POLITECNICO DI BARI

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
INGEGNERIA INFORMATICA

Test Report

Advanced Software Engineering

FINGERPRINT BASED KEY-PAIR GENERATION

Professors:
Prof.ssa Ing. Marina Mongiello
Ing. Marco Fiore

Group Code:
18

Group Members:
Domenico de Gioia
Ivan Maddalena

Academic Year 2022/2023

Contents

Executive Summary
Introduction (3)
Tests and analyzes (4-15)
Conclusions (16)

INTRODUCTION

The test report is a document issued by the programmer himself which provides information on the tests carried out on a product to ensure its conformity according to the standard used.

For the project in question, called "*Fingerprint based key-pair generation*", various tests were carried out on the code, to evaluate its suitability and other technical parameters, such as scalability.

The aim of this project was to create a code that could generate a pair of keys 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.

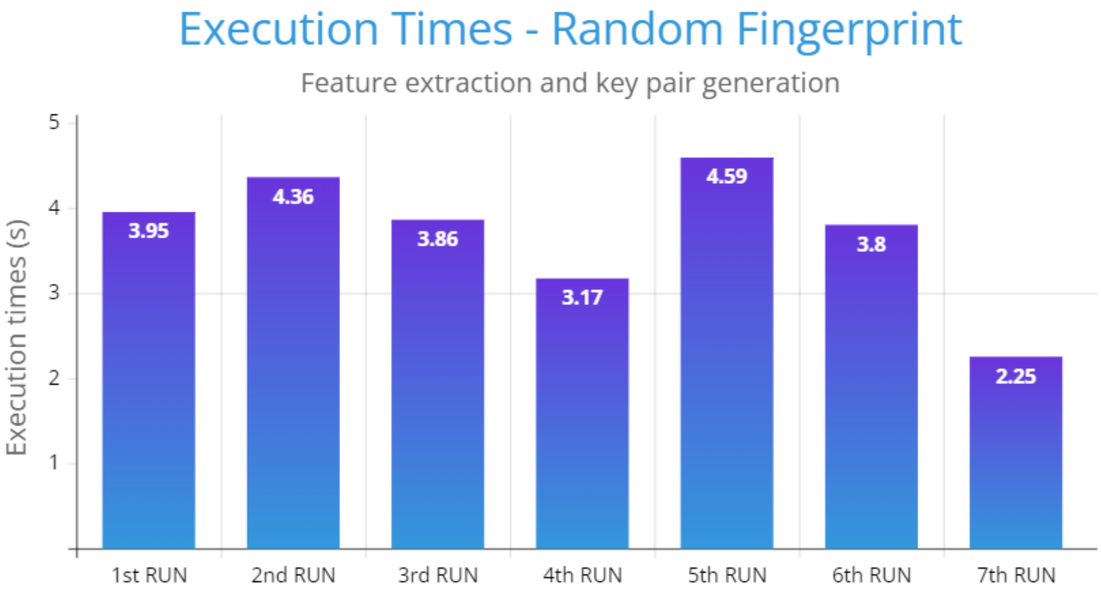
For testing, the code was ideally divided into three main subparts: feature extraction, key-pair generation, and fingerprint matching. This subdivision was useful to be able to understand the peculiarities and weaknesses of the code, to be able to intervene where necessary.

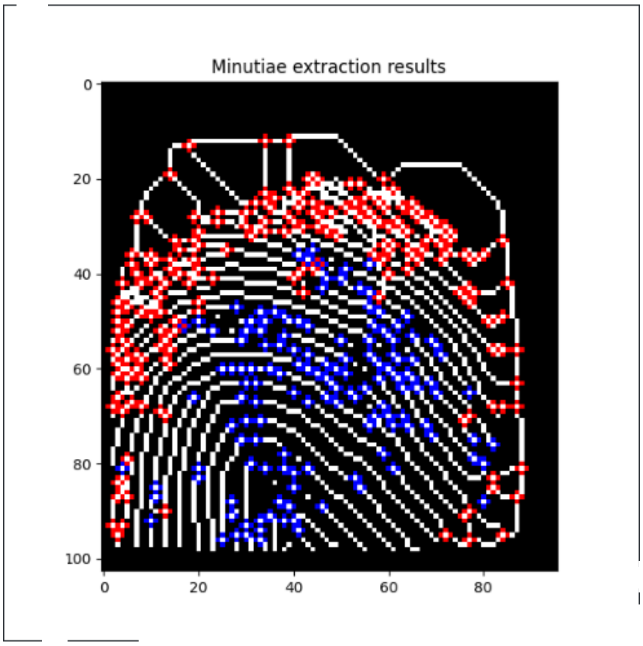
The data was collected by running the code in the development environment where it was written, namely PyCharm. The execution times, for example, have been collected thanks to the addition of appropriate lines of code.

The graphs shown in this document help to make the extrapolated data more understandable and to compare them with each other.

