the limit of 40 images per video. The rest were either too short to extract the full amount, did not contain enough frames with faces, or strike-based delays prevented the full acquisition. 20 videos provided only the minimum of 10 images.

See Table 5.3 for more information about the extracted images. Given more time, it would be interesting to use this data to automatically identify videos or channels as candidates to be filtered out.

Table 5.3: Image information by contributing channel and video.

Images (26189)	**Channels (69)**	**Videos (823)**
Mean number	379.6	31.8
std dev	245.0	10.5
Minimum number	11	10
25th percentile	171	23
50th percentile	350	40
75th percentile	537	40
Maximum number	1,117	40

The final, *Strict* dataset followed the Expanded dataset's creation, but with the number of strikes before rejection set to 6. It used the same list of 71 YouTube channels. This Strict dataset's initial results are contained in Table 5.4. It was further processed and used for testing.

Our datasets compare favourably to existing deep datasets in terms of number of images, and number of image sets per subject (see Table 5.5. In both the first and Expanded versions, the median video provided the maximum number of extracted images per videos. Further steps would include increasing the number of channels used as source.

A large number of problematic videos are rejected before the need for further

Table 5.4: Strict dataset details

Initial Number of Channels	71
Videos Reviewed	2,111
Contributing Number of Channels	59
Contributing Number of Videos	403
Contributing Videos Percentage	19.1%
Number of Images	10,895
Non Contributing Channels	12

Table 5.5: How the generated datasets compare to other deep datasets

Generated datasets	**Base**	**Expanded**	**Strict**
Total subjects	23	69	59
Set size	30	40	27
Sets per subject	16	11	5
IMG per subject	464	319.5	92.5
Total IMG	9,799	26,189	10,895
Public Datasets	**AT&T**	**AR**	**Essex94**
Total Subjects	40	126	153
Set Size	10	13	20
IMG per Subject	10	13×2	20
Total Images	400	4,000+	3,080

processing. Figure 5.1 shows that, even in the least strict case (the second, Expanded dataset), most videos are rejected before finding any images that would be used for extraction, if the video were accepted.

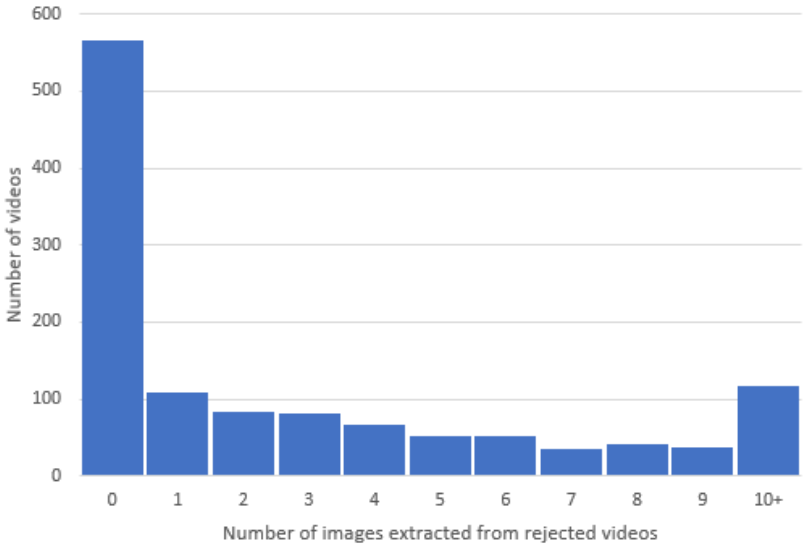


Figure 5.1: Number of images before final video exclusion.

5.1.2 Image Post-Processing

As mentioned in the previous chapter, the second pass filters out images for which the Dlib's shape predictor can't detect the facial landmarks. This is done as part of post-processing rather than during image acquisition as it is slower than the other detectors, and so is run offline. The images can then be cropped, resized, aligned, or filtered further according to perceptual hashing. Table 5.6 contains the derived dataset details where: **Algn** represents whether the images in the set have been aligned based on eye location, **p-Hash** represents whether only the 12 most similar images have been kept per video, and **Brdr** is the multiplicative size increase of the face bounding box before cropping.

Table 5.6: Post-processed dataset information.

Dataset	**Algn**	**p-Hash**	**Brd**	**Images**
Cropped_resize			1.0	10,668
Cropped_resize_sft12		✓	1.0	5,092
Cropped_m12			1.2	10,640
Cropped_resize_m12_sft12		✓	1.2	5,087
Cropped_m15			1.5	10,573
Cropped_resize_m15_sft12		✓	1.5	5,073
Aligned	✓		1.0	10,688
Aligned_sft12	✓	✓	1.0	5,097
Aligned_m12	✓		1.2	9,851
Aligned_m12_sft12	✓	✓	1.2	4,947
Aligned_m15	✓		1.5	10,427
Aligned_m15_sft12	✓	✓	1.5	5,047

5.2 Fuzzy Extractor Algorithm Testing

5.2.1 Testing Results

The modified algorithm was tested on the *Essex 94 dataset* [Spa07] (see 5.2.1), the *AT&T Database of Faces* [SH] (see 2.2.1), and the *AR Face Database* [MB98] (see 2.2.1), to serve as baselines for comparisons. Ideally, our modifications to the fuzzy extractor scheme do not negatively affect its performance.

