POLYTECHNIC OF BARI

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING – BARI

MASTER DEGREE COURSE IN COMPUTER ENGINEERING

BUSINESS REPORT

ADVANCED SOFTWARE ENGINEERING

FINGERPRINT-BASED KEY PAIR GENERATION

Professor:
Prof.ssa Ing. Marina MONGIELLO

Group members:
Domenico de GIOIA
Ivan MADDALENA

ACADEMIC YEAR 2022/2023

Summary

Introduction

Biometric information is unique and safe for every individual, and replication of biometric template is challenging which makes its application systems more secure compared to traditional methods such as password and PIN. However, due to the immutability of biometric information, a leakage of biometric information may cause consecutive security threats.

Approaches for generating cryptographic key from biometrics have considered various biometric traits, such as fingerprint, finger vein, iris, face, voice, handwritten signature, gait, ECG, retina and multi-modal. Fingerprint is the most widely used biometric trait among all these traits for practical applications.

Hence, fingerprint biometrics is considered as a part of cryptographic key pair generation. The stable binary string generated from biometrics can be used directly as a cryptographic key for symmetric encipherment, or it can be used as a seed value to generate the private–public key pairs.

An asymmetric-key cryptosystem is mainly used for secure exchanging of session key parameters and digitally signing electronic messages. It uses a pair of keys: the public key and private key.

So, after this market research, aimed at understanding what could be the key points of such a project, our Team has acquired fundamental information and has defined the project in detail. It is on this basis that the Team decided to focus on the development of this software.

The aim of this project was to create a code that could generate a keypair starting from a fingerprint and being able to recognize a fingerprint if it had already been entered previously, considering the risk that the data biometric changes over time and therefore managing to identify the fingerprint even in the face of drastic changes.

Capitolo 1

DESIGN

1.1 Requirements

The main functions and characteristics of the system were identified before the actual design phase and the implementation one. This step is fundamental because in this way mini-goals to be achieved are identified.

1.1.1 Functional requirements

1) The system must be implemented on a local server.

2) The system must allow the registration of the user by fingerprint.

3) The system must verify the correctness of the fields entered during registration.

4) The system must implement an algorithm able to extract a keypair starting from the user's fingerprint.

5) The system must store the user fingerprint image.

6) The system must allow the authentication of the user by fingerprint.

1.1.2 Non functional requirements

1.1.2.1 Implementation requirements

1) The part of the system concerning the extraction of the features from fingerprints, the keypair generation and the fingerprint matching must be developed in Python using different image processing packages and algorithms.

2) The development of the web platform will be carried out with Flask for the backend and with HTML and CSS for the frontend.

1.1.2.2 Quality requirements

1) The system must ensure fast response times during execution.

2) The system must guarantee the security and confidentiality of the data entered by the user, requiring authentication of the latter.

3) The system must guarantee the reliability of the data used.

4) The system must run on different software and hardware platforms, ensuring its portability.

1.2 Architecture

The code is divided into two main sections: the first one is represented by the feature extraction process and key-pair generation, while the second one is represented by the fingerprint matching. This subdivision is useful to understand the peculiarities and weaknesses of the code, and to be able to intervene where necessary.