EXECUTION TIMES FOR FEATURE EXTRACTION AND KEY PAIR GENERATION FOR RANDOM FINGERPRINTS

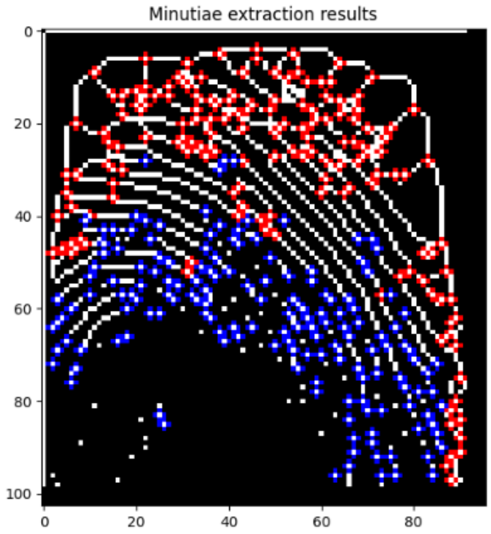
The following graph shows the execution times (in seconds) of the part of the code relating to feature extraction and key pair generation.
This test was performed to understand the average feature extraction and key pair generation times for random fingerprints.
Starting from the collected data, it is possible to observe that the execution times are proportional to the number of bifurcations and terminations detected on the processed fingerprint.
The average time taken by the code to produce the expected results, based on the data shown in the graph, is 3.71 s per fingerprint, which are times considered largely acceptable for an application of this type.



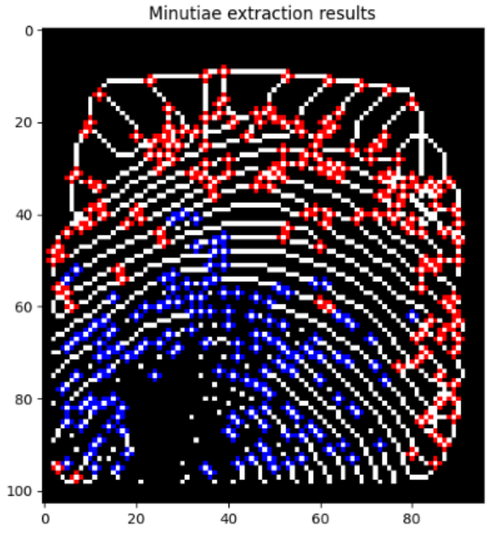


1st RUN
SOCOFing/Real\189__F_Right_middle_finger.BMP

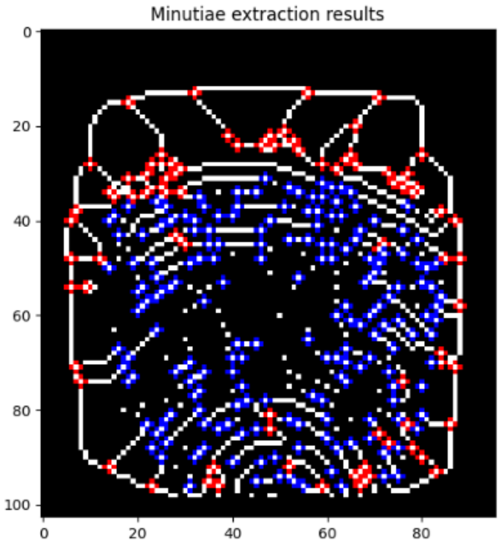
Execution Times: 3.95 s



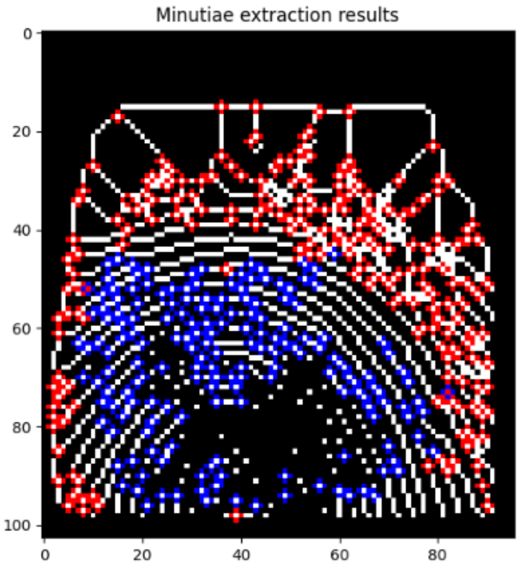
2nd RUN
SOCOFing/Real\426__M_Right_thumb_finger.BMP
Execution Times: 4.36 s



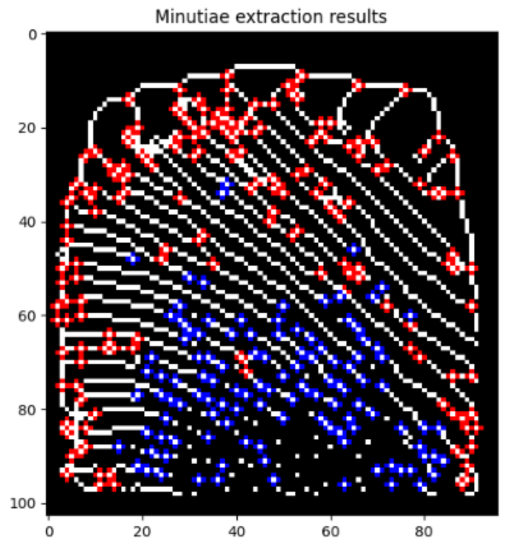
3rd RUN
SOCOFing/Real\358__M_Left_index_finger.BMP
Execution Times: 3.86 s



