Communication Systems
Semester Project
Bachelor Semester 6 - Spring 2022

Secure identification of users in platforms based on non-fungible tokens (NFTs)

Student: Clément Sanh

Supervised by: Changsheng Gao
Prof. Dr. Touradj Ebrahimi

June 15, 2022

Multimedia Signal Processing Group
EPFL

Abstract

The birth of Non-fungible tokens (NFTs) solved the problem of verifying the ownership of digital assets through tokens stored on the blockchain. The NFT marketplace is developed around NFT platforms where artists and NFT owners can buy and sell the proof of ownership to an asset. However, these platforms are introducing new challenges for the anti-money laundering (AML) regulators, as the anonymity provided by the blockchain build up interest to develop illegal activities. Furthermore, the absence of KYC to verify the ownership of an NFT opened up the possibility for individuals to sell the ownership to goods that wasn't their own, possibly representing up to 80% of the current NFT production. However, current NFT implementations are inefficient, requires human intervention, unsafe or have horrendous user experience. The aim of this project is to get familiarized with the world of NFTs and their applications in order to understand the specific needs of these platforms related to user identification solutions. Understanding the state-of-the-art in user identification solutions will be required to propose a solution meeting the current implementations with the specific requirements from NFT platforms.

Contents

1 Introduction

These past few years, Non-Fungible Tokens, or NFTs gained significant attention from both specialists and enthusiasts. Professionals increased financial investments for the research through partnerships and assisted the development of innovative start-ups. Enthusiasts transformed the initial spark into a trend that peaked last year, making NFT known to the general public. Within less than a decade, the market went from non-existing to one of the most hyped. Reaching over 17 billion USD worth of trades, the NFT market was forced to structure itself fast around platforms and marketplaces, shaping the evolution of this business. What started as a technological breakthrough became a serious alternative to classical financial investments and offered a new way to collect art and digital assets. Similar to the way music and video streaming revolutionized the way we consumed those media, NFT could very well be the next revolution in the way we purchase digital assets.

However, NFTs remain a fairly new field of research and a lot of challenges are yet to be overcome. Solving these challenges is crucial to secure a prosperous future while ignoring them could bury this as a temporary trend. The following challenges can be decomposed into five main categories: usability, security, governance, extensibility[1] as well as tackling environmental issues. In this project, we focused our attention on solving security and privacy issues by studying the question of identity proofing in the context of NFT platforms. As the NFT market gained success as an alternative to traditional investments, it also became a hotbed for money laundering and illegal financing. To tackle these issues, the traditional banking system used user identification solutions also known as KYC-AML (Know Your Customer - Anti Money Laundering). KYC is a process by which the authoritative entity collects and validates information about the applicant before verifying that the information belongs to the applicant. Classical methods to perform such KYC require the user to provide an official identification document with a picture of themselves.

However, most NFT platforms currently don't implement any identity proofing themselves and most implementations are sub-optimal, either being easily tricked, proposing a bad user experience, or necessitating the intervention of humans in the process. Proofing identity is crucial to enable trust for the digital services, however, as current solutions are bad, most platforms prefer not to implement them. By offering a better solution, we believe more platforms would be interested to implement them. Furthermore, since most of the academic research focuses on finding a more efficient blockchain to tackle environmental pollution and finding solutions to reduce the gas prices when minting NFTs, user identification solutions were left out by most researchers.

In this project, we worked on an authentication process matching faces from a selfie with a picture of an identity document. The implementation can be decomposed into two major steps. First, we worked on a face detection algorithm, that would detect faces from a picture. Indeed, face detection will be required to extract the face from the selfie and the identity document. Once the two faces have been retrieved, we perform the second major step: a face comparison algorithm to verify whether the two faces correspond to the same person. Once, this identity proofing solution was completed, we implemented the state-of-the-art KYC solution.