In biometric authentication testing, the *False Acceptance Rate* (FAR) is the likelihood that an authentication attempt is successful when the enrollment and authentication subjects were different. Similarly, the *False Rejection Rate* (FRR) is the likelihood that an authentication attempt is unsuccessful when the enrollment and authentication subjects are the same. By varying the acceptance threshold, one can be increased at the expense of the other. The *Equal Error Rate* (EER) is the likelihood of error when the acceptance threshold is set so the FAR and FRR are equal.

We use the equal error rate as our comparison metric when comparing testing results.

Essex Faces94 Database, 1994

Each of the 153 subjects' 20 images were converted to grayscale. Since these images were generated in a laboratory setting, these tests were done without preprocessing. This was the largest public dataset we used for testing.

The dataset was first used to populate a training set consisting of a randomly

chosen image of each user, used to fit the feature extractor and to generate the global codebook. The remaining 19 user images were used for testing. Of these 19 images, one was chosen randomly to be the authenticating image, while the other 18 were used for enrollment. Each of the 153 subjects provided an authentication attempt against each of the 153 subjects, for 23,409 total attempts. The EER at the various number of components and α values are shown in Figure 5.2.

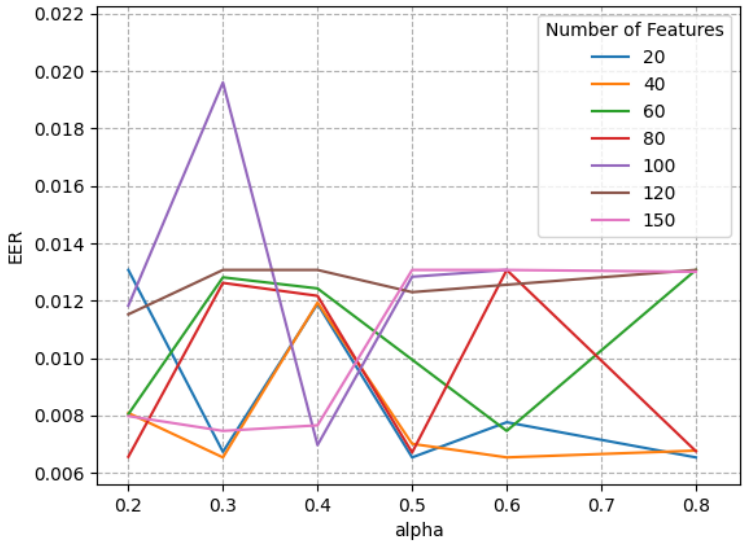
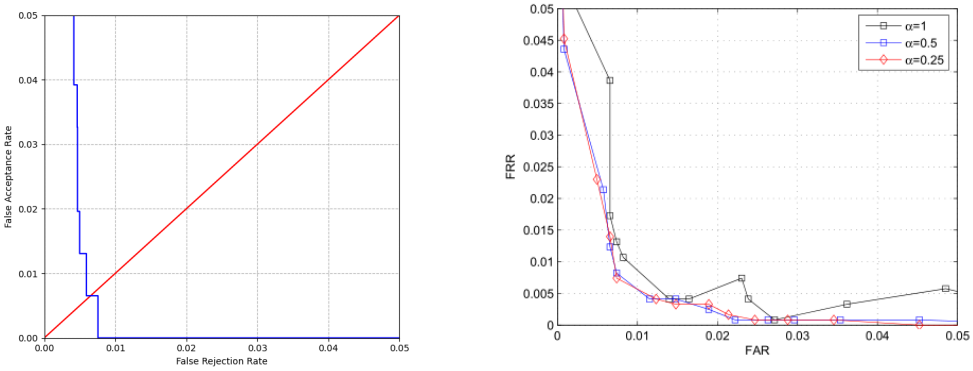


Figure 5.2: Average EER value for each α and number of components tested for the Essex Faces94 Database.

The EER value does not vary much with either α or the number of components, and is consistently very low, with multiple combinations of α and number of component features providing results near our best EER value of 0.0065, with 20 components and an $\alpha = 0.5$. This is consistent with the results in [SLM07] where the EER is below 0.01 but above 0.005. The exact value is not given, but a graph is used to represent the results. The

comparison is shown in Figure 5.3.



(a) Our implementation

(b) Implementation by Sutcu [SLM07]

Figure 5.3: Comparison of results for the ROC curve on the Essex94 Database.

This indicates that the dataset is an easy benchmark. This could be due to a high degree of inter-subject image similarity across samples, and large intra-subject differences, as shown in Figure 5.4. Both our and the comparison tests had EER below 0.01.

Figure 5.4: Sample of Essex94 Database images.

AR Face Database

As mentioned in Chapter 2, a subset of the AR Face Database was created for testing, consisting of 10 cropped images per subject for 26 subjects. At 25 PCA components, an $\alpha = 0.2$, we got an (EER) of 0.038, provided by a β value of 1.23. Despite coming from a selected subset of the original AR Dataset, these results have been provided for completeness sake.

