

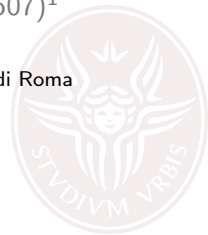
# Group Project

## Biometric Systems

Valerio Casalino (1916394)<sup>1</sup>    Mario Tobia Vendrame (1922290)<sup>1</sup>  
Shaahin Sabeti Moghaddam (1917507)<sup>1</sup>

<sup>1</sup>Cybersecurity Master @ Sapienza Università di Roma

Fall 2019



SAPIENZA  
UNIVERSITÀ DI ROMA

# Table of Contents

**General Concepts & Decisions**

Front-end Implementation

Data-set Management

Biometric Scanning Integration

Performance Assessment

Conclusions



SAPIENZA  
UNIVERSITÀ DI ROMA

# Premise

Before we start, let us say that all of our work, included this own presentation, is open sourced and available on Github:



<https://github.com/casalinovalerio/biosys-project>

There is also a script to replicate our setup for future projects.



# Overview

We wanted a face recognition based authentication application that is simple, yet particular. We deployed our test using:

- ▶ A **web interface** that works as a demonstrative placeholder. It gets the face with the camera, makes requests to our API server, which returns only a binary value for the success of the authentication.
- ▶ An **API server**<sup>1</sup> that queries the faces database and recognizes faces using the **@ageitgey's tool**<sup>2</sup>.
- ▶ A **database based on Blockchain**<sup>3</sup> that is an open source wrapper for a blockchain database that can be queried with standard SQL syntax. Implemented on the API server too.

---

<sup>1</sup>Hosted by Digital Ocean: <https://www.digitalocean.com/>

<sup>2</sup>Github project here: [https://github.com/ageitgey/face\\_recognition](https://github.com/ageitgey/face_recognition)

<sup>3</sup>Implemented by Bigchaindb: <https://www.bigchaindb.com/>

# Description

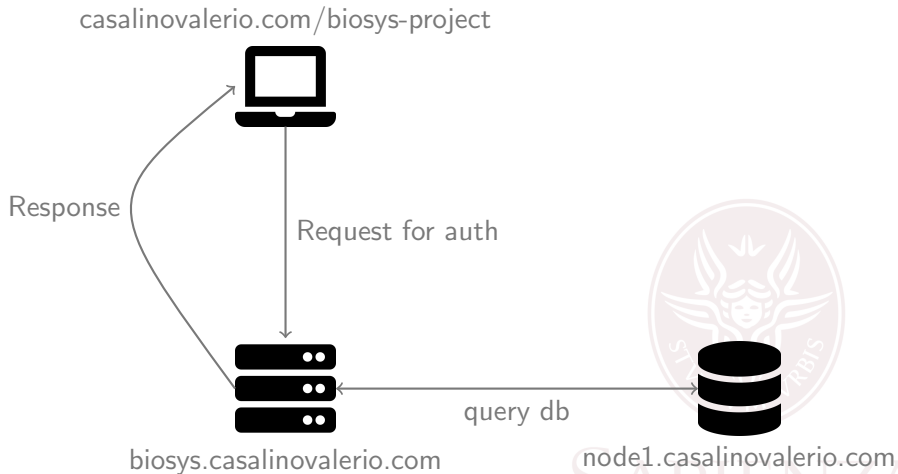
We built a face recognition service based on **identification**, which matches the probe with an **open set** approach.

The gallery is stored in a **Blockchain database** and queried each time, but one of the nodes is stored in the same place in which the request is made, that makes the whole process **faster**.

Each match, if there is one, is returned with a **distance** parameter.