Then, over the state-of-the-art solution, we implemented a layer of security in the form of a liveness check. The goal of this step is to verify that the user doing the identity proofing is actually in front of the camera and not simply a still picture, that could be fake or stolen. To achieve that, we ask the user to perform random actions in front of the camera. Using the live feed, we will automatize the process to verify that the user is performing the tasks. The rest of the report will follow the development process of the project and explain the thoughts happening to develop a secure user identification process

2 State-of-the-art

2.1 Non-Fungible Tokens

2.1.1 Tokens

To define what are Non-Fungible Tokens, we will start by defining the words composing the expression. Tokens are “a thing serving as a visible or tangible representation of something abstract” [2]. This definition places a token as an asset that can embody abstract concepts onto the physical world under a visible or physical form. This definition matches the definition of a diploma, which is the physical representation of the theoretical knowledge of its owner. This choice of definition is broad, however, we can restrain the definition of the token to the domain of NFTs.

Tokens can also be defined as "an asset digitally transferable between two people issued on a blockchain"[3]. This explanation of the word token focuses on the idea of a token as a tradeable element that could be exchanged between users. By putting the two definitions together, tokens can be defined as digital assets embodying something abstract that could be transferred on a blockchain. These tokens could be used to manage the relationship of ownership, the concept of ownership being abstract and where transferring the token would be the equivalent of transferring the ownership to someone else.

2.1.2 Non-Fungible

Fungibility is the property of being interchangeable, meaning that we could substitute a certain asset or good with another asset of equal value and of the same type. Fungible commodities are commonly used for exchange and trade, making the process easier as the assets can be interchanged. For example, physical money is fungible as we can convert a one-dollar bill into another dollar bill, four quarters, etc. Similarly, cryptocurrencies are also fungible since each Bitcoin can be exchanged with another Bitcoin.

By taking the contrary, a non-fungible asset is an asset that is unique, irreplaceable, and non-interchangeable with another of the same kind. Each instance of an NFT is uniquely distinguishable and cannot be exchanged like-for-like, making it adapted for ownership and non-fungible asset identification.

2.1.3 NFTs as a whole

A non-Fungible Token is an alternate form of cryptocurrency that emerged theoretically in 2014 and was first widely implemented in 2018 thanks to the development of a new Ethereum Improvement Proposal (EIP)-721[5]. When cryptocurrencies like Bitcoin are standard coins that are indistinguishable from another, NFTs are cryptographic assets that "cannot be traded or exchanged at equivalency"[4]. Therefore, they are suitable to identify and represent physical, digital assets, or intellectual property in a unique way. More precisely, NFTs are "unique date records containing a verifiable reference to an asset. The record is stored on a ledger in the form of metadata and conditions of the transaction, which could be distributed or centralized."[6] The uniqueness and the non-fungible properties of the NFTs makes them eligible to be used for proof of ownership. The field of use cases is extremely wide and could be used to prove the ownership of assets like digital art, videos, physical paintings, 3D images, music, gaming collectibles, etc.

Table 1 presents a few examples of tokens separated along with their interchangeability and their tangibility.

	Fungible	Non-Fungible
Physical	Currencies (eg: dollar bills), Gold...	Car, Real Estate, Physical artpiece
Digital	Company shares, Cryptocurrencies	NFTs

Table 1: Comparing tokens using examples for digital and physical assets

2.2 Technical description

2.2.1 Smart Contracts

Smart contracts are transaction protocols whose goal is to execute and control actions according to set rules. Smart contracts are used to overpass the necessity of having an authoritative third-party that may cause additive costs as well as increase the risks of leaks and errors. Ethereum developed smart contracts in the blockchain system very early and blockchain-based smart contracts started flourishing. Exchanges between strangers without a trusted intermediary have been enabled thanks to this new technique.

NFTs are smart contracts linking a cryptographic token to a digital or physical asset. Indeed, the goal of the NFT is to create a certificate of ownership that could be traded between an artist and the client without necessitating the intervention of a specialist like in traditional art. In our case study, when a creator decides to create an NFT by minting it, he creates a smart contract that will be stored on the blockchain. It plays the role of a sale agreement between the seller and the buyer, storing in a transparent and immutable manner the information indicated in a transaction. To keep the interoperability between the different NFT marketplaces, smart contracts agreed to follow the same standard: ERC-721. However, to overpass the limitations enforced by this standard, the token contains new fields in the form of metadata to store additional content. Once it has been entirely written, the smart contract can then be deployed on the blockchain and the NFT can be minted. After the creator did mint his NFT, he then can transfer or sell it to anyone else by recording the transaction on the blockchain.