1.2.1 Keypair generation from fingerprint biometrics

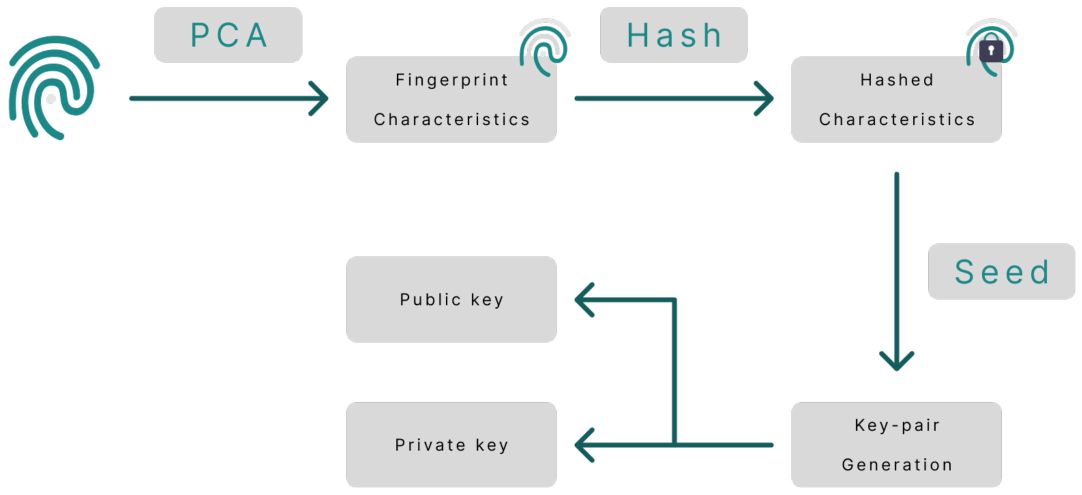


Figura 1.1 - Generation of cryptographic keys from biometrics.

1.2.1.1 Feature extraction

Obtaining a stable key string from fingerprint biometrics is the main objective of this proposed work.

Minutiae points in a fingerprint uniquely characterize each individual. Hence, accurate minutiae points from a captured fingerprint image are necessary to generate a stable key string for each individual. In particular:

- a point where a ridge ends or terminates immediately is termed as a ridge ending;
- a point where two ridge lines are obtained by splitting a single ridge line is termed as ridge bifurcation.

Ridge bifurcation and ridge ending are the most widely accepted minutiae point. Hence, only ridge ending and ridge bifurcation points are used in this project.

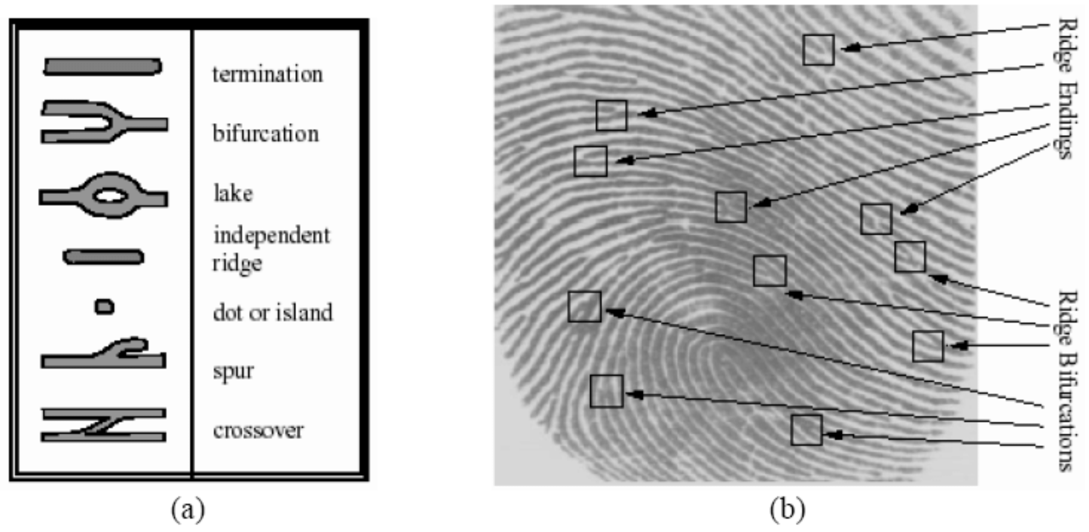


Figura 1.2 – (a) Different minutiae types, (b) Ridge ending and bifurcation.

The traditional approach to this problem consists in various phase. The performance of feature extraction and comparison algorithms are strictly related to the image quality.

Supposing to receive a grayscale image as fingerprint input, the first step is the binarization (a) of the image that is a method of converting any grayscale image into a black-white image or a binary image. To perform binarization process, first find the threshold value of gray scale with mean thresholding method and check whether a pixel having a particular gray value or not. If the gray value of the pixels is greater than the threshold, then those pixels are converted into the white, while similarly if the gray value of the pixels is lesser than the threshold, then those pixels are converted into the black.

The next step is the skeletonization (b) of the binary image that reduces the binary image to 1-pixel wide representations retaining the original size of the object. This operation let us to study the behavior of the ridges.

After that, the minutiae locations are derived with Crossing Number function defined as half of the sum of differences between intensity values of two adjacent pixels. If Crossing Number for a specific minutiae point is 1, 2 and 3 or greater than 3, then that minutiae points are classified as termination, normal ridge, and bifurcation respectively.

