BioWallet: cold wallet proposal using a biometric cryptosystem

Pasquale Celani

A.Y 2024/2025

1 Introduction

Nowadays, the topic of cryptocurrencies is rapidly gaining popularity with Bitcoin leading the way. Bitcoin is a decentralized peer-to-peer cryptocurrency designed to offer an alternative to the highly centralized traditional banking system and conventional methods of currency production. Apart from the decentralization one of the most relevant properties of bitcoin is that is pseudo-anonymous. Thus mean that in Bitcoin you can make financial transaction without providing your real identity. Indeed, in Bitcoin there is no concept of traditional accounts or personal identities. Instead, the system relies on cryptographic keys that represent a user's identity. More specifically, Bitcoin uses a pair of cryptographic keys one public and one private as part of a digital signature scheme. These keys are used to sign transactions and indicate where the coins are coming from and where they're being sent.

These keys are managed trough apposite system called wallets. Currently there are various types of cryptocurrency wallets, including paper wallets, brain wallets, hardware wallets, and software wallets. One of the main challenges with wallets is determining the most secure way to store private keys. If a private key is lost or stolen the associated funds are permanently inaccessible, with no way to recover them. The challenge of securely storing private keys is not new in the field of security, and various methods have been researched to address it. One such approach involves leveraging an individual's biometric traits. This document presents a proposal for a new type of software wallet that leverages biometric facial recognition to generate and recover cryptographic keys eliminating the need to remember passwords or codes. Access is granted simply by showing one's face to the camera and proving your identity.

In the following sections of this document, we will first review the essential background knowledge needed to understand the topic, followed by a detailed explanation of the methodology used during the system's experimental evaluation, the implementation process, the results obtained, and finally, the conclusions drawn from the study of this project.

To conclude this section, a brief overview of related work in the field of biometric cryptosystems will be provided. The approach used in this project follows a key-based cryptosystem model known as key release, as defined in the survey conducted by Prabhjot Kaur et al [6]. In this key construction mechanism, the biometric template and key is stored together as two separate products, and the key is released upon successful verification. Other methods of key based system could be Key binding or key generation. In the first approach a key is monolithically bound to the biometric template, whereas in the second approach the key is directly derived from a biometric trait. For example of a key generation mechanism is presented in the study by Cascone Aldo [3], which aims to generate a cryptographic key from a facial template. The method uses Reed-Solomon error correction to address fuzzy values, along with quality assessment techniques based on pose and illumination to ensure the probe maintains the highest possible quality.

As highlighted in the survey by Prabhjot Kaur et al. and the study by

Cascone these systems have limitations. One of the main issues in biometric crypo systems is the variability of biometric probes, which can be influenced by factors such as pose, illumination, expression, and age (A-PIE). Specifically, in image-based approaches like face recognition. Indeed, even a slight variation of just 1 bit in the template could lead the key generation process to return different key. Additionally, error correction techniques have their limits in terms of how much they can correct. This results led this project research to investigate alternatives methods such as a biometric cryposystem based on a key release mechanism which is still not very researched and see if it could lead to good results, that means maintain a false acceptance rate (FAR) of 0 without limitations in a specific context. As we will see, the answer is yes, through a combination of quality assessment processes and the use of deep learning models such as FaceNet512 and RetinaFace.

2 Background

In this section we will focus on the concepts of biometric systems and biometric cryptosystems, explaining the key principles behind them. Topics such as Bitcoin and distributed ledger technologies like blockchain will not be covered here, as the primary focus of this project is to explore the feasibility of the key generation and retrieval using face recognition.

Therefore, the first step is to clarify the key structure used in Bitcoin as this defines what our system needs to generate. Bitcoin uses an asymmetric key schema based on Elliptic Curve Cryptography (ECC), which more in detail it uses the secp256k1 elliptic curve. Where based on [11] an Elliptic Curve is a curve given by an equation of the form $y^2 = x^3 + Ax + B$ where the discriminant satisfy the following constraint $\Delta = 4A^3 + 27B^2 \neq 0$. This last constraint is basically stating that the given polynomial has distinct roots ensuring the curve is not singolar. Secp256k1 is an elliptic curve of the following form $y^2 = x^3 + 7$. The graphical representation of the curve is illustrated in Figure 1. As can be observed Secp256k1 is an elliptic curve of the general form, with parameters $A = 0$ and $B = 7$. However, A and B are not the only parameters defining Secp256k1. The curve also includes the following additional parameters [1]:

- $n = 115792089237316195423570985008687907852837564279074904382605163141518161494337$, the order of the elliptic curve group;
- $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$, the order of the finite field which the curve is based on;
- $G_x = 55066263022277343669578718895168534326250603453777594175500187360389116729240$, generator point of the curve for the x component;
- $G_y = 32670510020758816978083085130507043184471273380659243275938904335757337482424$, generator point of the curve for the y component.

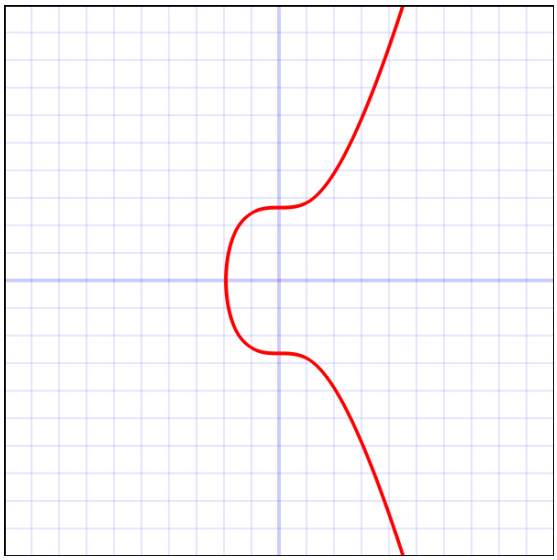

Figure 1: This is a graph of secp256k1's elliptic curve $y^2 = x^3 + 7$ over the real numbers. *Image taken from:* https://en.bitcoin.it/wiki/File:Secp256k1.png

With this information we can now define the concepts of private and public keys. A private key in Elliptic Curve Cryptography is defined as $S_k \in \{d \in \mathbb{N} : d \geq 1 \wedge d < n - 1\}$ and is randomly chosen. Concretely, it is a randomly generated 256-bit value. The corresponding public key is computed through scalar multiplication $P_k = S_k \cdot G \mod p$ where $G = (G_x, G_y)$ is the generator point on the elliptic curve. The result is a point on the curve, but for our purposes we consider the compressed form of the public key $P_k = \text{prefix} + (P_k)_x$ where the prefix is "$x02$" if $(P_k)_y$ is even and "$x03$" otherwise. Finally, both the private and public keys are ultimately represented in hexadecimal format.

The reason for choosing facial recognition lies in its high universality, collectability and most importantly its strong user acceptability, as shown in Table 1 Acceptability is a crucial factor for a wallet system as users need to feel comfortable and willing to entrust the system with their funds. That said, facial recognition is not among the most reliable biometric traits due to intraclass variations caused by pose, illumination, expression and aging (PIE-A). However, in our context these limitations are less critical. The system will typically be used in controlled indoor environments and the user is expected to cooperate, after all, access to their funds depends on it. More specifically, the biometric template got from face considered is the embedding obtained from the final layer of the used deep neural networks just before the classification stage. This embedding is represented as an array of numerical values.