AT&T Database of Faces

The AT&T Database of Faces' 40 subject-images were separated into a training, enrollment, and authentication set. As with the Essex 94 dataset, each subject contributed one random image to the training set and one other random image to the authentication set. The remaining 8 images were used for enrollment. Each of the 40 subjects provided an authentication attempt against each of the 40 subjects, for 1,600 total attempts. The tests were repeated 12 times, and the results averaged. These averaged EER, by α and number of components, are shown in Figure 5.5. The best EER value (0.072 at an $\alpha = 0.4$ and with 25 components) is slightly lower than the lowest EER in [SLM09], which is near 0.08. While the results do not vary much when $\alpha \leq 8$, it can be seen in [SLM09] that the EER worsens as the α increases outside this range.

We do as well, or better than, the other fuzzy extractor implementations in the literature on the two laboratory datasets for which we have found implementations, and do very well on the other publicly available datasets. From this we conclude that this implementation does not negatively affect the precision of the algorithm on laboratory datasets compared to other fuzzy extractor implementations. These results are promising, but the performance

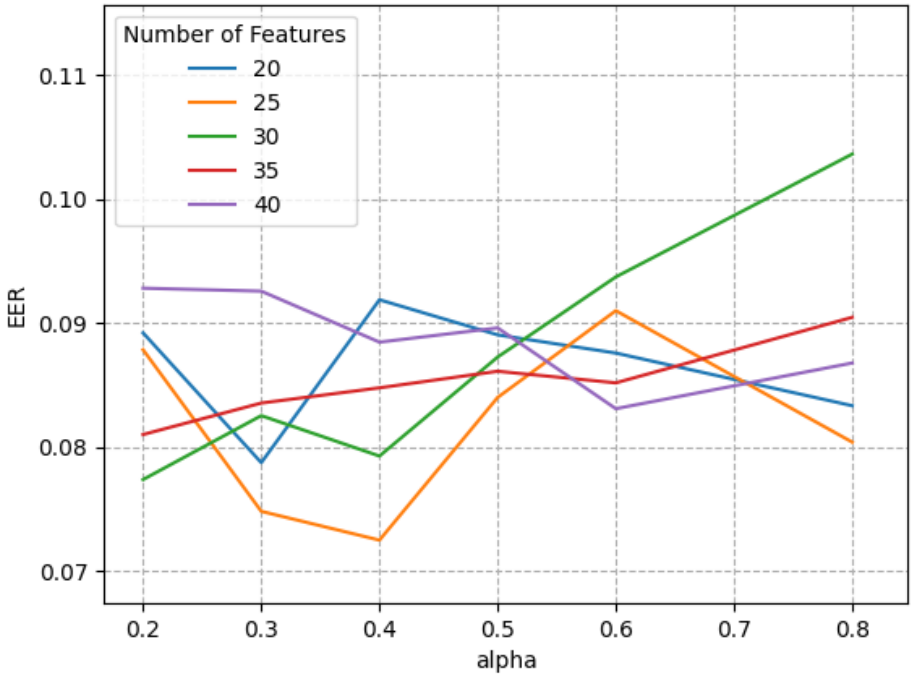


Figure 5.5: Average EER value for each α and number of components tested for the AT&T Database.

on uncontrolled settings by user-taken images, which tend to be quite variable, is likely to suffer.

Our Generated Dataset

Our generated dataset has a smaller number of subjects than some laboratory ones (59, compared to Essex 94's 153), but a larger number of image sets, grouped by source video. These can be used to create the semblance of multiple authentication attempts per user. When testing, one image from each video was taken for training and codebook generation purposes. A

further image was saved for later authentication testing. The rest of the images are used for user enrollment.

An authentication attempt is made on each enrollment set from each other subject in the dataset, which should fail. Each enrollment set also receives an attempt from the authentication image from the same source video, which should succeed.

We achieved an EER of 0.117 with 20 PCA components, $\alpha = 0.2$, and β near 1.43 (Figure 5.6). This is a higher rate than on the public deep datasets, but is explained by the less constrained nature of the testing dataset. For comparison, conventional PCA, on a dataset of manually adjusted phone images achieved an EER of 0.112 in [KSH12], and and EER of 0.124 on unprocessed images from the AT&T dataset in [DDW08].

We also tested the baseline fuzzy extractor algorithm implementation described by Sutcu et al. in [SLM09]. The algorithm achieved an EER of 0.164 (60 components, $\alpha = 0.6$), which is near our results.

For easy comparison, the four Receiver Operating Characteristic curves of our implementations on the four datasets are presented in Figure 5.7.

The increase in error between laboratory datasets and the generated ones is significant. This is, however, expected. As mentioned in Chapter 3, PCA is very sensitive to lighting differences, and to variation in face location. While our preprocessing steps partially address this, we were unable to fully correct these issues. Our results are similar to others using PCA as feature extractors on non laboratory datasets.