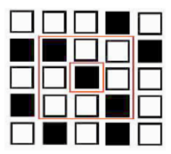
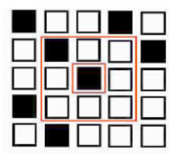
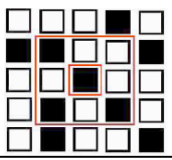
	Crossing Number =2. Normal ridge pixel.
	Crossing Number =1. Termination point.
	Crossing Number =3. Bifurcation point.

Figura 1.3 – (a) Binarization, (b) Skeletonization, (c) Minutiae extraction

This last step gives the list of n minutiae coordinate values $m_i = (x_i, y_i)$ as the output.

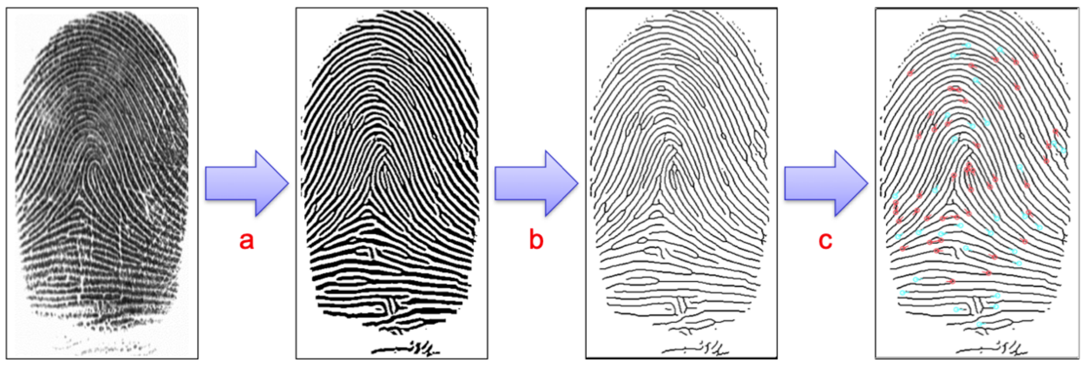


Figura 1.4 – (a) Binarization, (b) Skeletonization, (c) Minutiae extraction

1.2.1.2 Stable seed generation

Euclidean distance between each pair of n minutiae points is calculated from the selected consistent set of minutiae points. Let two minutiae points m_i and m_j be located in the coordinates (x_i, y_i) and (x_j, y_j), respectively. The Euclidean distance between this pair of minutiae points is computed as:

$$d_{i,j} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{1}$$

Euclidean distance between pair of minutiae points is considered here due to its rotational and translational invariance. Despite rotation and/or translation induced changes in the absolute coordinate values of the minutiae points, the distance between each pair of minutiae points does not vary. Hence, it does not affect the stable key string generation process.

These Euclidean distance values ED_i are sorted in ascending order and they are converted to 8-bit binary number $B_1, B_2, \dots, B_n$. Then, the 8-bit binary number is converted into 8-bit gray code $G_1, G_2, \dots, G_n$.

Gray code is a reflected binary code in which two successive values differ in only one bit. Due to biometric uncertainty, if there is slight change in the distance value, then there will be only minimum change on the generated bit string. So, a gray code-based encoding is used to generate a stable key string from fingerprint biometrics.

The 8-bit representations of each gray code are concatenated to obtain a single key bit string $G = G_1 \cup G_2 \cup \dots \cup G_n$. To generate the key string having a consistent length of 256 bits, the generated stable bit string G is hashed with the SHA-256 algorithm in a deterministic way. This hash value is considered as a seed value, that is stable for the same fingerprint, from which to generate the keypair.

Example of stable string:

```
102835736030916179805136905511247252246064256194822790465423795724867574327811
```

1.2.1.3 Keypair generation

For this purpose, "Solana" Python package and its classes were used. Other packages for keypair generation were evaluated but Solana was chosen for the format of the keys and the best execution times as reported in the test report.

This approach generates two distinct cryptographic key strings for two different users. Therefore, the chance of producing the same cryptographic key string by a genuine user and an impostor is eradicated. Furthermore, this seed value ensures the generation of the same keypair every time. The experimental results show that the proposed approach can ensure a stable generation of the key.