As mentioned in section 1, the chosen approach for the Biometric Cryptosystem (BCS) is the key release method as defined in the survey study by Prabhjot

Biometric traits	Universality	Uniqueness	Permanence	Collectability	Performance	Acceptability	Circumvention
Fingerprint	Moderate	High	High	Moderate	High	Moderate	High
Face	High	Low	Moderate	High	Low	High	Moderate
Iris	High	High	High	Moderate	High	Moderate	Low
Heartbeat	High	High	High	Moderate	High	Moderate	Low
Veins	Moderate	Moderate	Moderate	Moderate	Moderate	Moderate	Low
Ear	High	High	Moderate	Moderate	Moderate	Moderate	Moderate
Hand geometry	Moderate	Moderate	Moderate	High	Moderate	Moderate	Moderate
Retina	High	High	Moderate	Low	High	Moderate	Low
Signature	Low	Low	Moderate	High	Moderate	Moderate	High
Gait	Moderate	Low	Low	High	Low	Moderate	Moderate
Voice	Moderate	Low	Low	Moderate	Low	High	High
Keystroke & Mouse	Low	Low	Low	Moderate	Low	Moderate	Moderate

Table 1: Comparison of various biometric traits based on different evaluation criteria. [7]

Kaur et al. The core idea follows the classic biometric verification procedure with a few twists. The first key difference lies in the enrollment phase where not only are the biometric templates stored in the gallery but one or more cryptographic keys are also generated and stored in it. These keys are not directly bound to the biometric templates, instead, they are stored alongside the user's identity within the gallery. Upon successful verification the corresponding key/s are released and returned to the user. One of the main advantages of this approach is its tolerance to slight variations in biometric data. Since it relies on a threshold-based verification mechanism, it is more resilient to minor changes in input related to PIE-A. Another benefit is the flexibility to generate multiple keys, which is often a requirement for wallet applications. Additionally, the keys can be generated entirely at random without requiring the user to remember any passcode, since they are stored and retrieved through the biometric gallery. An overview of this approach is illustrated in the following Figure 2.

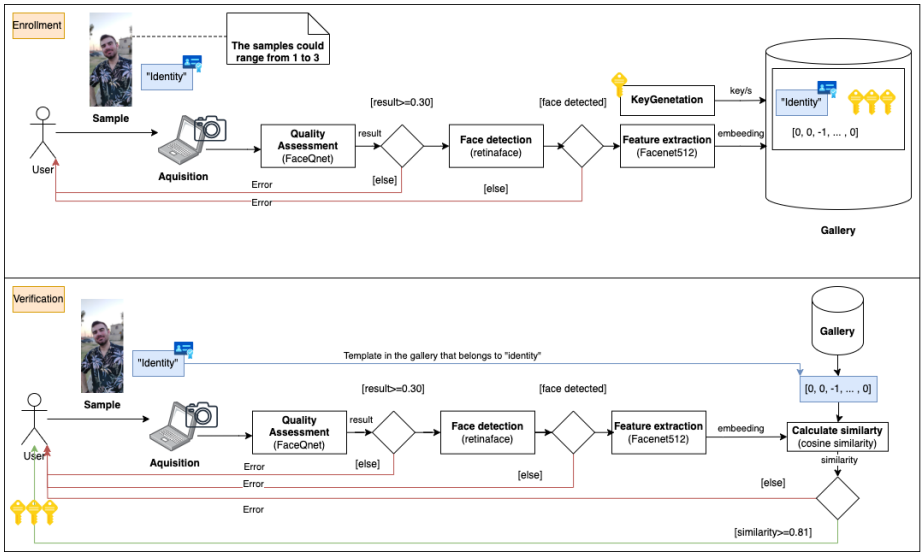

Figure 2: Overview process for enrollment and verification in BioWallet.

As illustrated in the Figure 2 the core components quality assessment, face detection and feature extraction rely on deep learning models such as FaceQnet [5], RetinaFace [4] and FaceNet512 [10]. As will be explored in section 3 and section 5 FaceNet512 is not the only deep network used for feature extraction in this study. In fact, additional models such as DLib and DeepID [12] were also evaluated. Now, let's briefly introduce the main deep learning models used in this study.

RetinaFace is a single-stage face detection framework that performs dense face localisation which is a broader definition of face detection, that more in detail includes face detection, face alignment, pixel wise face parsing and 3D dense correspondence regression. RetinaFace is built upon a feature pyramid network (FPN) for detecting faces at different scales using a ResNet backbone, where features from multiple layers of the network are taken from the lowest to highest resolution and then are combined in a pyramid feature. Then, the pyramid feature is passed to a context cascade module which it's objective is to increase the receptive field and enhance the rigid context modelling power. Then, following the context module the multi-task loss function is calculated for each anchor. The multi-task loss try to minimize 4 different levels which are the following:

- Face classification loss $\mathcal{L}_{cls}(p_i, p_i^*)$: A SoftMax loss for binary classes (face or not faces) , where p_i is the predicted probability of anchor i being a face and p_i^* is 1 for the positive anchor and 0 for the negative anchor.

- Face box regression loss $\mathcal{L}_{box}(t_i, t_i^*)$: Regression loss for the bounding box where $t_i = \{t_x, t_y, t_w, t_h\}_i$ and $t_i^* = \{t_x^*, t_y^*, t_w^*, t_h^*\}_i$ represent the coordinates of the predicted box and ground-truth box associated with the positive anchor.

- Facial landmark regression loss $\mathcal{L}_{pts}(l_i, l_i^*)$: Regression loss of 5 facial landmarks (2 eyes, nose and 2 corners of the mouth) where $l_i = \{l_{x1}, l_{y1}, \ldots, l_{x5}, l_{y5}\}_i$ and $l_i^* = \{l_{x1}^*, l_{y1}^*, ..., l_{x5}^*, l_{y5}^*\}_i$ represent the predicted five facial landmarks and ground- truth associated with the positive anchor.

- Dense regression loss $\mathcal{L}_{pixel}$: In this case a 2D Mesh face is constructed employing an efficient differentiable 3D mesh renderer to project a colored mesh $\mathcal{D}_{P_{ST}}$ onto a 2D image plane with camera parameters $P_{cam} = [x_c, y_c, z_c, x_c', y_c', z_c', f_c]$ (i.e. camera location, camera pose and focal length) and illumination parameters $P_{ill} = [x_l, y_l, z_l, r_l, g_l, b_l, r_a, g_a, b_a]$ (i.e. location of point light source, colour values and colour of ambient lighting). Once the rendered 2D face $\mathcal{R}(\mathcal{D}_{P_{ST}}, P_{cam}, P_{ill})$ is got it will be compared pixel wise difference of the rendered and the original 2D Face using $\mathcal{L}_{pixel}$ and the Dense regression loss is minimized.

  - $\mathcal{L}_{pixel} = \frac{1}{W*H} \sum_{i}^{W} \sum_{j}^{H} \parallel \mathcal{R}(\mathcal{D}_{P_{ST}}, P_{cam}, P_{ill})_{i,j} - I_{i,j}^* \parallel_1$, where W is the width, H is the height and $I_{i,j}^*$ the anchor crop.

RetinaFace as stated in [4] archives state of the art performance in the face detection task, indeed, it was one of the main reason that RetinaFace was chosen for the implementation of this study project. In the Figure 3 is shown a general overview of RetinaFace.