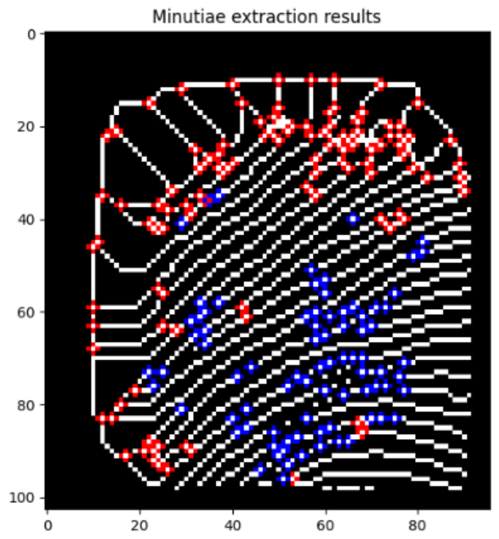
4th RUN
SOCOFing/Real\431__M_Right_ring_finger.BMP
Execution Times: 3.17 s



5th RUN
SOCOFing/Real\49__M_Left_ring_finger.BMP
Execution Times: 4.59 s



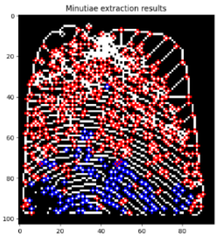
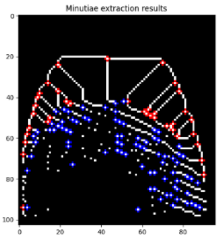
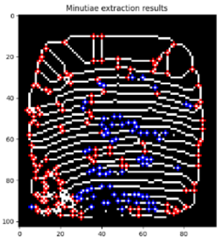
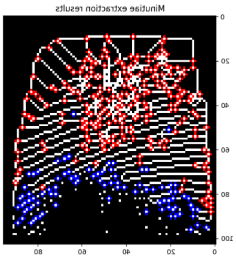
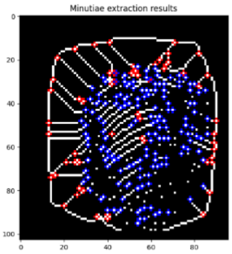
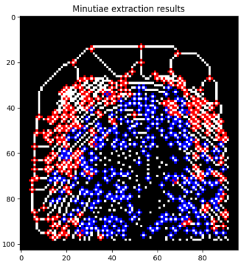
6th RUN
SOCOFing/Real\237__M_Right_thumb_finger.BMP
Execution Times: 3.80 s



7th RUN
SOCOFing/Real\566__M_Left_thumb_finger.BMP
Execution Times: 2.25 s

EXECUTION TIMES

Max value: 4.59 s (5th RUN)
Min value: 2.25 s (7th RUN)
Avg value: 3.71 s

FINGERPRINT	DIGEST’S LENGTH	EXECUTION TIMES
	4706312	8.49 s
	262088	1.61 s
	1042568	2.77 s
	2230272	4.48 s
	744200	2.24 s
	3317888	6.10 s

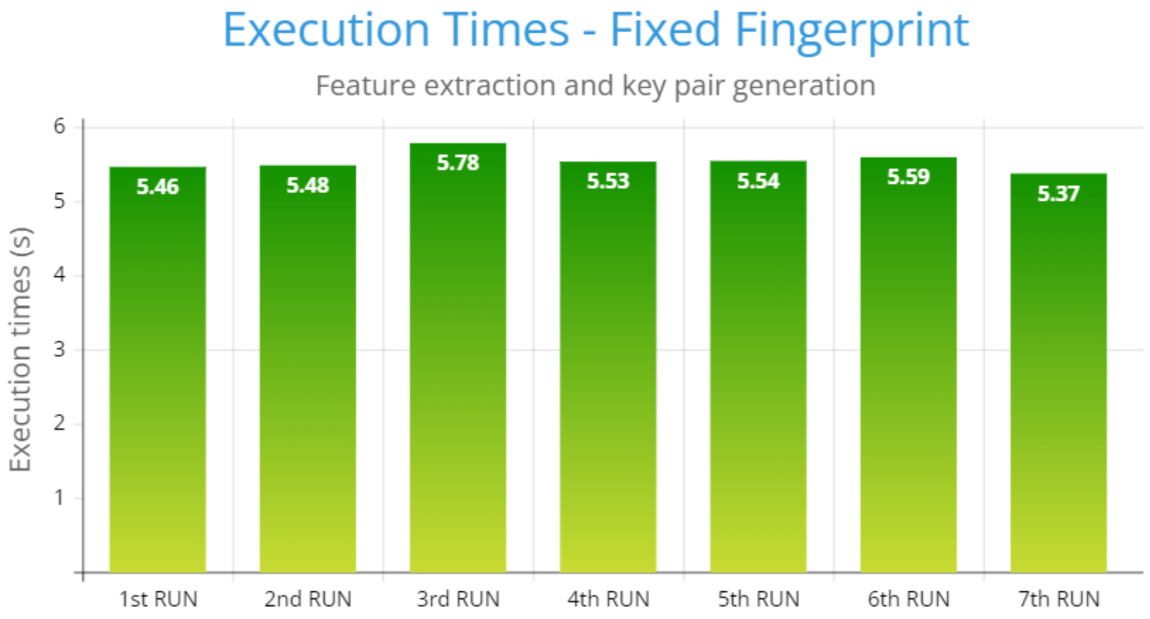
Starting from the considerations made previously about the relationship between the number of terminations and bifurcations and execution times, we have chosen to verify this deduction by considering the length of the digest for each fingerprint, i.e., the concatenation of the distances of all the minutiae points in a fingerprint.