Example of keypair generation:

```
    public key:6RLJp57QPuaWxnHWojZ5f8C4qBMgDRv9Z2MFbVyLVHKS
    secret_key:e35af6c5f6a2ef4061d091fa0cf55422b418d0e3ed01f7c35961b43453926a0
3508634641c928f500c2295eab1ef04154e84208dd2cec22033e02a1876fc33fd
```

1.2.1.4 Fingerprint matching

For this purpose, OpenCV Python package is used. `SIFT` object is a Scale invariant Feature Transform object. In this case, the `SIFT` object represents an algorithm that finds the image keypoints, the ridges, in the fingerprints. Each keypoint is a special structure with many attributes like its coordinates, size of relevant neighborhood, the angle which specifies its orientation, and response that specifies the strength of keypoints, etc.

Thanks to this object keypoints and descriptors can be identified and detected in both the database images and in the test one. These structures are scale and rotation invariant.

After the detection and the computation are over, we start the matching algorithm. We shall use the `FlannBasedMatcher` (FLANN, Fast Local Approximate Nearest Neighbors) functionality to compare the descriptors of the original fingerprint image and the sample one. `FlannBasedMatcher` and similar algorithms work very fast and efficient for large datasets. It returns a list of each pair of points whose distances is calculated.

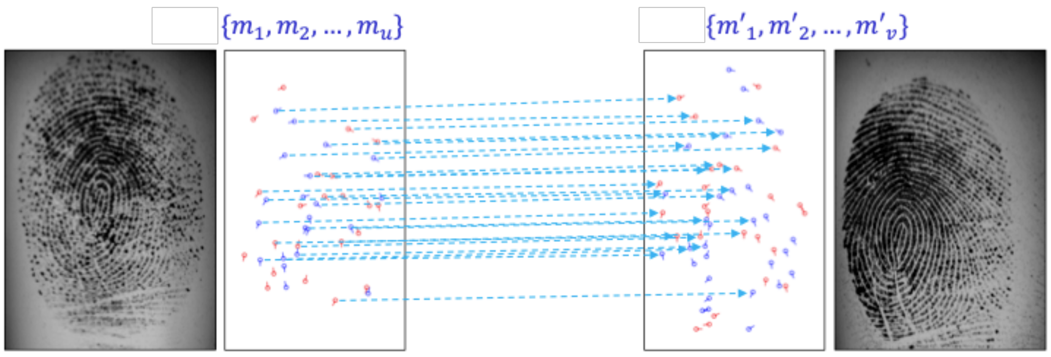


Figura 1.5 – Fingerprint matching.

After that, we calculate the match score as:

$$score = \frac{number\ of\ pairs}{\min\{u, v\}} \tag{2}$$

1.2.2 Flask local server

The Flask environment let us start a local development server. The simple steps to build a Flask application are the following.

- Create an instance of the `Flask` class with the application name. This is needed so that Flask knows where to look for templates, static files, and so on.

- Use the `route()` decorator to tell Flask what URL should trigger our function. The function is given a name which is also used to generate URLs for that particular function and returns the message we want to display in the user's browser.

- By default, a route only answers to `GET` requests. The methods `argument` of the `route()` decorator can be used to handle different HTTP methods.

- Web applications also need static files, for example CSS and JS files. Just create a folder called `static` in our package and it will be available at `/static` on the application.

- Web applications also need generating HTML from within Python. To render a template just use the `render_template()` method providing the name of the template and the variables to pass to the template engine as keyword arguments. Flask will look for templates in the `templates` folder.

- To react to the data a client sends to the server by forms, Flask provide the global `request` object. The current request method is available by using the `request.method` attribute while to access form data (data transmitted in a `POST` or `PUT` request) you can use the `request.form` attribute.

- Flask let us handle uploaded files easily setting `enctype=”multipart/form-data”` attribute on your HTML form and looking at the `request.files` attribute on the request object. Uploaded files are stored in memory or at a temporary location on the filesystem. Each uploaded file is stored in that dictionary. The file also has a `save()` method that allows you to store that file on the filesystem of the server.