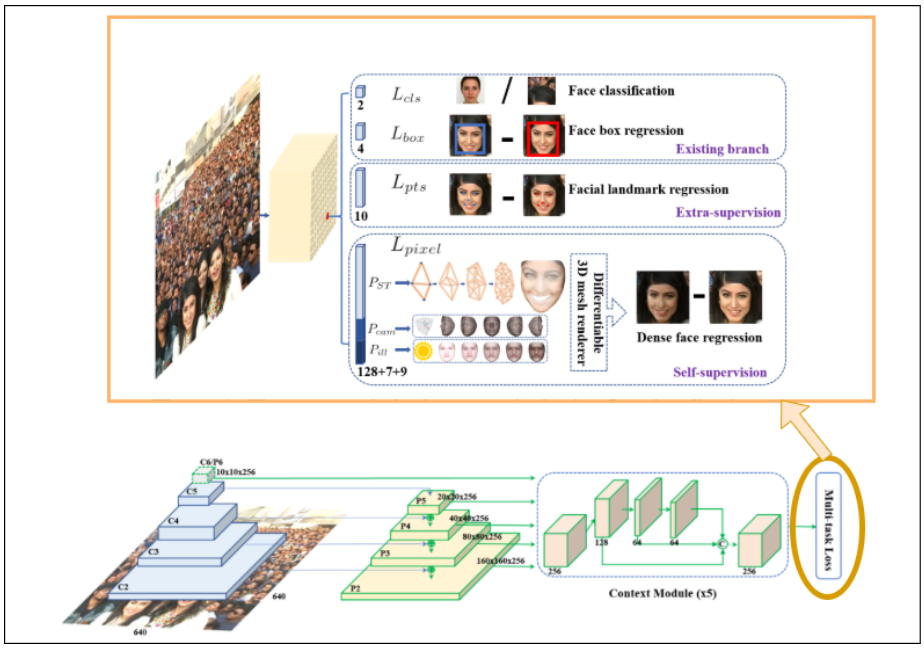

Figure 3: Retinaface overview. *Images taken from:* [4]

Now that a brief overview of RetinaFace have been provided it's time to take a closer look at FaceNet. FaceNet is a unified system for face verification, recognition and clustering based on deep convolutional neural network (CNN). The input of the CNN uses a patch based approach where the initial image is subdived into batches. The CNN is trained to learn a Euclidean embedding such that the squared L2 distances in the embedding space directly correspond to face similarity. Where the L2 is a normalization technique where given the L2 norm $\| x \|_2 = \sqrt{\sum x_i^2}$, each element x_i of the vector x is normalized as $\frac{x_i}{\|x\|_2}$, so we will get that $\| x \|_2 = 1$. The L2 norm calculates the distance of the vector coordinate from the origin of the vector space. One of FaceNet's major strengths is its ability to generate embeddings that are highly discriminative keeping embeddings from the same individual close together, while pushing those from different individuals further apart. To achieve this FaceNet utilizes a loss function known as triplet loss. The high level idea of triplet loss is to keep anchor image to be closer to positive images (images of the same individual) as compared to negative images. Put simply, the goal is to ensure that the distance between the embedding of an anchor image and that of a positive image is smaller than the distance between the anchor and a negative image.

Formally the loos function to be minimized is defined as:

$$\mathcal{L} = \sum_{i}^{N} [\| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + a]_+$$

Where:

- $f(x) \in \mathbb{R}^d$ is the embeeding of the image x into a d-dimensional space;
- N the cardinality of the training set;
- $\| f(x) \|_2 = 1$ (constraint embedding to live on the d-dimensional hypersphere);
- x_i^a is the anchor of a specific person closer to all other positive images x_i^p than it is to any image x_i^n of any other person. Building on the previous explanation of triplet loss we can formally express the objective as the following constraint:

  $$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathcal{T} \quad \| f(x_i^a) - f(x_i^p) \|_2^2 + a \; < \| f(x_i^a) - f(x_i^n) \|_2^2$$

  Where $\mathcal{T}$ is the set of all possible triplets, and a is a margin that is enforced between positive and negative pairs.

As can be noted in the last point the constraint applies to a set of triplets for each element. However, many image triplets may already satisfy this condition meaning the model won't learn much from them leading to slower convergence. To improve training efficiency it is essential to select triplets that violate the constraint as these are the most useful in the learning procedure. Ideally, the negative image should be as close as possible to the anchor, while the positive image should be as far as possible. This process, known as triplet selection, can be performed either offline precomputing before training or online dynamically selecteing during training using mini-batches for computational efficiency. The Figure 4 provides a visual summary of the concepts discussed regarding FaceNet.

Now that we have covered RetinaFace, FaceNet, the key generation process and the foundational concepts of biometric cryptosystems we will conclude this section by introducing FaceQNet v1. FaceQNet v1 is a crucial component of our system, responsible for assessing the quality of incoming facial samples, both during enrollment and verification. Its primary goal is to ensure that only high-quality samples are processed helping to reduce the risk of false rejections and false acceptances caused by low-quality samples subjected to high intraclass PIE-A variations. FaceQnet v1 is a novel open-source face quality assessment tool, inspired and powered by deep learning technology, which assigns a scalar quality measure to facial images, as prediction of their recognition accuracy. The first important point is how the groundtruth is generated.

The first step is to use an automated tool such as the BioLab framework to automatically label the dateset images with their quality. BioLab framework outputs a score between 0 and 100 for each one of its 30 individual ICAO

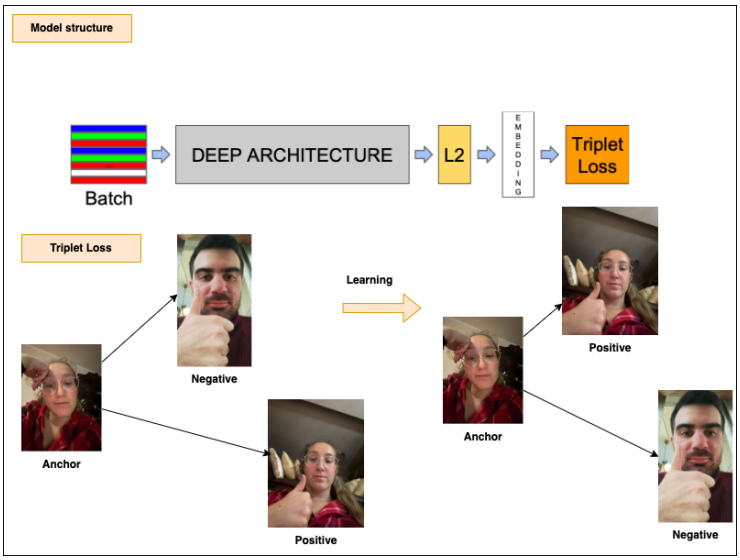

Figure 4: visual summary of the concepts discussed regarding FaceNet. *Image for model structure taken from:* [10]

compliance tests, which is defined as the quality reference for face. However, not all of these tests have the same relevance for face recognition, so a subset of them have been selected and then computed into a final averaged global ICAO compliance value. More specifically, the following test have been selected: blur level, too dark/light illumination, pixelation, heterogeneous background, roll/pitch/yaw levels, hat/cap presence, use of glasses and presence of shadows. The system as dataset uses VGGFace 2 [2] composed of 300 subject containing 3.31 million images of 9,131 different individuals, with an average of 362.6 images for each subject. The images were acquired under unconstrained conditions and present large variations in pose, age, illumination, etc.

Now, for each subject in the training set are selected the images with the highest ICAO compliance value as the reference image, and the rest are used as probe images, this is done since recognition scores depend on two samples being compared. Then, each probe image in the training dateset is given to the FaceNet, DeepSight and Dlib face recognition models to extract the three different embeeding as a 128-dimensional feature vectors. Then, using the embeddings generated by the face recognition models, the Euclidean distance is calculated between each reference image and all other samples belonging to the same subject. These distances quantify the dissimilarity between each probe image and its corresponding ICAO-compliant reference template. As a result, we obtained three distinct mated distances for each image pair. The idea is to assume that the selected reference image represents perfect quality due to its low intraclass variability. Therefore, if the distance is low, it suggests that the probe image have too low interclass variation, implying that the image in

question is also of high quality. Conversely, a high distance indicates greater inter-class variation, suggesting lower image quality.The three distinct mated distances are transformed into a similarity score $s \in [0, 1]$ defined by $s = \frac{1}{1+e^d}$, where d is each mated distance with zero mean and unitary standard deviation. Finally, the three normalised similarity scores were averaged to obtain the final groundtruth quality measures for training FaceQnet v1.