These error rates demonstrate the difficulty of doing facial recognition in even partially unconstrained contexts. This suggests that this is a successful proof

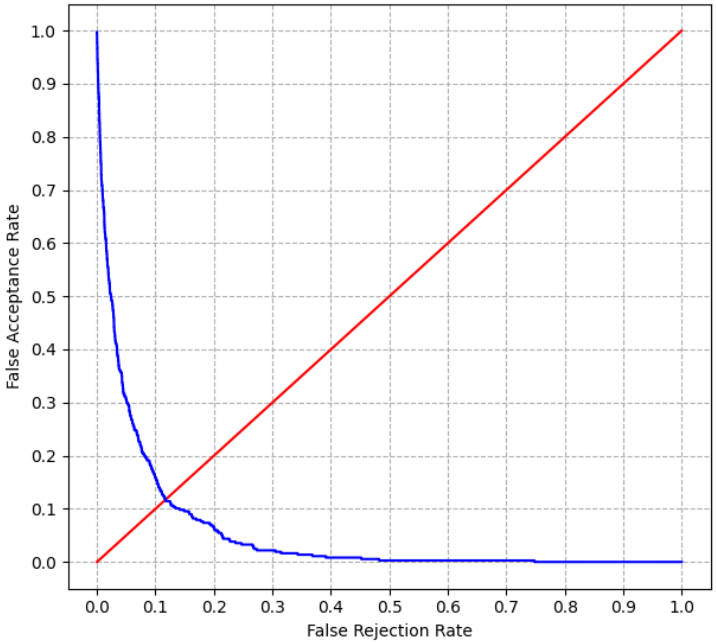


Figure 5.6: 11.7% Equal error rate; 20 components; $\alpha = 0.2$

of concept for using fuzzy extractors for privacy preserving authentication schemes. And further progress would be found with a better feature extractor and more elaborate preprocessing.

5.2.2 Further Testing

Since our dataset contains a large number of image-sets per subject we can expand beyond the regular method of testing biometric authentication with deep datasets. By using a set of enrollment images belonging to a certain video, but using different video sources of the same user, we can replicate authentication attempts across different image sources.

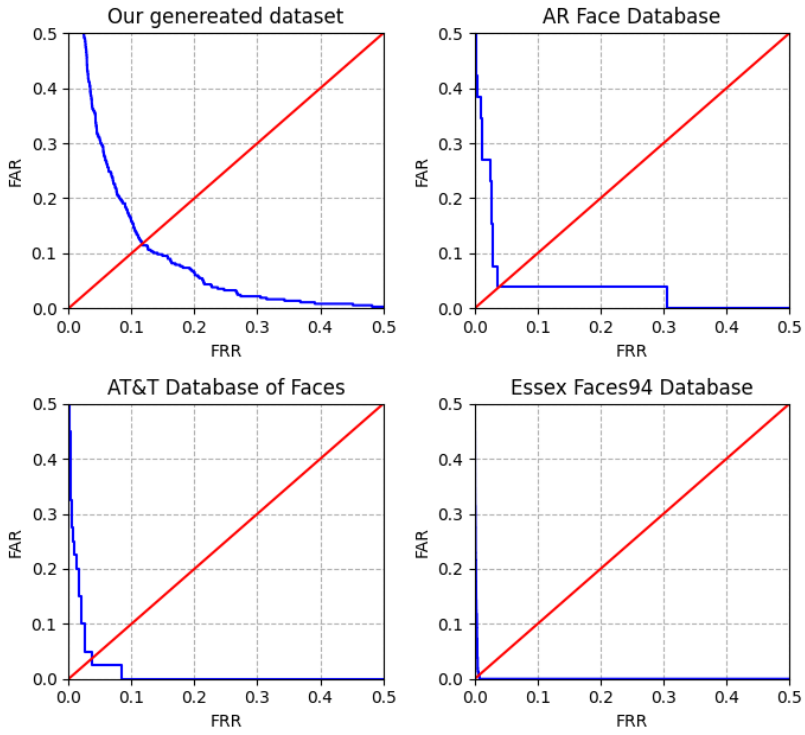


Figure 5.7: The algorithm performs well on each dataset, but with a higher error rate on the generated one.

Normally, when incorporating multiple images sources of the same subject, the images are combined into a single, highly variable, set. These aggregated sets can then be used normally for testing. By limiting the enrollment set to a single source, we can emulate the equivalent of a "very difficult" authentication challenge.

While our goal for the project was to demonstrate that fuzzy extractors can be used with non laboratory images meant to emulate real world usage, this could serve as a next step. In the most extreme case, this could emulate a one time enrollment while in an uncontrolled environment, followed by a random authentication attempt at a later time, in a different setting.

With these restrictions, our implementation reached an EER of 0.355, which is near the 0.34 of PCA on the AR Database with no preprocessing, in the literature [HPA02]. While very high, this is the first such attempt to our knowledge, and this provides the opportunity and direction for further research and improvement.

5.3 Performance Enhancement Attempts

5.3.1 Image Similarity Metric

When provided with a large sample of biometric images, we have found significant improvement to the error rate of our scheme when using the subset of images that are most self-similar (see Figure 5.8).

The methods used to find self similar images are based on the grid-like perceptual hashing method detailed in 4.4. We tested similarity metrics that more harshly penalized fewer large differences, such as summing the square of the differences, and those that instead penalized a larger number of differences, such as summing the square root of the differences. As seen in Figure 5.9, there were no meaningful differences between these methods, when used on the same original sets of image.