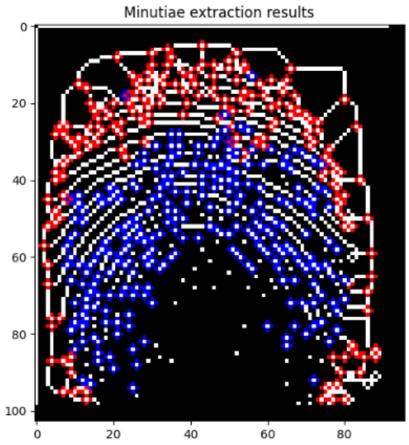
From the data, reported in the table above, it is evident that the relationship between minutiae points and execution times is verified, and that therefore the more terminations and bifurcations are present, the more time the execution of the feature extraction and key pair generation will require.

Taking the first case as an example, a length of 4706312, i.e., the highest of those recorded, corresponds to the highest execution time among those recorded, i.e., 8.49 seconds.
The shortest length, 262088 of the second fingerprint, corresponds to the shortest execution time, 1.61 s.
The same trend can be observed with all reported cases.

EXECUTION TIMES FOR FEATURE EXTRACTION AND KEY PAIR GENERATION FOR A FIXED FINGERPRINT

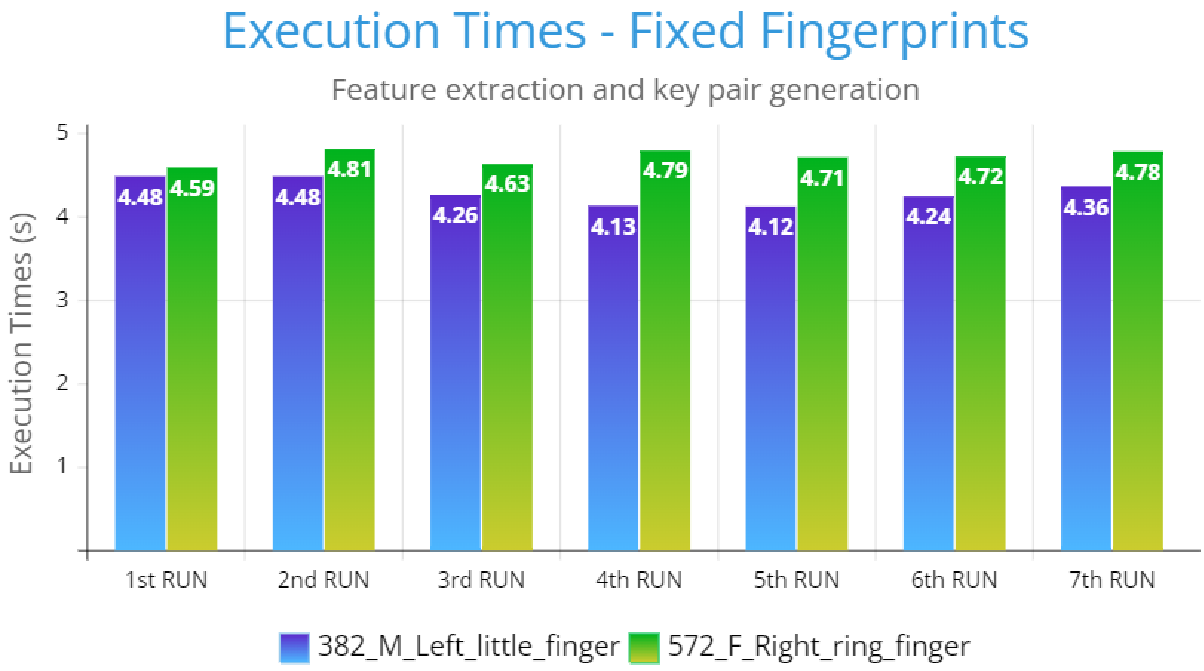
The graph below shows the execution times for feature extraction and key pair generation, exactly like the previous graph, but in this case the same fingerprint is always considered.
This test was performed to verify what the behavior of the code was if the same operation was repeated on a fixed fingerprint.