Following the generation of ground truth for the training dateset the subsequent step is the training of the Deep Regression Model to perform an end-to-end regression for quality estimation. The idea is to use a pretrained CNN based on a ResNet-50 on VGGFace2 to do face recognition and then applying knowledge-transfer to change its domain from face recognition to quality assessment. Additionally, the final classification layer of the base model is removed and replaced with two Fully Connected (FC) layers tailored for quality estimation. The first FC layer compresses the embedding into a 32-dimensional feature vector that encapsulates quality-related information. The second FC layer performs a regression operation to produce a final quality score. A dropout layer is introduced before the first FC layer to mitigate overfitting and enhance generalization across diverse datasets and scenarios. The network receives input face images of size 224 x 224 x 3 which are cropped and aligned using MTCNN. During training, all original model weights are frozen, and only the newly added layers are trained using the generated quality ground truth. Once trained, FaceQnet functions as a black-box system that takes a face image as input and outputs a quality score ranging from 0 to 1. This score can be interpreted as a measure of similarity between the input image and a theoretically ideal ICAO-compliant face image. A visual representation of the architecture used is shown in the Figure 5, and a general overview of the hole process is shown in the Figure 6.

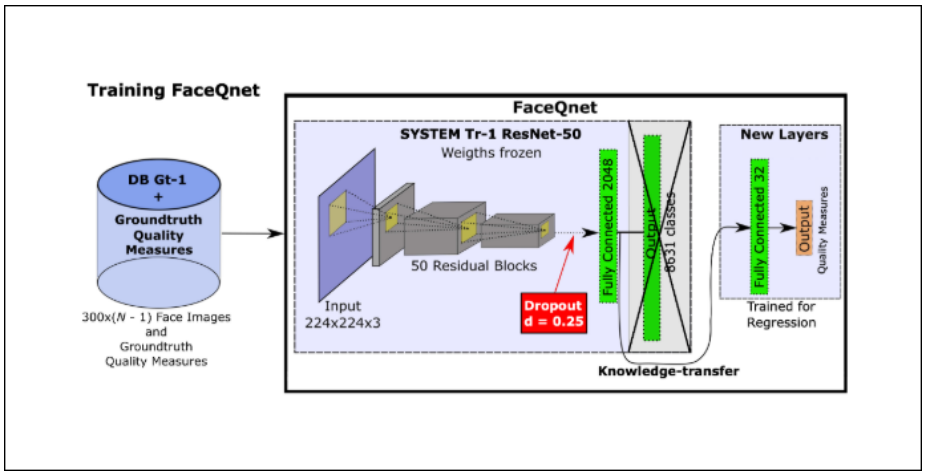

Figure 5: visual representation of the architecture of the Deep Regression Model. *Image taken from:* [5]

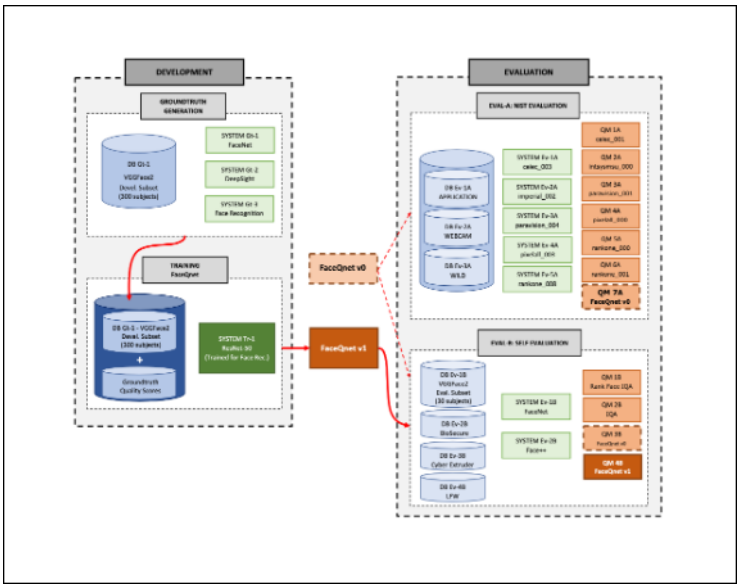

Figure 6: general overview of the groundtruth generation, training of the Deep Regression Model and evaluation in FaceQnet v1. *Image taken from:* [5]

3 Methodology

In this section we will examine two key aspects such as the datasets used and the experimental procedure followed for evaluating the system. The implementation details of the system will be discussed in section 4 while the results of the evaluation will be presented in section 5.

As discussed in section 2, the core components of the system rely on deep learning models including RetinaFace, FaceNet and FaceQnet. For both the experimental evaluation and the actual implementation, these models were used in their pre-trained form. Where RetinaFace was trained on the WIDER FACE dataset [14], while both FaceQnet and FaceNet were trained on the VGGFace2 dataset [2]. Additionally, FaceNet's training also included the CASIA-WebFace dataset. As previously mentioned FaceNet was not the only face recognition model used for evaluation. Dlib and DeepID were also tested. Both models were used in their pre-trained forms, where Dlib was trained on VGGFace2 and FaceScrub, while DeepID was pre-trained on the YouTubeFaces dataset [13].

Since all the models used in this study project are pre-trained, we only require test datasets for evaluation purposes. To this end, we utilize the AT&T [9] dataset along with a custom-built dataset to perform a cross-dataset evaluation that from now on we will refer to it as DB-1. Lel's begin now discussing the motivation behind the selection of our datasets. AT&T is composed of 40 subject with 10 images for each one. The images are in black and white, featuring faces in frontal or slightly rotated positions with varying expressions and in

some cases wearing glasses. AT&T is useful for the study purpose of this project because as discussed in section 2 users are expected to cooperate in order to avoid losing access to the funds associated with their keys. Therefore, we can reasonably assume that the user will present a frontal pose without unusual facial expressions. However, one of the major limitations of the AT&T dataset is the complete absence of color information, which prevents the evaluation of variations caused by different illumination conditions. Additionally, due to the dated nature of the dataset the images are of low resolution and contain only the head, without any background or upper body context, unlike typical real-world scenarios where a camera would capture a wider frame. At this point, one might argue that given the desktop nature of our system and its primary use in indoor environments lighting conditions may not be a critical factor. However, this is not entirely accurate as even indoor settings can present significant variability in illumination which can affect facial recognition accuracy. To address these limitations and better simulate realistic conditions DB-1 was created.

DB-1 is a custom-built dataset designed specifically for evaluating biometric cryptosystems in the context of cryptocurrency wallets. It consists of 143 images across 30 subjects with each subject represented by 2 to 5 images. The dataset was created ad hoc to better reflect the realistic conditions under which the systems would operate. The concept behind DB-1 was inspired by the YouTubeFaces dataset which extracts frames from YouTube videos. In a similar fashion, DB-1 focuses on capturing streamers as they typically look directly into the camera, much like a user would when verifying their identity to access their funds. However, a known limitation of YouTubeFaces is the presence of many nearly identical frames. DB-1 overcome tihis issue by carefully capturing the images manually from each video. The following aspects were considered during the dataset creation:

- Images captured under varying indoor illumination conditions, poses and facial expressions;
- Images captured from both older and more recent videos of the same subject (when available) to account for changes due to aging;
- Subjects were selected to ensure a balanced representation of gender and ethnicity minimizing potential bias. Specifically, no single gender or ethnic group is excluded in the dataset. More specifically, the distribution of gender and ethnicity in the dataset is as follows:
  - 60% males and 40% females;
  - The ages of the subjects range approximately from 18 to 60 years old.
  - Individuals from a range of ethnographic backgrounds such as European/White Caucasians, Africans/Afro Americans, South Asian, and East Asian populations have been considered, with representation from both men and women.