# Overview scheme<sup>4</sup>



<sup>4</sup>Icons are licensed under CC-BY 4.0. <https://fontawesome.com/license>

# Table of Contents

General Concepts & Decisions

**Front-end Implementation**

Data-set Management

Biometric Scanning Integration

Performance Assessment

Conclusions



SAPIENZA  
UNIVERSITÀ DI ROMA

# The web Application

The web application perform the authentication as follows:

- ▶ Captures frames in a canvas.
- ▶ Analyzes them through the opencv's javascript<sup>5</sup>.
- ▶ As the time one button is pressed, the canvas frame is sent to our server<sup>6</sup>, which can register the face, or match the face with an already registered user.
- ▶ In the end, you can be registered, or you can get the authentication response status.

---

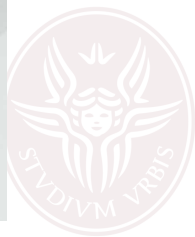
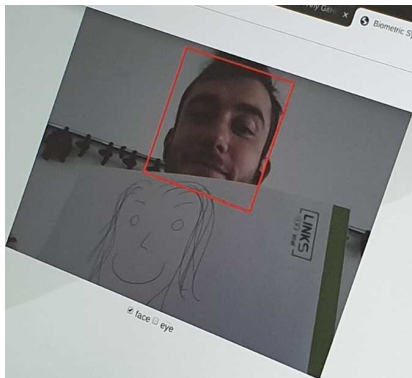
<sup>5</sup> <https://tinyurl.com/s2yprk7>

<sup>6</sup> <https://biosys.casalinovalerio.com>



# Does it work?

We actually did some **serious** testing on it. As you can clearly see in the picture below, it works!



# Table of Contents

General Concepts & Decisions

Front-end Implementation

**Data-set Management**

Biometric Scanning Integration

Performance Assessment

Conclusions



SAPIENZA  
UNIVERSITÀ DI ROMA

# The Block-chain database

As database for new faces, we implemented a **Block-chain**.

We used an open-source implementation of it, called BigchainDB<sup>7</sup>.

We also used Docker<sup>8</sup> to deploy 4 containers running the application.

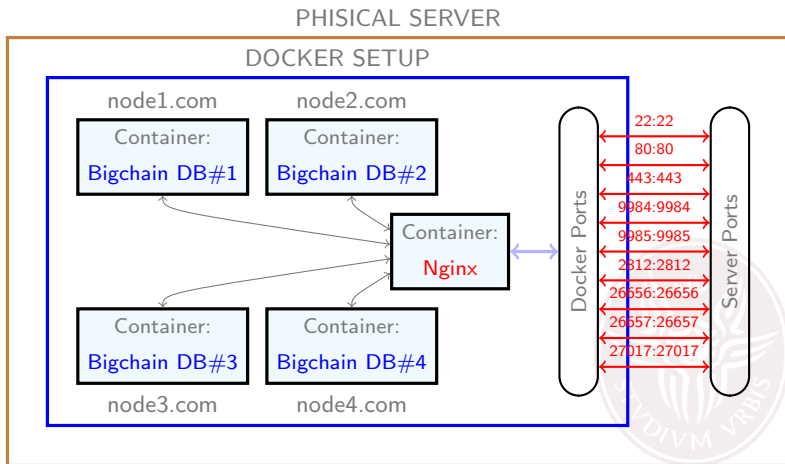


---

<sup>7</sup>Main page: <https://www.bigchaindb.com>. Documentation [here](#).

<sup>8</sup>Main page: <https://www.docker.com>.

# Architecture Implementation<sup>9</sup>



<sup>9</sup>This is absolutely not meant for a real deployment!!

# How to interact with the DB

We are assuming that we have an established connection set up.

## Query data

```
connection.searchAssets('AwesomeAsset')  
.then(assets => console.log('Found assets:', assets))  
// Read the console to look at the assets
```

## Load data (make a transaction)

```
// Create transaction first (txTransferBob)  
driver.Transaction.signTransaction(txTransferBob,  
  alice.privateKey);  
conn.postTransactionCommit(txTransferBobSigned);
```

Simple as that...