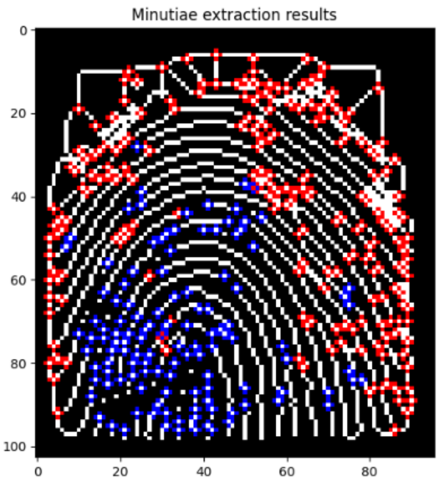




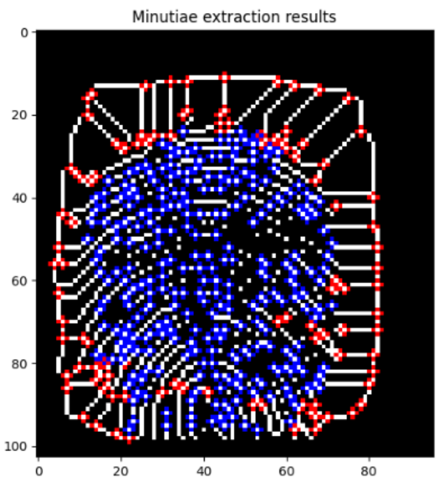
Based on the data highlighted by the graph, we can see that there is no trend in the execution times as the executions pass, but the times remain essentially constant.
The average execution time for feature extraction and key pair generation for this specific fingerprint is 5.54 s, the execution that required the longest was the 3rd, while the shortest was the 7th. However, as mentioned, the difference is a few tenths of a second, therefore not very relevant, and furthermore a trend cannot be detected in the displayed data.

Following the guidelines of the test above, we compared the execution time trend of two fixed fingerprints, inserting the respective times in the same graph.





SOCOFing/Real\382__M_Left_little_finger.BMP



SOCOFing/Real\572__F_Right_ring_finger.BMP

With this graph we have confirmed that the execution times remain stable for each individual fingerprint, and that there are no trends between executions.

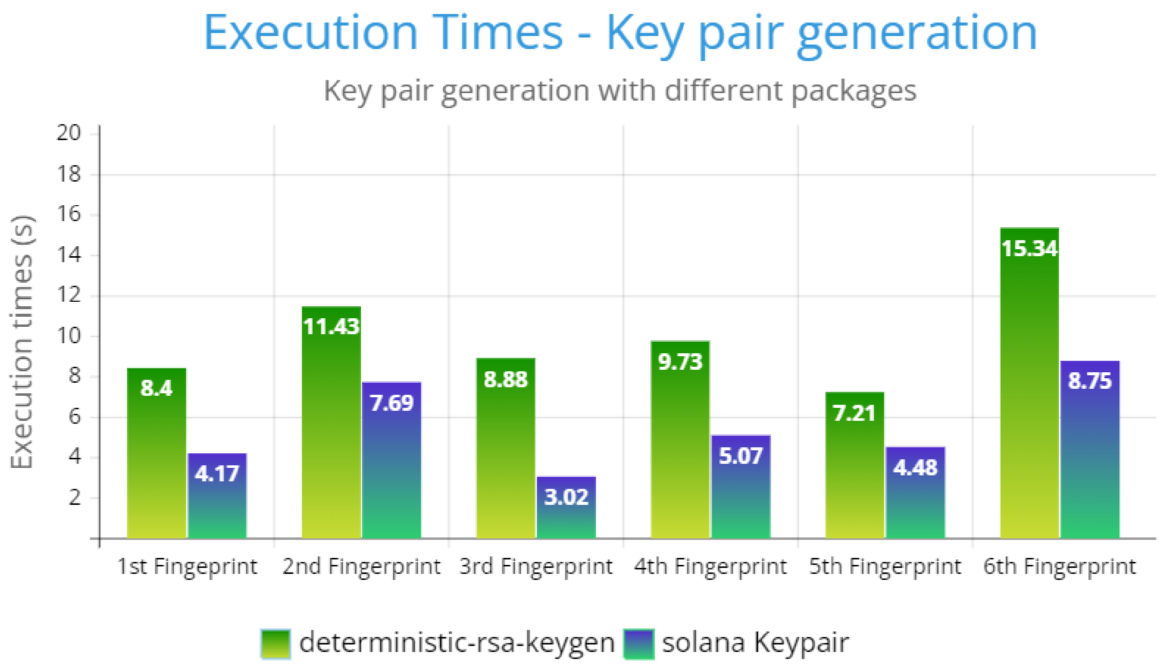
What we can observe is precisely the stability of these data. In fact, the lowest value of the execution times of *572__F_Right_ring_finger*, i.e., 4.59 s in the first execution, is in any case higher than the highest value for *382__M_Left_little_finger*, i.e., 4.48 s in the first two executions.

This testifies to the stability of the execution times of this piece of code for a fixed fingerprint.

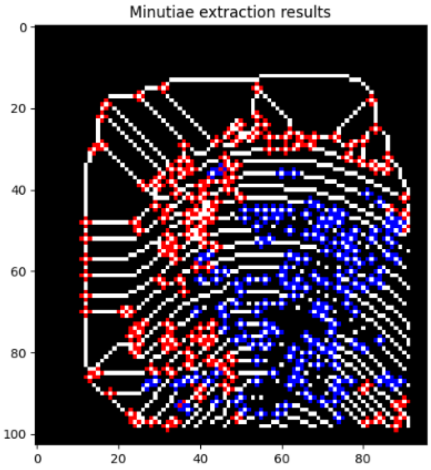
EXECUTION TIMES FOR THE GENERATION OF KEY PAIRS WITH DIFFERENT PACKAGES AND MODULES