Figure 7: Overview of some of the sample images included in the DB-1 dataset.

The Figure 7 provides an overview of some of the samples images included in the DB-1 dataset.

Having now examined the datasets used we can proceed to outline the methodology adopted for evaluating the system. The adopted methodology, executed separately for each dataset, proceeds as follows:

1. Subdivide the dataset test set into probes and gallery

    1.1. For each subject in the dataset one image sample is randomly selected to serve as the probe and the others to serve as the gallery in the verification context. This random selection is intended to minimize selection bias.

    1.2. Once the dataset subdivision into probes and gallery is completed the next step involves computing similarities in a $PROBE$-against-$ALL_{GALLERY}$ context. This process results in the generation of a similarity matrix. The operation is carried out for each of the face recognition models introduced in section 2, namely, FaceNet512, Dlib, and DeepID all of them using RetinaFace as the face detector. The resulting similarity matrices are stored in CSV files as they are computed only once due to the high computational cost associated with their calculation.

2. Once the similarity matrices for each model are computed the next step is to define the role of the probe in the verification process. Specifically, whether it should act as a genuine attempt or an impostor.

- In this phase each similarity matrix was evaluated under different distributions of genuine and impostor attempts. Specifically, scenarios included are all probes as impostors, all as genuine, a 30-70 split between impostors and genuine (and vice versa) and finally a balanced 50-50 distribution. Moreover, the identity of each impostor is randomly selected from among all other identities. This procedure was done to stress the system under various different genuine/impostors scenarios.

3. For each impostor/genuine setup and for every similarity matrix the actual verification benchmarking is performed. In this process, each probe is compared against the gallery templates of the claimed identity. If multiple templates are present for a given identity, the one with the highest similarity score is selected. Then, for each threshold value t ranging from 0 to 1 in steps of 0.005, the following evaluation is made:
   - $max\_similarity \geq t \land identity = impostor \Rightarrow FA + 1$ (False Acceptance);
   - $max\_similarity < t \land identity = genuine \Rightarrow FR + 1$ (False Rejection);
4. To evaluate the system's performance for each threshold t the False Rejection Rate (FRR) and False Acceptance Rate (FAR) are computed as follows:
   - $FRR = \frac{FR}{|Genuinie|}$, where $|Genuine|$ is the number of genuine probes attempts;
   - $FAR = \frac{FA}{|Impostors|}$, where $|Impostors|$ is the number of impostors probes attempts;

As we will observe in section 5 by following this evaluation procedure an Equal Error Rate (EER) of 0 was achieved using FaceNet512 and RetinaFace on both the DB-1 and AT&T datasets even without applying quality assessment via FaceQnet. Nonetheless, for robustness an evaluation on both datasets quality have been made. Figure 8 and Figure 9 illustrate the distribution of probabilistic quality density for the AT&T and DB-1 datasets, respectively. Based on these considerations and the empirical analysis of image quality, a threshold of 0.30 was still established to filter out low-quality probe samples in the actual implementation.

4 Implementation

In this section will be presented an overview of the actual implementation of the system which will be presented as the system's architecture and an outlining of the technologies used and the motivations behind their selection. Moreover,

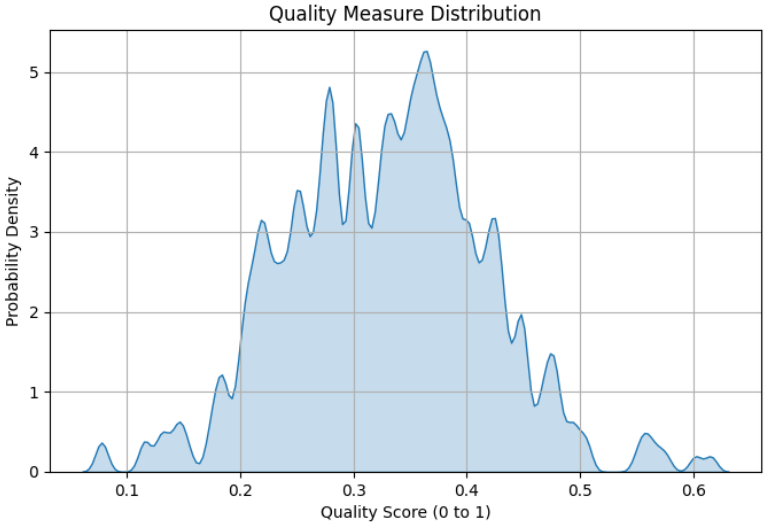

Figure 8: Distribution of probabilistic quality (FaceQnet quality measure) density for the AT&T dataset.

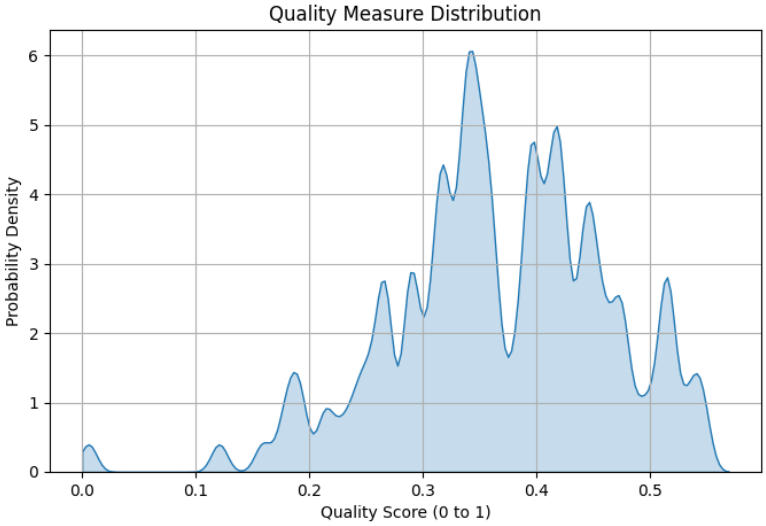

Figure 9: Distribution of probabilistic quality (FaceQnet quality measure) density for the DB-1 dataset.

each component of the system will be described and by the end of the section the key elements related to biometrics and cryptographic keys will be highlighted.

Let's start by taking a look at the architecture overview shown in the Figure 10. BioWallet is a desktop based application designed to run locally through the user's web browser As illustrated in the architecture the system follows the modern web based paradigm instead of a classical native GUI, separating the presentation layer (frontend) from the business logic and database access (backend). The main reason behind this choice is the greater flexibility and scalability offered by web based applications compared to traditional GUI based solutions. Additionally, it could be notice how it is technically possible to deploy the application components on a web server and operate it over the internet, however, the system was not developed with this use case in mind. As such, doing so could introduce potential security vulnerabilities.