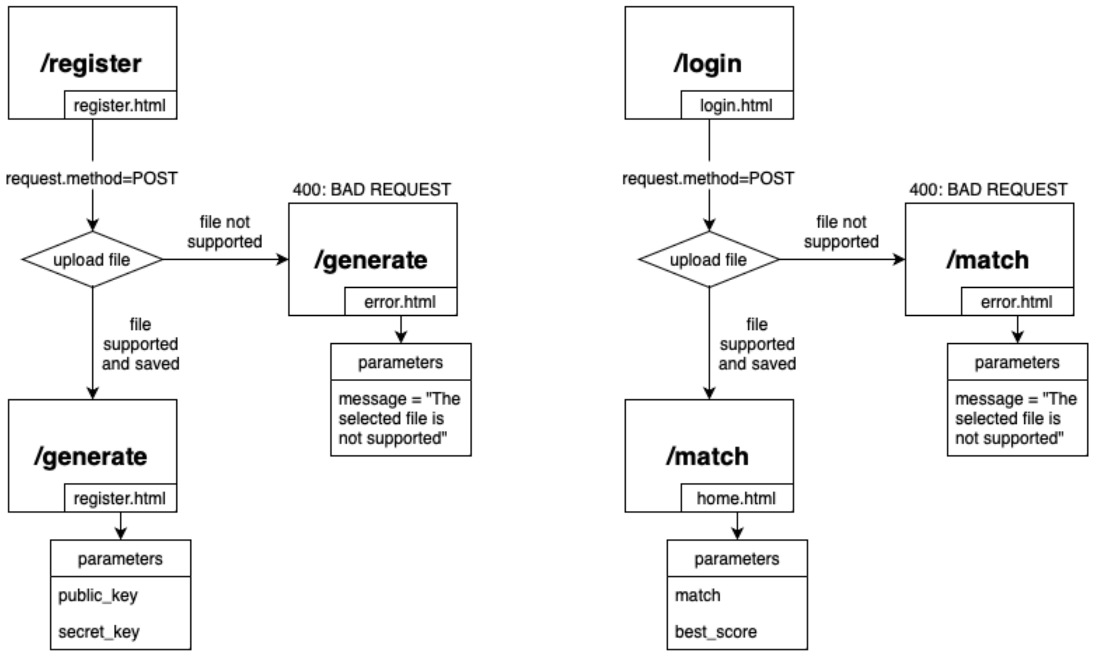


Figura 1.6 – Flask architecture of the fingerprint based authentication system.

Capitolo 2

VALIDATION

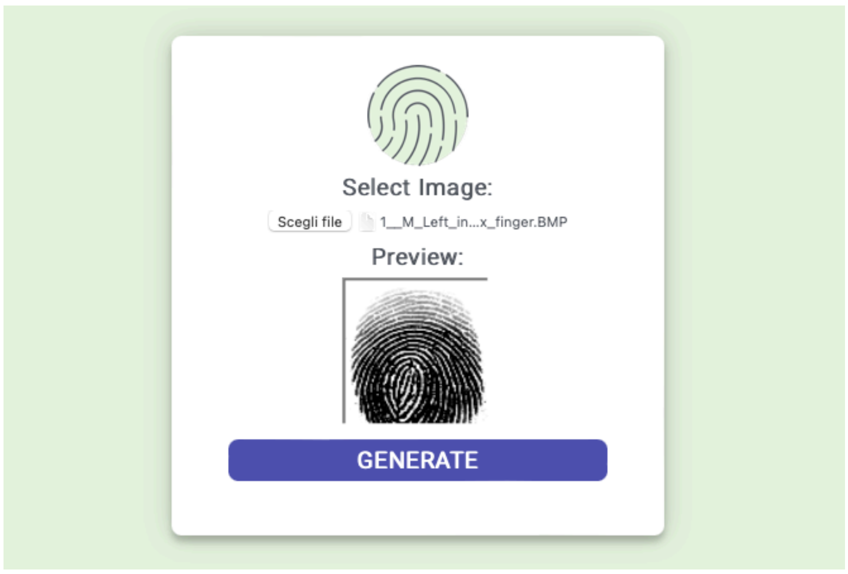
Validation is the step in which you build a version of the product according to the requirements. We then analyze the bugs and logistical problems found in the system at the end of the development process. To this end, the use of PyCharm, as a development environment, and some of its functions was critical. The Team was thus able to verify and validate the correct functioning of each section of the application.