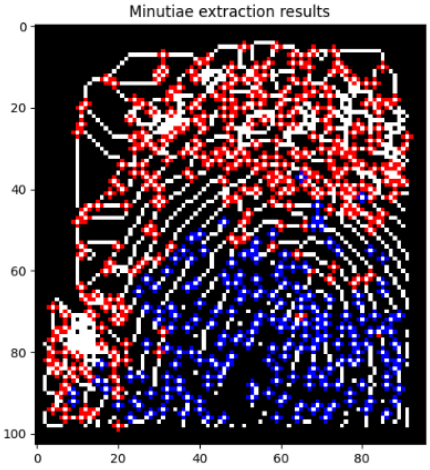
The execution times for key pair generation are obviously influenced by the package and modules chosen for this goal.
Analyzing the technical characteristics we needed, the choice was reduced to just two packages: *solana* and *rsa/deterministic-rsa*.
In particular, *solana.keypair* is a module to manage public-private key pair, *while deterministic-rsa-keygen* generate a deterministic RSA key pair and perform encrypt and decrypt operations.

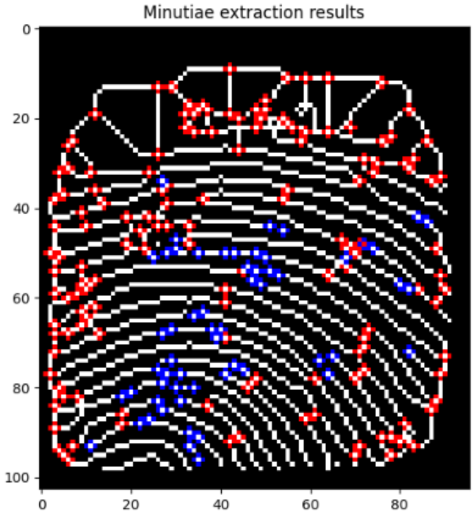
The graph below shows the execution times of the key pair generation and allows us to compare the two packages in terms of timing.

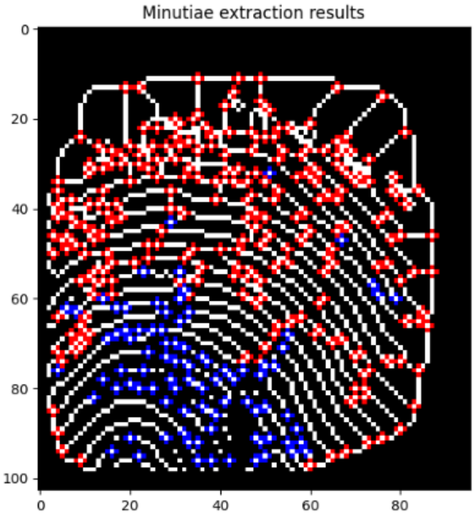


1st Fingerprint 2nd Fingerprint



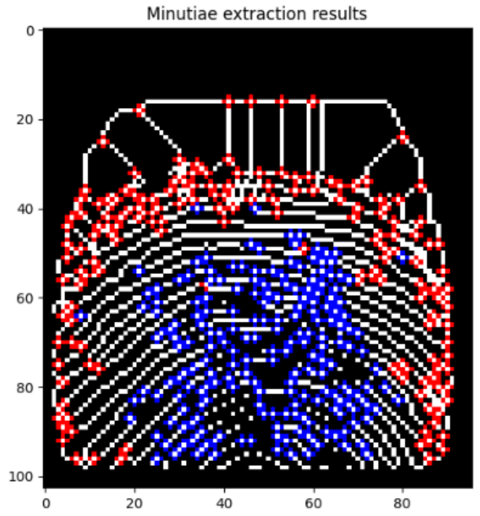




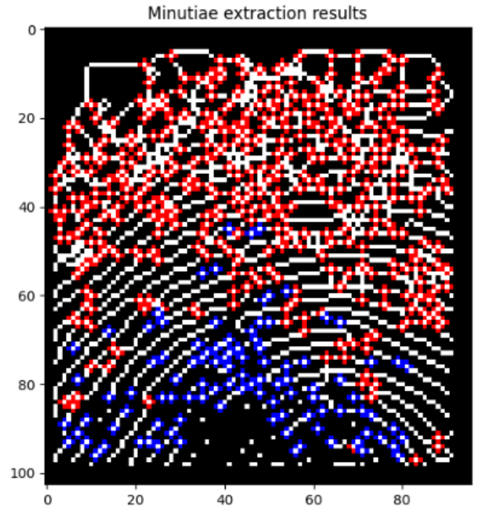


3rd Fingerprint

4th Fingerprint



5th Fingerprint



6th Fingerprint

Looking at the graph, it is clear why in the end the *solana* package was chosen: the execution times are in fact much reduced compared to *rsa*.