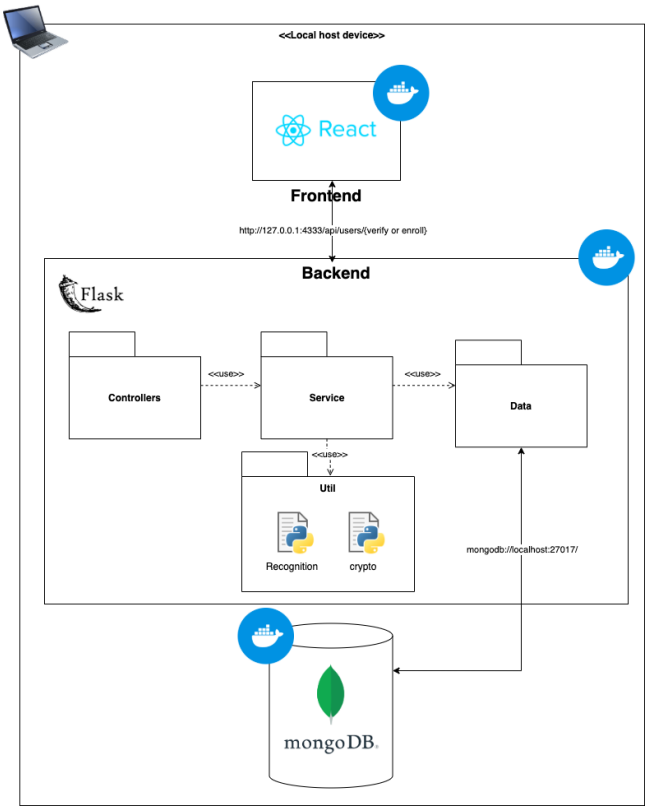

Figure 10: Architecture overview of Biowallet.

Each main component is marked with the docker symbol, indicating that it runs within its own docker container locally on the machine. This approach offers several advantages. First, it simplifies sharing and deploying the project by eliminating concerns about dependencies or complex installation procedures,

users only need docker and docker-compose installed, then a few commands are enough to launch all three systems. Second, using docker enhances local security by adhering to the principle of separation of privilege [8] isolating components ensures that if one fails or is compromised the others remain unaffected. Each docker container is isolated from the underlying operating system functioning almost like a lightweight virtual machine. These containers communicate with each other through network bridges allowing for controlled and secure interaction between system components.

For the purposes of this project the most relevant aspect is the backend functionality, which includes the core components of the biometric cryptosystem and the structure of the database representing the gallery. While the frontend is only briefly mentioned, it is worth noting that React is a javascript library which was chosen for the presentation layer due to its component-based architecture which enables flexible and reusable development. On the backend side, Flask was used, a lightweight python micro framework ideal for creating APIs in a minimal and efficient way. The backend follows a three layered architecture, divided into controller, service and data layers. Of particular interest is the service layer where the biometric cryptosystem is implemented. These functionalities are futher organized within a utils package with biometric processing and key generation logic separated into distinct modules.

Let's now take a look at the relevant sections of code for the enrollment and verification phases. The code for enrollment is shown in Listing 2 while Listing 1 contains the verification.

```
    def verify(id, image):
        user = dao.get_user_by_id(id)
        if user  == None:
            raise Exception(f"The identity {id} is not pressent in
    the system.")
        qualities = recognition.images_quality([image])
        for i in range(1, len(qualities)):
            if qualities[i][0] < 0.30:
                raise Exception(f"The image {qualities[i][1]+1} has
     poor quality. \
                Please try to align the face with the camera and
    maintain an appropriate distance, similar to a passport photo."
    )
        probe = recognition.get_embeeding_from_image(image)
        count = 0
        for template in user["templates"]:
            similarity = recognition.get_simmilarity(probe,
    template)
                if similarity < 0.81:
                    count += 1
        if count == len(user["templates"]):
            raise Exception(f"The provided image is not of the
    given identity")
        token = jwt.encode( { 'user_id': id, 'exp': datetime.
    datetime.now() + datetime.timedelta(hours=1) },  "
    sjfdjfJsbhasUik!u3y^^^^&623", algorithm='HS256')
```

```
19          return token
```

Listing 1: code for verification in the service layer

```
1   def enroll(id, images):
2       data = { "id" : id, "templates" : [], "keys" : []}
3       if dao.get_user_by_id(id) != None:
4           raise Exception(f"The identity {id} is already pressent
     in the system.")
5       qualities = recognition.images_quality(images)
6       for i in range(1, len(qualities)):
7           if qualities[i][0] < 0.30:
8               raise Exception(f"The image {qualities[i][1]+1} has
     poor quality. \
9               Please try to align the face with the camera and
    maintain an appropriate distance, similar to a passport photo."
    )
10      for img in images:
11          templates = recognition.get_embeeding_from_image(img)
12          data["templates"].append(templates)
13      for i in range(0, 5):
14          sk = crypto.generate_private_key()
15          pk = crypto.generate_public_key(sk)
16          data["keys"].append(
17          {"pubblic_key" : pk, "private_key" : sk})
18      dao.insert_user(data)
```

Listing 2: code for enrollment in the service layer

As can be observed the two code segments share some structural similarities in certain parts. For instance, both codes include checks to verify whether the user exists or not and whether the submitted image sample meets the required quality standards. In the enrollment phase multiple samples can be submitted while in the verification phase only one sample is typically provided. However, due to the way the image_quality function is implemented, expecting a list as input, even a single image is passed as a list. Then, the two functions diverge. During enrollment embeddings are extracted from each image sample and then the public and private key pairs for the user are generated. Subsequently the data is stored in the MongoDB database following exactly the structure shown in line 2 of Listing 2. Lastly for enrollment, the choice to generate five pairs of public and private keys is arbitrary. Indeed, there is no strict requirement for this number, it simply demonstrates the system;s capability to support multiple key generations. Instead, the verify function it gets the probe by extracting the embedding from the submitted image sample. It then compares this embeddings with the stored templates associated with the claimed identity in the gallery. Since there may be multiple templates verification is considered successful if at least one template has a similarity score greater than or equal to 0.81 a threshold, which is determined through the experimental evaluation made in section 5. Toward the end of the function could be notice how a token is generated. This part is not directly related to the biometric process, instead, it serves a technical purpose ensuring that only successfully verified users receive a valid JWT token, which is required to access the keys page.

As seen in the previously shown code for the verify and enroll functions some functions were used as black boxes. These are the utility functions mentioned at the beginning of this section when the system architecture was introduced. Their implementation can be found in Listing 3 and Listing 4.

```
import os
import ecdsa

def generate_private_key():
        return os.urandom(32).hex()

def generate_public_key(private_key):
    private_key_bytes = bytes.fromhex(private_key)
    sk = ecdsa.SigningKey.from_string(private_key_bytes, curve=
    ecdsa.SECP256k1)
    vk = sk.get_verifying_key()
    public_key_bytes = vk.to_string()
    x = public_key_bytes[:32]
    y = public_key_bytes[32:]
    if int.from_bytes(y, 'big') % 2 == 0:
        prefix = b'\x02'
    else:
        prefix = b'\x03'
    compressed_public_key = prefix + x
    return compressed_public_key.hex()
```