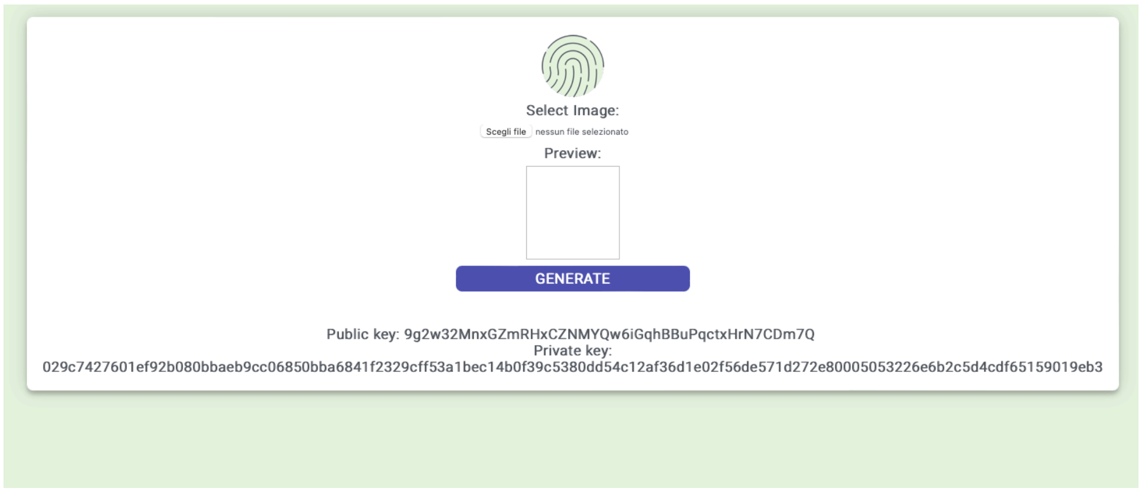
The tests were also carried out throughout the coding process. It concerned the correction of bugs in the code and the correct display of screenshots.

The test report contains data on the controls carried out on the different sections of the system:

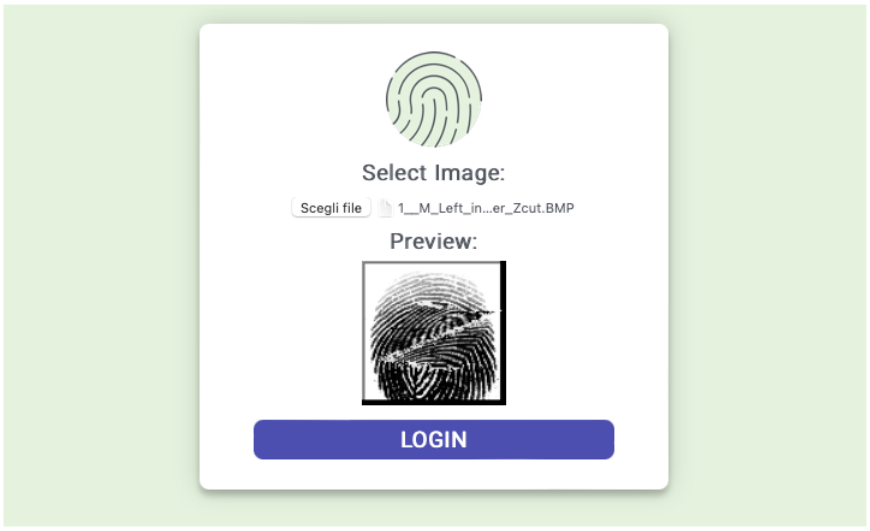
- user registration and cryptographic key-pair generation;
- user authentication and fingerprint comparison.

This has allowed us to validate our product, thus ensuring key features for every application such as efficiency, speed of execution and high-level performance. For any corrections that were made to the system, further tests were performed on the project to verify that the integrity of the project was maintained.





```
Public key: 9g2w32MnxGZmRHxCZNMYQw6iGqhBBuPqctxHrN7CDm7Q
Private key:
029c7427601ef92b080bbaeb9cc06850bba6841f2329cff53a1bec14b0f39c5380dd54c12af36d1e02f56de571d272e80005053226e6b2c5d4cdf65159019eb3
```