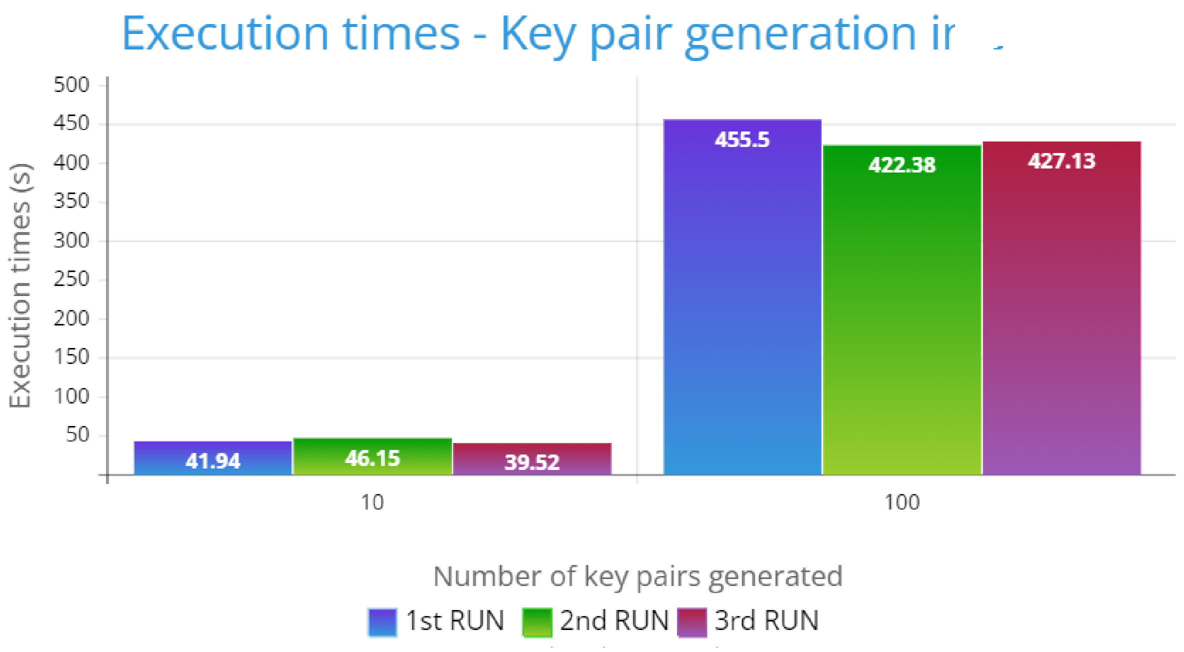
The average execution time for *solana* is 5.53 s, while for *rsa* it is 10.17 s.

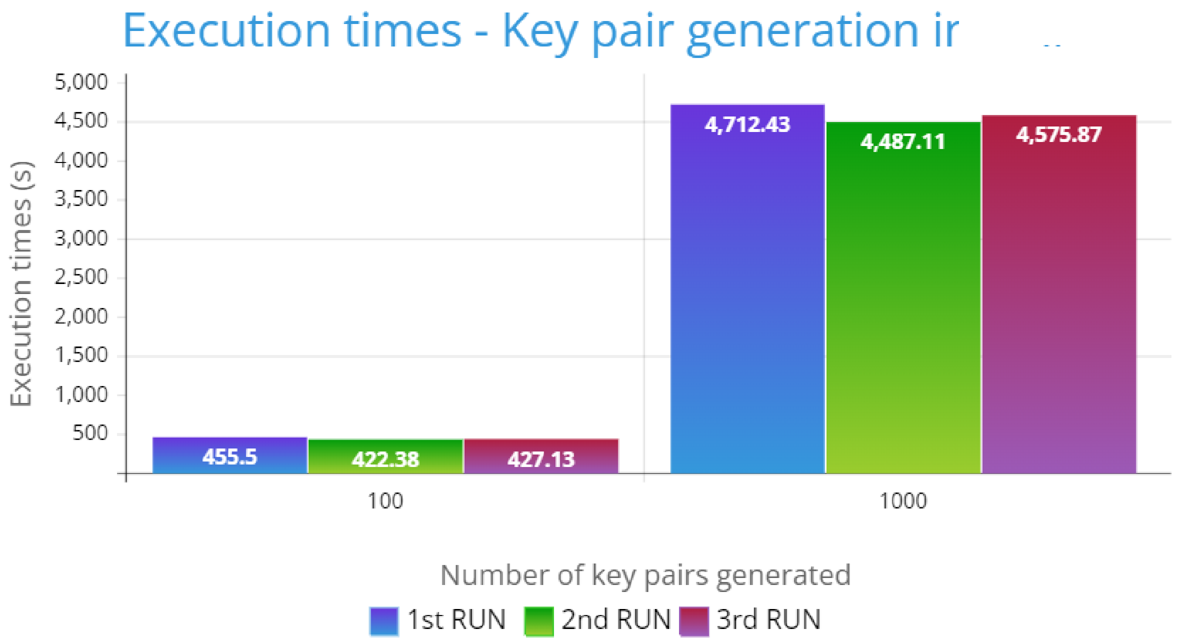
As we expected, the execution times are in any case proportional to each other, and this confirms what has been highlighted previously, i.e., that a fingerprint with more terminations and bifurcations involves longer execution times (in this case, for example, the seventh fingerprint turns out to be the one with the highest execution times, and in fact it is the one with the most minutiae features).

EVALUATION OF THE SCALABILITY OF THE CODE WITH ANALYSIS OF THE EXECUTION TIMES RELATED TO THE GENERATION OF BLOCKS OF 10/100/100 KEY PAIRS

The scalability of the code was tested by subjecting it to the generation of blocks of 10/100/1000 key pairs.