As it was impractical to ask for a large numbers of selfies, this image self-similarity measure wasn't added to the proof of concept for the system. In the future, it would be worth exploring the use of a process like the one used to create the testing dataset from videos. Automated face-image capture, on a mobile application, would allow for more numerous, higher quality selfies, and would be easy to use. This may even lead to the possibility of asking

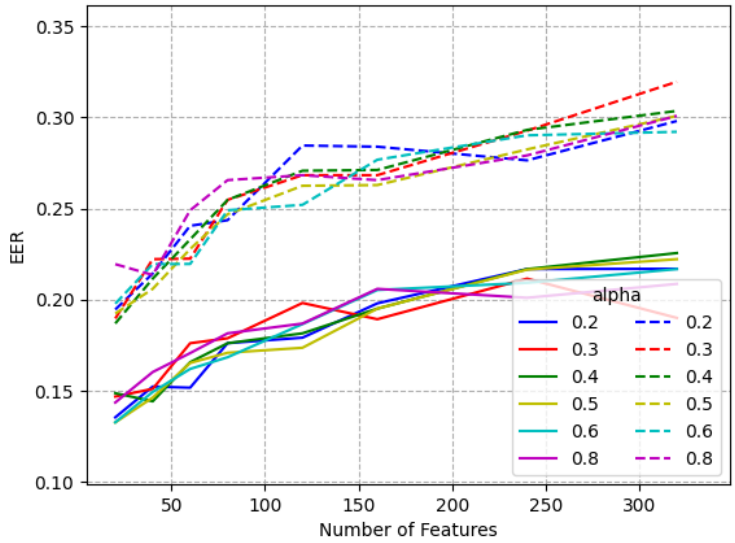


Figure 5.8: 12 most similar images (solid line) performs better than using the complete set (dashed line)

users for multiple quick bursts of selfies over a few days to really improve the quality of the authentication system.

5.3.2 Alternate Feature Extractors

As noted previously, the implemented scheme's fuzzy extractor component acts as a "wrapper" on top of a conventional feature extractor. For our project we used Principal Component Analysis[TP91] (also known as Eigenfaces) as our feature extractor for reasons of convenience. While PCA has a long history of use in facial biometric contexts, it has significant weaknesses. In particular, it is very sensitive to changes in lighting. While the feature extractor was meant to be replaced with advanced commercial or proprietary software after the proof of concept, we experimented with other methods

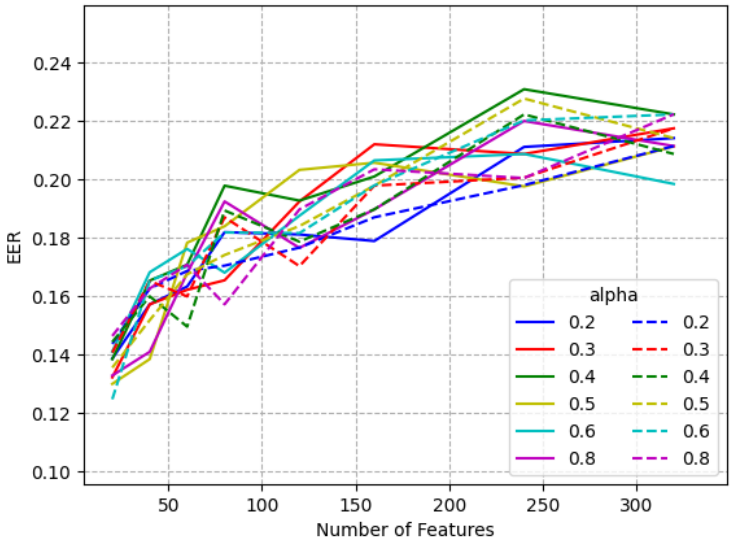
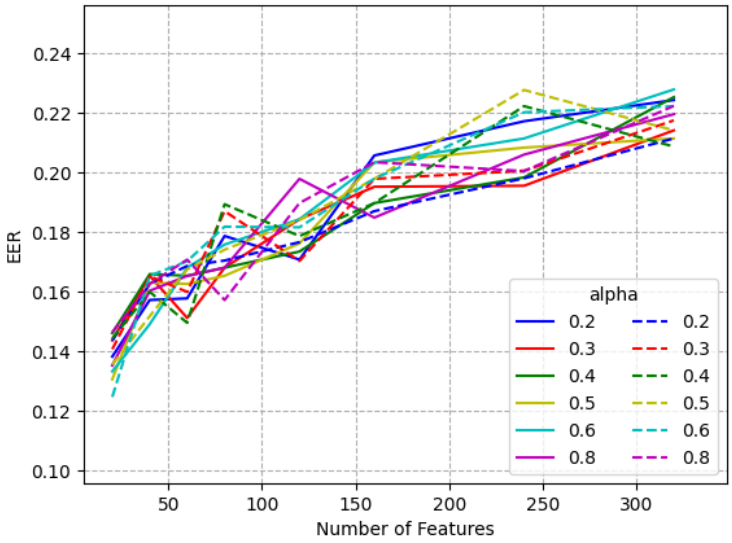
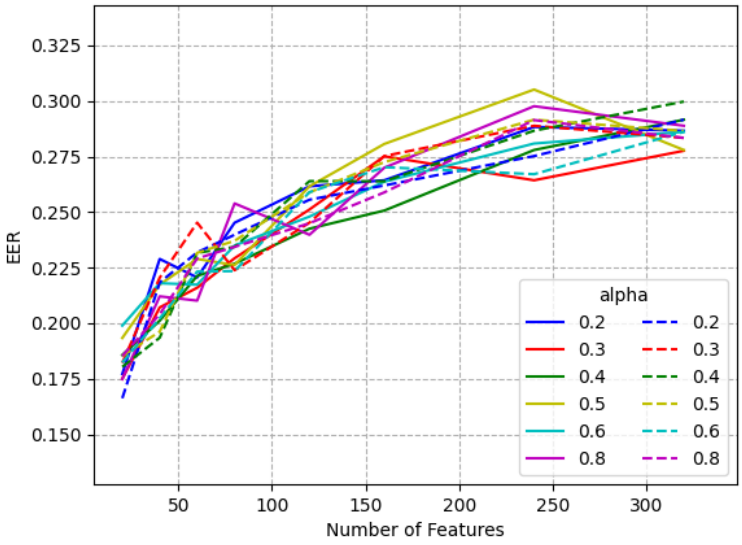