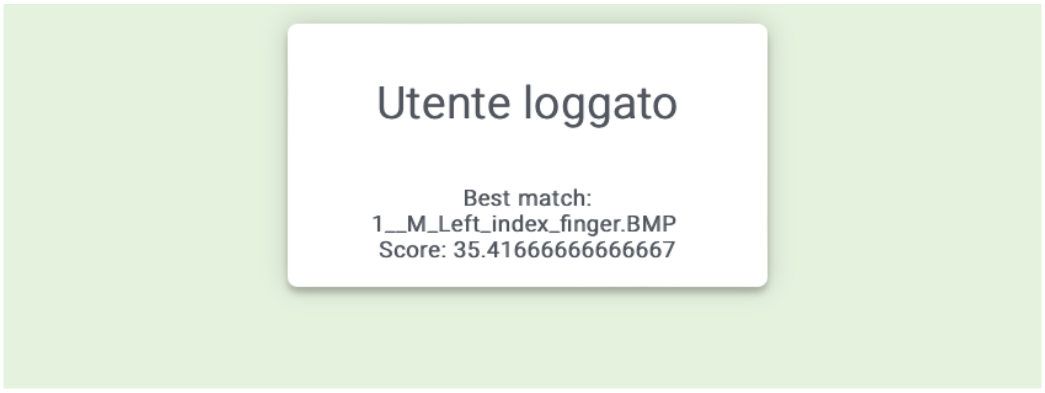


Select Image:

Scegli file 1__M_Left_in...er_Zcut.BMP

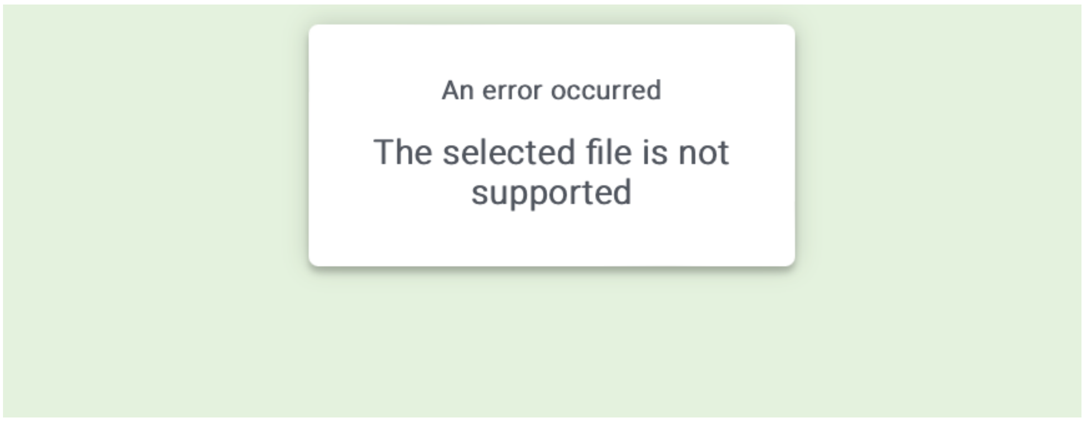
Preview:

LOGIN



Utente loggato

Best match:
1__M_Left_index_finger.BMP
Score: 35.41666666666667



An error occurred

The selected file is not supported

Conclusion

The final product of this project is a fingerprint-based authentication system which allows the user to register for the platform generating a keypair from a fingerprint as input and access it through login with fingerprint.

The system has two main features:

- The keypair generation process behaves deterministically, generating the same keypair per user. This aspect resolves a possible sybil attack by preventing a user to generate multiple keypairs.

- There is a risk that the biometrics may vary over time. For example, a cut on the finger rather than a burn, may vary the features extracted. In this sense, the system recognizes the similarity between two fingerprints even in the event of significant deterioration of the same.

A significant challenge is the storage of biometrics. Without it, it would not be possible to carry out the comparison of fingerprints. The problem can be solved by using, for example, a distributed filesystem such as IPFS. It is a peer-to-peer storage network. Content is accessible through peers located anywhere in the world that might relay information, store it, or do both. The main disadvantage of HTTP system is that all resources in HTTP are housed on a centralized single server and that server can be inactivated or censored based on what entity manages it, thereby eliminating the only source of the data housed on it. On the contrary, IPFS content is housed in several locations: it means that there is no single point of failure in a single server, and no one entity can censor or eliminate the data.

Project design, implementation, testing, and documentation took about 3 months, between November 2022 and January 2023. This path comes to an end on January 16, 2023, with the presentation of the project. In these months, the 18 Group Team has worked hard and with passion, with the sole purpose of delivering a high-performance product that meets the demands and requirements. The documentation phase was carried out meticulously, with strict attention to data and a graphic section that was clear, unambiguous and interesting.

The system can of course be updated in the future, thus implementing new features and fixing bugs that may arise. The project has given the expected results and for this reason the whole team is very satisfied. It is not excluded that in the future the Team can continue to work together to develop new versions of the system or new project.