# Table of Contents

General Concepts & Decisions

Front-end Implementation

Data-set Management

**Biometric Scanning Integration**

Performance Assessment

Conclusions



SAPIENZA  
UNIVERSITÀ DI ROMA

# Connecting the Web app to API server

## Send faces function in web app

```
...  
var canvas = document.getElementById("canvasOutput");  
picture.src = canvas.toDataURL();  
...  
xhr.open('POST', 'url/send-faces.php', true);  
...
```

## Recognize face function in web app

```
...  
var canvas = document.getElementById("canvasOutput");  
picture.src = canvas.toDataURL();  
...  
xhr.open('POST', 'url/reco-faces.php', true);  
...
```

# Table of Contents

General Concepts & Decisions

Front-end Implementation

Data-set Management

Biometric Scanning Integration

**Performance Assessment**

Conclusions



SAPIENZA  
UNIVERSITÀ DI ROMA



# Testing approach (Training)

We trained the system registering 8 people:

Real name	id	#samples in gallery	#probes
Bill Gates	bill	60	10
Barak Obama	doubleb	60	10
Margot Robbie	harley	60	10
Miriam Leone	mlion	60	10
Scarlett Johansson	redlucy	60	10
Robert Downey Jr	robman	60	10
Tom Hardy	tommy	60	10
Mark Zuckerberg	zuck	60	10

## Testing approach (Querying)

Then, we collected 10 other random pictures from the same people, plus 10 pictures for other 5 people:

- ▶ Jimmy Fellon
- ▶ Donald Trump
- ▶ Tim Cook
- ▶ Jeff Bezos
- ▶ Alfred Hitchcock

To be quick, we used yet another script<sup>10</sup> to upload pictures to be queried.

### Use the script

```
./test /path/to/test-faces http://server.com /path/to/results.cvs
```

---

<sup>10</sup>This one: <https://git.io/JvsQv>

## Results (raw)

We let the script run and the results can be seen in this spreadsheet:



Google Sheets

<https://tinyurl.com/rbk4dnc>

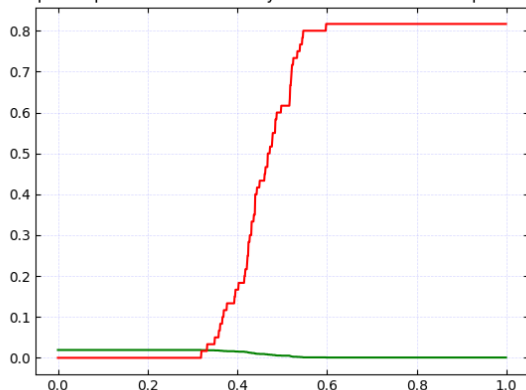


SAPIENZA  
UNIVERSITÀ DI ROMA

# Results (Description)

We analyzed the results with a python script<sup>11</sup>, and obtained these FAR and FRR:

FAR (green) & FRR (red)  
Graphic representation of the system with threshold step of 0.001



<sup>11</sup><https://git.io/Jvsp6>

# Results (Description)

We surely had more to test, because seems too good to be true...

That's why, instead of use our home-made data set, we used the Labeled Faces in the Wild<sup>12</sup>.

This time,

---

<sup>12</sup><http://vis-www.cs.umass.edu/lfw/>

# Table of Contents

General Concepts & Decisions

Front-end Implementation

Data-set Management

Biometric Scanning Integration

Performance Assessment

**Conclusions**



SAPIENZA  
UNIVERSITÀ DI ROMA

# Conclusions

It wasn't an easy project, but we did our best. At the end of it all, we learned:

- ▶ How to gather information on standards and OS solutions available.
- ▶ Deploy an actual service relying on ourselves.
- ▶ How to evaluate a BS performance in an automatic way.

Performance consideration here.



## Greetings from the group

This is a really great ending message from the "creative" chilled-capibaras!



And this is a real cool catchy phrase!!