Listing 3: implementation of the public and private key pair generation logic.

```
from deepface import DeepFace
import numpy as np
import cv2
import base64
from tensorflow.keras.models import load_model

def get_embeeding_from_image(img):
    embedding = DeepFace.represent(
        img_path = img,
        model_name = "Facenet512",
        enforce_detection=True,
        detector_backend="retinaface",
        align=True
    )[0]['embedding']
    return embedding

def images_quality(imgs):
    model = load_model("./util/models/FaceQnet_v1.h5")
    faces = []
    result = [["image", "score"]]
    for i in range(len(imgs)):
        data = imgs[i].split(',')[1]
        image_bytes = base64.b64decode(data)
        image_array = np.frombuffer(image_bytes, dtype=np.uint8)
        img_cv = cv2.imdecode(image_array, cv2.IMREAD_COLOR)
        try:
            face_box = DeepFace.extract_faces(img_path=imgs[i] ,
    detector_backend = "mtcnn")[0]["facial_area"]
            padding = 0
```

```
            y1 = max(face_box["y"] - padding, 0)
            y2 = min(face_box["y"] + face_box["h"] + padding,
    img_cv.shape[0])
            x1 = max(face_box["x"] - padding, 0)
            x2 = min(face_box["x"] + face_box["w"] + padding,
    img_cv.shape[1])
            cropped_face = img_cv[y1:y2, x1:x2]
            cropped_face_resized = cv2.resize(cropped_face, (224,
    224))
            faces.append((cropped_face_resized, i))
    except:
        result.append([0, i])
    y = np.array([f[0] for f in faces], copy=False, dtype=np.
    float32)
    score = model.predict(y, batch_size=1, verbose=1)
    for i in range(len(score)):
        result.append([score[i][0], faces[i][1]])
    return result

def get_simmilarity(embedding1, embedding2):
    similarity = np.dot( np.array(embedding1),  np.array(embedding2
    ) )/(np.linalg.norm(np.array(embedding1))*np.linalg.norm(np.
    array(embedding2)))
    minx = -1
    maxx = 1
    normalize_similarity = (similarity-minx)/(maxx-minx)
    return normalize_similarity
```

Listing 4: implementation of the utility functions supporting biometric operations

Listing 3 shows the implementation of the public and private key generation which is carried out using the ecdsa library to maintain clarity and simplicity in the code. The procedure of this code follows the exact steps described at the beginning of section 2. Listing 4 relies on two widely used libraries such as Tensorflow keras and DeepFace. In particular, DeepFace has been used due to its modularity and simplicity offering a unified interface that wraps several well-known face recognition and detection models along with alignment capabilities. This not only make the implementation simpler but also helped for evaluating multiple recognizes without making major changes to the code. Moreover, it also provides convenient methods for extracting embeddings from different models. At the end of the file can be notice a custom similarity function that is implemented to compare those embeedings with the result normalized within the $[0, 1]$ range with the help of the Numpy library to perform efficiently the vector operations. Instead, tensorflow keras is used to load the pre-trained FaceQnet model. Before using this model, however, the images must undergo several preprocessing steps. As seen in the image_quality function after the model is loaded each submitted image is first decoded from its base64 format and then converted into a buffer format, making it readable by OpenCV's imdecode function which will later be used for model predictions in the NumpyArray. The images are cropped to a default size of 224x224 pixels around the detected face using the MTCNN face detection method as required from FaceQnet (as de-

tailed in section 2). If no face is detected a default value of 0 quality is inserted. Once the predictions are made, the results are stored in a list, including the quality score and the corresponding image position if the imgs list.

In this section just an overview of Biowallet implementation have been shown. However, the hole code base and results are available and can be found in the Github repository of this project.

5 Evaluation

This section presents the experimental results obtained using the methodology described in section 3. The goal of the experiment was to achieve a FAR of 0 while minimizing the FRR within the specific context of biometric cryptosystems for wallet implementation, assuming user collaboration. Indeed, aside from image quality and user collaboration they are the only assumptions. This effectively reduces the problem to a specific subset focused on evaluating the efficiency of the tested models within our defined context. Therefore, it is not within the scope of this experiment to test images captured in the wild with challenging conditions such as extreme poses, poor illumination or low resolution. The experiments were conducted on a CPU based setup, specifically, using an AMD Ryzen 5 2600 processor running on Debian 12 with 16GB of RAM. No GPU was utilized for inference, which unfortunately resulted in high computational costs for the evaluation process.

Figure 11 through Figure 16 present the evaluation results of the selected models plotting the FAR and FRR across the error rate. As observed in Figures Figure 11, Figure 12 and Figure 13 a FAR of 0 is achieved. Notably, the Equal Error Rate (EER) is also 0, indicating that both FAR and FRR are simultaneously zero. Moreover, it is important to note that the presented results correspond to the scenario where the probability of a probe being an impostor or a genuine claim is 50%. However, as discussed in section 3, several other scenarios with varying impostor to genuine setup were also evaluated. To summarize the additional scenarios, Facenet512 combined with RetinaFace consistently achieved an ERR of 0 across all tested impostor to genuine distributions in both the AT&T and DB-1 datasets. For the remaining models, the following observations can be made:

- DeepID achieved an ERR of 0 on the AT&T dataset when the probability of an impostor claim was set to 70%;
- Dlib consistently achieves an ERR of 0 across all scenarios on the AT&T dataset. In other impostor genuine claim distributions it also maintains an ERR of 0, except, in the case where 70% of the claims are impostors, there it achieves a FAR of 0 but with a FRR of 0.375.

This is a summary of the key results for the other scenarios, however, the complete set of evaluations (file) are available in the project's GitHub repository and can be visualized using the `plot.py` script located in the verification submodule.

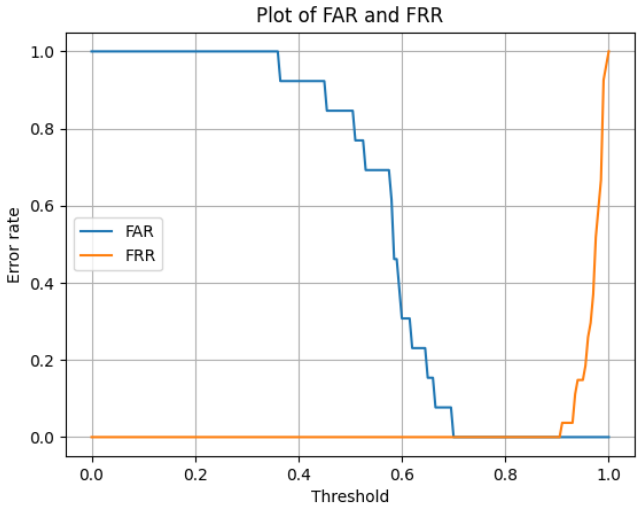

Figure 11: FRR-FAR with 50% probability of being either an impostor or genuine claim, Retinaface is used as face detector together with Facenet512 as recognition module. The image is aligned and the dateset for used is AT&T.

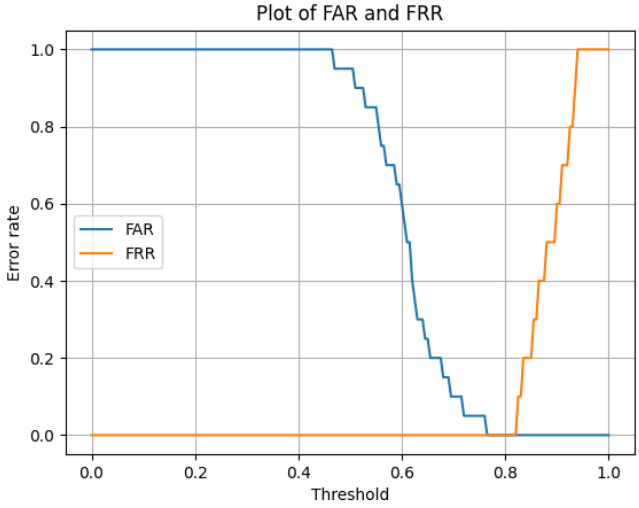

Figure 12: FRR-FAR with 50% probability of being either an impostor or genuine claim, Retinaface is used as face detector together with Facenet512 as recognition module. The image is aligned and the dateset for used is DB-1.