Figure 5.9: Images with large differences minimized (solid line) or maximized (dashed) perform similarly

of feature extraction. Of note, we also tested Kernel Principal Component Analysis (KPCA) as a feature extractor, as in [KJK02]. KPCA was used concurrently with PCA for much of the testing. During early testing with laboratory datasets, both feature extractors performed similarly. When testing on our generated dataset, KPCA did not perform meaningfully better than PCA and proved to be slower. This can be seen in Figure 5.10.



(a) Generated Dataset, Aligned



(b) Generated Dataset, Cropped

Figure 5.10: PCA (solid line) and KPCA (dashed line) perform similarly as feature extractors

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis was built in two parts. In the first, we implemented a *key-generating quantization-based fuzzy extractor* scheme for facial feature biometrics based on the work by Dodis et al., Li et al., and Sutcu et al. in [DRS04, LSM06, SLM07, SLM09]. With the modified scheme:

1. We increased user privacy in the cases where a malicious actor may gain access to the user's biometric-derived quantizing *codebook*. This involved the new codeword construction that consisted of a single *generating codeword* g_ij and the *user step* s_ij. This prevents the re-discovery of the midpoint codewords by brute-forcing the possible combinations when having access to the helper data.

2. We addressed some implementation-based issues, such as floating point errors, and codebook alignment issues. The former required a change

in the way we calculate the *biometric key*, and the latter required the removal of the sketch from the pre-quantization authentication features.

3. We also implemented rudimentary preprocessing and tested our implementation on the *Essex Faces94* dataset used in [LSM06], where we achieved similar results. This goes to show that our implementation does not significantly affect the scheme's performance, while providing additional protection against malicious actors that may gain access to the user codebooks stored on the server and tailoring the scheme for our use - implementation and testing of the scheme on data meant to emulate real world use.

Therefore, this thesis provides a jumping off point for using fuzzy extractors for faces in real world contexts, though with a more advanced feature extractor and preprocessing algorithm.

The second part consists of the creation of a process to automate most of the generation of deep datasets suitable for the testing of similar schemes from the YouTube [YT] video platform.

This process required a curated list of YouTube channels where the main subject was the most common person in the frames. The process would extract facial images from the videos posted to that YouTube channel when said video was deemed likely to contain images that matched the images we expected to receive during real world use - images that contained faces, where the subject was front-facing, images taken from mobile phones or tripod cameras, with backgrounds that were not necessarily optimized for easy recognition. The extracted images were grouped by subject and preprocessed to generate a dataset that would enable the testing of the implemented scheme described above.

The process led to the creation of an example deep dataset larger in size than those available for free online for minimal work, and showed that these datasets can be further expanded with only little additional work: only requiring more YouTube channels. This larger dataset allowed for a recognition challenge similar to conventional deep datasets, using either a single video-set per user, or combining all images from the same user into a single set. This also enabled us to show that in the conventional case, our implementation performed like similar non-commercial schemes.

In addition, the generated dataset also allowed for a more difficult challenge, by using a single video-set for enrollment images and using authentication images exclusively from different video-sets. While accuracy is reduced in the difficult challenge, it provides a higher benchmarks for the future. Further refinement would be necessary if this is to be compared to commercial applications.

As seen in Chapter 2, while fuzzy extractors are often used in conjunction with fingerprints or with iris scans, they are less frequently used with face biometrics. Some of the literature, as late as 2017, still conducted some tests on the laboratory datasets such as Essex Faces94. It is possible that the difficulty in finding representative datasets may stifle the research of fuzzy extractors, specifically when working with facial biometrics.

This dataset generation process allows for the creation of such representative samples, and with a fair degree of control over the resulting dataset, through the list of input channels. This could allow further testing of fuzzy extractor for faces.

In addition, this provides a baseline for individuals and small groups to generate their own datasets, with little effort, in ways that may more closely

match their use cases than existing, publicly available datasets.

6.2 Future Work

In sum, we were able to show that fuzzy extractor schemes do have potential for use in real world contexts, and that it is possible to create less constrained deep datasets from content available online. However, there is more to be done for commercial viability.

Here we detail further research that may lead to interesting results, split according to the part of the overarching project they relate to.