The results are shown in the following graphs:



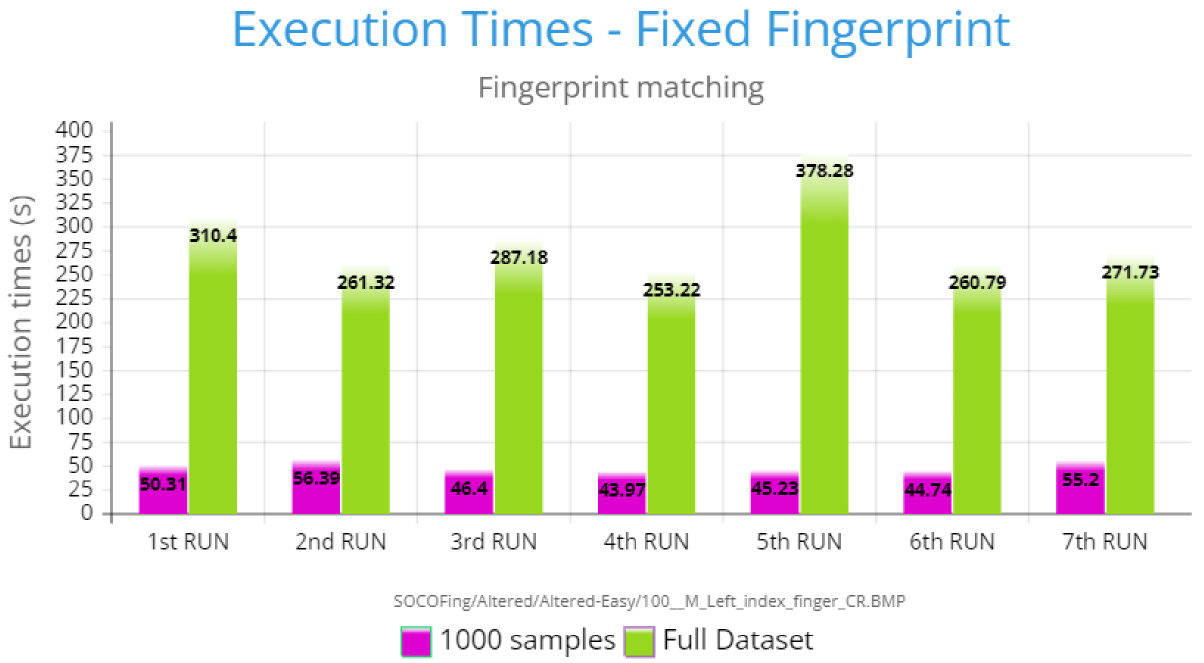


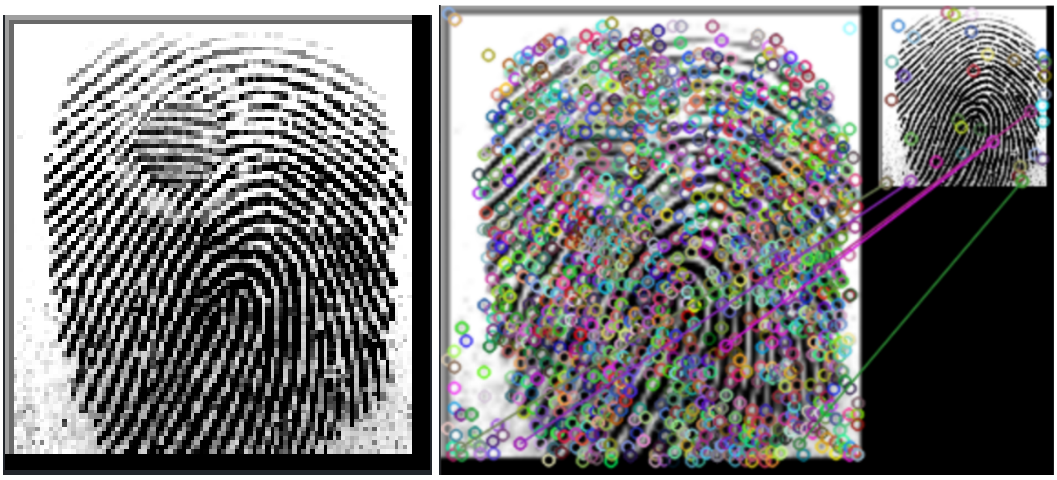
Evaluating the data shown by the graphs, we can see how the execution times are practically directly proportional to the number of key pairs generated.
This allows us to say that the code is scalable.

FINGERPRINT MATCHING EXECUTION TIMES WITH DATASETS OF VARIABLE SIZES

As regards the section of code relating to fingerprint matching, i.e., the ability of the code to recognize a previously encountered fingerprint, the execution times were analyzed if the fingerprint was to be searched for in a dataset of 1000 elements or in the entire dataset (6000 files).

The results obtained were shown in the following graph:





Also in this case, as previously, the execution times are directly proportional to the number of elements in the considered dataset.

This happens because the code evaluates each fingerprint with a score, and outputs the fingerprint with the highest score, i.e., the one closest to the fingerprint entered, and must therefore evaluate all the fingerprints present in the dataset (the threshold of the score can be varied to avoid the association with different fingerprints, but which have some points in common, ed).

CONCLUSIONS

The performance tests carried out have shown how the code responds well to the tasks to which it is called.
The execution times recorded for each section of the code are largely satisfactory and would allow the deployment of an application based on such work on a large scale.
In terms of scalability, the code has responded well and as expected.