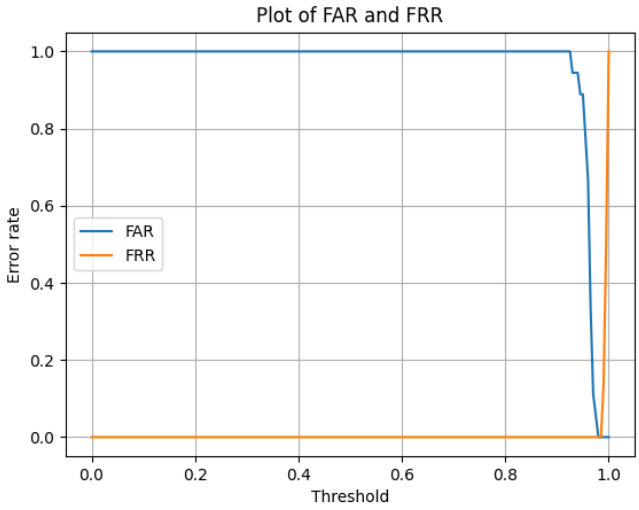

Figure 13: FRR-FAR with 50% probability of being either an impostor or genuine claim, Retinaface is used as face detector together with Dlib as recognition module. The image is aligned and the dateset for used is AT&T.

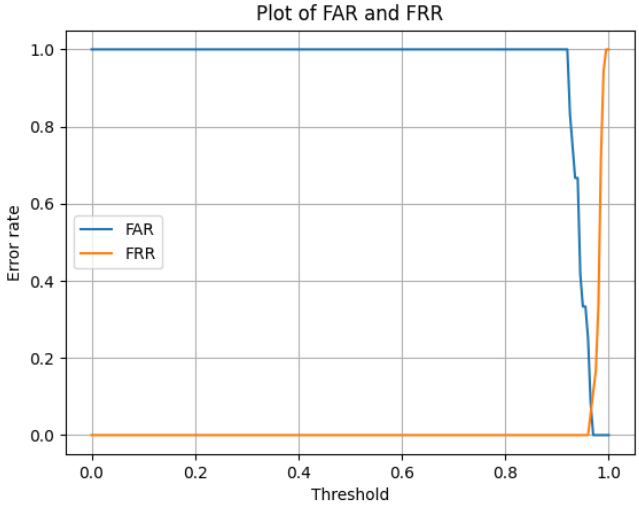

Figure 14: FRR-FAR with 50% probability of being either an impostor or genuine claim, Retinaface is used as face detector together with Dlib as recognition module. The image is aligned and the dateset for used is DB-1.

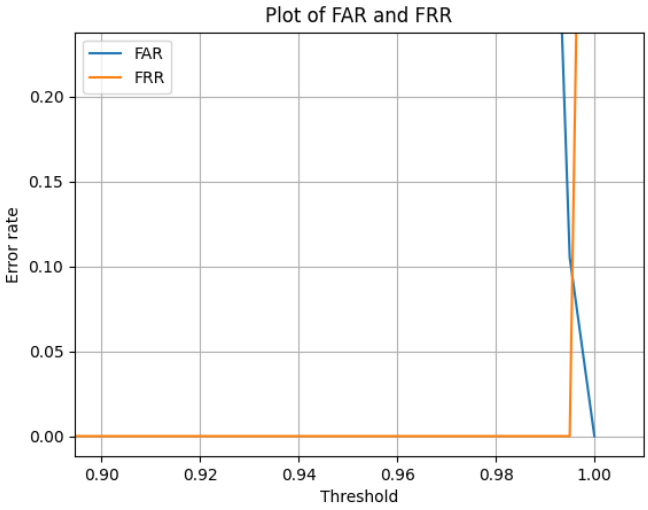

Figure 15: FRR-FAR with 50% probability of being either an impostor or genuine claim, Retinaface is used as face detector together with DeepID as recognition module. The image is aligned and the dateset for used is AT&T.

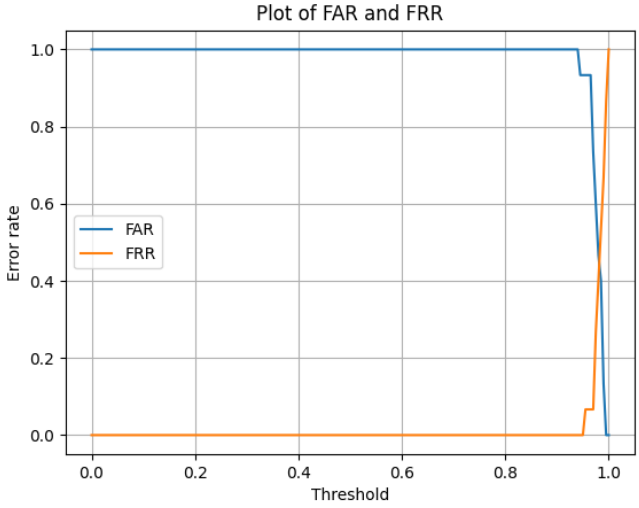

Figure 16: FRR-FAR with 50% probability of being either an impostor or genuine claim, Retinaface is used as face detector together with DeepID as recognition module. The image is aligned and the dateset for used is DB-1.

6 Conclusions

Through this project a proposal of a cryptocurrency wallet, such as for Bitcoin, based on a biometric cryptosystem using facial recognition have been proposed. The biometric system follows a key release approach where the cryptographic key is only released after a successful verification. The system incorporates a standard biometric verification phase, enhanced by a quality assessment step using the deep learning model FaceQnet, which ensures that image samples meet a minimum quality for use as either a probe or a template in the gallery. The evaluation showed that among the tested models Facenet512 paired with RetinaFace for face detection and alignment proved to be the most suitable for the verification process. It consistently achieved an equal error rate of 0 across all scenarios and datasets, including AT&T and DB-1. These results confirmed our objective, which was demonstrating that it is possible to reach a FAR of 0 in a specific and collaborative context.

Looking ahead, future work could include evaluating a much larger number of samples from the DB-1 dataset to test the scalability and reliability of the results. Furthermore, since the application's core functionality has been implemented, the next step would be to integrate it with blockchain technologies to provide a fully functional cryptocurrency wallet. Indeed, the goal of this project was not to develop a fully functional cryptocurrency wallet, but rather to evaluate the feasibility of a biometric cryptosystems, focusing specifically on the generation and release of cryptographic keys through facial verification.

References

[1] Bitcoin Wiki. Secp256k1. https://en.bitcoin.it/wiki/Secp256k1, 2025.

[2] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[3] Aldo Cascone. Using face templates as cryptographic keys. 2023.

[4] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. *arXiv preprint arXiv:1905.00641*, 2019.

[5] Javier Hernandez-Ortega, Javier Galbally, Julian Fiérrez, and Laurent Beslay. Biometric quality: Review and application to face recognition with faceqnet. *arXiv preprint arXiv:2006.03298*, 2020.

[6] Maheep Singh Prabhjot Kaur, Nitin Kumar. Biometric cryptosystems: a comprehensive survey. *Multimedia Tools and Applications*, 82, 2023.

[7] Muhammad Khurram Khan Reem Alrawili, Ali Abdullah S. AlQahtani. Comprehensive survey: Biometric user authentication application, evaluation, and discussion. *Computers and Electrical Engineering*, 119, part A, 2024.

[8] Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[9] F.S. Samaria and A.C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 138–142, 1994.

[10] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[11] Joseph H. Silverman. An introduction to the theory of elliptic curves. Summer School on Computational Number Theory and Applications to Cryptography. University of Wyoming. https://www.math.brown.edu/johsilve/Presentations/WyomingEllipticCurve.pdf, 2006.

[12] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1891–1898, 2014.

[13] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534, 2011.

[14] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.