Biometric Representation

- We used Principal Component Analysis (PCA) as our feature extractor for ease of implementation, and because it allowed us to compare our implementation to other fuzzy extractor based schemes with the same feature extractor. However, there are stronger feature extractors in terms of accuracy. The next steps would involve using stronger feature extractors with the same fuzzy extractor "wrapper".
- Similarly, there is much room for improvement with our preprocessing. It addressed some of the possible causes for errors, but not all. Notably, one that can address directional light variation would be useful.
- There could also be the use of different representations of the biometric identifiers, for example, we could test with infrared cameras, or using three dimensional representation of faces. This would, however, require new datasets.

- It could also be possible to change the scheme to work with short videos rather than on discrete images. This could make image acquisition at enrollment easier, but would require a new process for the dataset generation.

Image Acquisition

- The image acquisition process could be overhauled. The current version processes images in real time, as the video runs in the browser. This is the by-product of the series of circumstances the project has been through. A possible future implementation would scrape the channels as we do, then download the videos to do the face detection and image winnowing offline. This would increase the freedom when extracting, comparing, and manipulating images.

- Using a face location tracker could help determine whether the subject in consecutive frames is the same person. This could supplement or replace the strike system and may prove to be more reliable. This could also be expanded to generate a dataset of short video clips of people.

- With access to a face tracker and offline processing, it would be interesting to explore the selection of frames in ways that maximize the differences between faces. This could be done by collecting the faces most different from others within an interval, or the face most different from the last collected face. This could increase the variation between images for each subject.

- The process, if further automated and made stricter, could be given a significantly larger number of channels. Using inter-video information,

it may be possible to flag channels as problematic and either discard them, or request manual review.

Dataset Processing

- It might be worth exploring different detectors. Convolutional neural networks are often used for image recognition tasks. These tend to be slower than the Haar Cascade and Histogram of Oriented Gradients detectors we had access to, but are more advanced and, depending on the training data and effort, can perform better. If processing the videos offline, this would be a very interesting avenue to explore. Interestingly, we deviate from common models by wanting to avoid the detection of profile, occluded, or heavily distorted faces.

- We could also use facial landmarks for stricter filtering. Landmarks could provide points of interest where abrupt differences in shading or colour could indicate partial occlusions, such as microphones.

- We may be able to generate a better dataset by using different post-processing on the extracted images. Specifically, our perceptual hashing algorithm could be improved. Comparing each region to a weighted average of the regions around it could prove to be interesting. This could address more localized changes in lighting, allowing for a better estimation of similarity between faces in images.

- Instead of the current approach which finds images closest to the average image in the set, clustering algorithms could be used on the hashed images to generate sets of very similar images. Care would have to be taken to ensure low variability issues would not arise again.

Bibliography

[arc] `archive.org`.

[BDHV07] 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 '07, page 353–355, New York, NY, USA, 2007. Association for Computing Machinery.

[BGP13] Joan-Isaac Biel and Daniel Gatica-Perez. The youtube lens: Crowdsourced personality impressions and audiovisual analysis of vlogs. *Multimedia, IEEE Transactions on*, 15(1):41–55, 2013.

[BTA] R. Brien, A. A. Tambay, and C. Adams. Improved fuzzy extractor for faces. (paper submitted in April 2020).

[CL08] T Cootes and A Lanitis. The fg-net aging database, 2008.

[Das17] Ashok Kumar Das. A secure and effective biometric-based user authentication scheme for wireless sensor networks using smart card and fuzzy extractor. *International Journal of Communication Systems*, 30(1):e2933, 2017.

[DDW08] MA Dabbah, SS Dlay, and Wai Lok Woo. Pca authentication of facial biometric in the secure randomized mapping domain. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–5. IEEE, 2008.

[DFM98] G. I. Davida, Y. Frankel, and B. J. Matt. On enabling secure applications through off-line biometric identification. In *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No.98CB36186)*, pages 148–157, May 1998.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 523–540, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[EM17] M. M. Eid and M. A. Mohamed. A secure multimodal authentication system based on chaos cryptography and fuzzy fusion of iris and face. In *2017 Intl Conf on Advanced Control Circuits Systems (ACCS) Systems 2017 Intl Conf on New Paradigms in Electronics Information Technology (PEIT)*, pages 163–171, 2017.

[FAD06] Feng Hao, R. Anderson, and J. Daugman. Combining crypto with biometrics effectively. *IEEE Transactions on Computers*, 55(9):1081–1088, 2006.

[Far94] Leslie G Farkas. *Anthropometry of the Head and Face*. Raven Pr, 1994.

[GBK97] A Georghiades, P Belhumeur, and D Kriegman. Yale face database. *Center for computational Vision and Control at Yale University, http://cvc. yale. edu/projects/yalefaces/yalefa*, 2(6):33, 1997.

[GBK01] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[GE16] Karthi Govindharaju and M. Ezhilarasan. Securing biometric template using a hybrid scheme. In *Proceedings of the International Conference on Informatics and Analytics*, ICIA-16, New York, NY, USA, 2016. Association for Computing Machinery.

[GZH+16] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. *CoRR*, abs/1607.08221, 2016.

[HMBLM08] Gary B. Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*, Marseille, France, October 2008. Erik Learned-Miller and Andras Ferencz and Frédéric Jurie.

[HPA02] Thomas Heseltine, Nick Pears, and Jim Austin. Evaluation of image preprocessing techniques for eigenface-based face recognition. In *Second International Conference on Image and Graphics*, volume 4875, pages 677–685. International Society for Optics and Photonics, 2002.

[Jol] Mitchell Jolly. https://www.kaggle.com/datasnaek/youtube-new.

[JRP04] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Trans. Cir. and Sys. for Video Technol.*, 14(1):4–20, January 2004.

[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.

[Kag] https://www.kaggle.com/.

[Kin09] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[KJK02] Kwang In Kim, Keechul Jung, and Hang Joon Kim. Face recognition using kernel principal component analysis. *IEEE signal processing letters*, 9(2):40–42, 2002.

[KSH12] Emir Kremic, Abdulhamit Subasi, and Kemal Hajdarevic. Face recognition implementation for client server mobile application using pca. In *Proceedings of the ITI 2012 34th International Conference on Information Technology Interfaces*, pages 435–440. IEEE, 2012.

[KSSMB16] Ira Kemelmacher-Shlizerman, Steven M. Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces

for recognition at scale. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[LGM+17] N. Li, F. Guo, Y. Mu, W. Susilo, and S. Nepal. Fuzzy extractors for biometric identification. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 667–677, 2017.

[lHH12] R. Álvarez Mariño, F. Hernández Álvarez, and L. Hernández Encinas. A crypto-biometric scheme based on iris-templates with fuzzy extractors. *Information Sciences*, 195:91 – 102, 2012.

[LNB+18] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari. A robust ecc-based provable secure authentication protocol with privacy preserving for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 14(8):3599–3609, 2018.

[LSM06] Qiming Li, Yagiz Sutcu, and Nasir Memon. Secure sketch for biometric templates. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, pages 99–113, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[MB98] AM Martinez and R Benavente. The ar face database, cvc, univ. autonoma barcelona, barcelona. *Technical report, Spain, Technical Report 24*, 1998.

[MBPVG14] Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. In *European conference on computer vision*, pages 720–735. Springer, 2014.

[MKR17] Kathleen Moriarty, Burt Kaliski, and Andreas Rusch. Pkcs# 5: Password-based cryptography specification version 2.1. *Internet Engineering Task Force (IETF)*, 2017.

[MRFS19] Michele Merler, Nalini K. Ratha, Rogério Schmidt Feris, and John R. Smith. Diversity in faces. *CoRR*, abs/1901.10436, 2019.

[MS17] Anup Kumar Maurya and Vinjamuri Narsimha Sastry. Fuzzy extractor and elliptic curve based efficient user authentication protocol for wireless sensor networks and internet of things. *Information*, 8(4):136, 2017.

[MY19] Alzahraa J Mohammed and Ali A Yassin. Efficient and flexible multi-factor authentication protocol based on fuzzy extractor of administrator's fingerprint and smart mobile device. *Cryptography*, 3(3):24, 2019.

[NW14] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE international conference on image processing (ICIP)*, pages 343–347. IEEE, 2014.

[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.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RCB01] N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, 2001.

[SH] F Samaria and A Harter. The orl database of faces.

[SJ19] Y. Shi and A. K. Jain. Docface+: Id document to selfie matching. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 1(1):56–67, 2019.

[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.

[SLM07] Yagiz Sutcu, Qiming Li, and Nasir Memon. How to protect biometric templates, 2007.

[SLM09] Yagiz Sutcu, Qiming Li, and Nasir Memon. Design and analysis of fuzzy extractors for faces, 2009.

[Spa94] L Spacek. Faces 96: The university of essex. *Face recognition data set. Available*, 7, 1994.

[Spa07] Libor Spacek. Collection of facial images: Faces94. *Computer Vision Science and Research Projects, University of Essex, United Kingdom, http://cswww. essex. ac. uk/mv/allfaces/faces94. html*, 2007.

[Spa18] L Spacek. Faces95 database. *URL: http://cswww.essex.ac.uk/mv/allfaces/faces95.html (accessed 01.07. 2017)*, 2018.

[SSM05] Yagiz Sutcu, Husrev Taha Sencar, and Nasir Memon. A secure biometric authentication scheme based on robust hashing. In *Proceedings of the 7th Workshop on Multimedia and Security*, MM&Sec '05, pages 111–116, New York, NY, USA, 2005. ACM.

[TN05] Andrew BJ Teoh and David CL Ngo. Cancellable biometerics featuring with tokenised random number. *Pattern Recognition Letters*, 26(10):1454–1460, 2005.

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

[TSF+15] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 1(8), 2015.

[vin] Archived at `vine.co`.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, Dec 2001.

[VJ04] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, May 2004.

[WHM11] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534, 2011.

[YT] `https://www.youtube.com/`.

[YT8] `https://research.google.com/youtube8m/explore.html`.

[YTh] `https://support.google.com/youtube/answer/6180214?hl=en&ref_topic=9257109`.

[ZZZ15] Min Zhang, Jiashu Zhang, and Ying Zhang. Remote three-factor authentication scheme based on fuzzy extractors. *Security and Communication Networks*, 8(4):682–693, 2015.