As shown in Figure 1, the process of creating an NFT result from the interaction between the user side where the requests and the parameters are sent, and the server side where the verification and the storage of the tokens are done. Moreover, the figure made explicit the storage schema showing that, since storing data on the blockchain is costly, the content of the digital asset as well as the metadata as stored in external storage. Only the smart contract is stored on the blockchain and is linked with the rest of asset with a robust link.

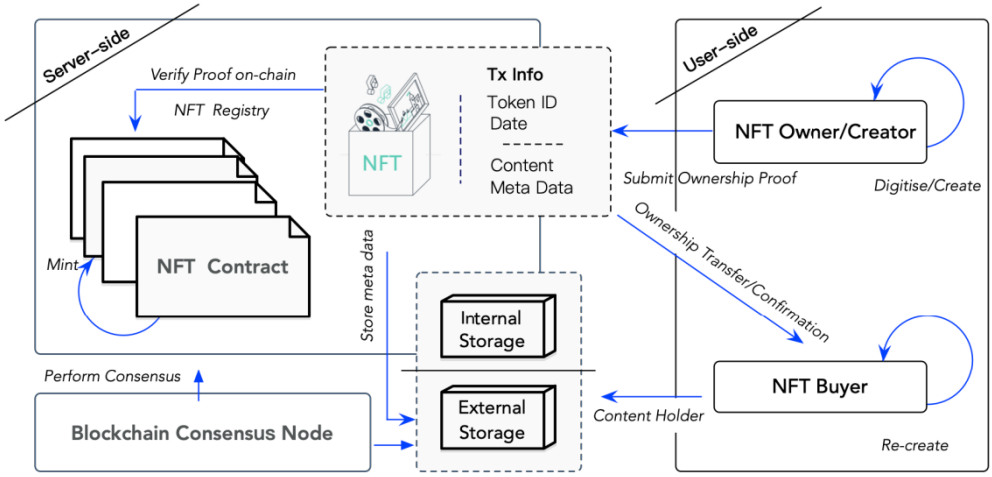


Figure 1: Workflow of NFT Systems[1]

2.2.2 Blockchain platforms, differences

Another building block of non-fungible tokens is known as blockchain. The blockchain corresponds roughly to a distributed database maintaining the transactions records. It is the mechanism assuring that the information about the transfer record of a given NFT remains secure and can be maintained and accessed without requiring any centralized trusted entity. The choice of blockchain is crucial and impacts directly properties of the smart contracts.

NFTs were created thanks to the creation of a new standard of the Ethereum blockchain allowing the implementation of smart contracts. However, since their creation, a multitude of different blockchain platforms have appeared. While, the more popular for NFTs remained Ethereum, alternatives started gaining interest thanks to innovative features. Massive improvements are yet to be developed and the interest to find the next Ethereum is at the core of current research. From direct alternatives like Tezos, Solana, or Cardano to layer-2 and layer-3 blockchain platforms, the scope of possible ameliorations is wide.

Most of the new blockchain implementations concentrate their attention to reduce the environmental impact, increase the maximum number of transactions per second, and finally to lower the gas' fees and the cost to mint an NFT. Other elements are being discussed like the choice for the consensus mechanisms which has a direct impact on the security of the blockchain as well as the environmental impact or the choice of programming language that impacts the ease to build applications over it for third parties. Table 2 gives a comparison of different blockchain platforms to see what are these platforms proposing to make them differ from the leader Ethereum.

	Ethereum	Tezos	Solana	Flow	Polygon
Programming language	Solidity	Michelson	Rust	Cadence	Solidity
Consensus mechanism	PoS + PoH	Liquid PoS, LPoS	PoW -> PoS	PoS	PoS + PoH
Scalability (in tps, transactions per seconds)	14.7	100	65k	1k	65k
Transaction cost	20$	0.02$	0.00025$	0.03$	No gas fees
Pros	Largest smart contract platform High security and true decentralization Ability to develop lots of dApps	Highly modular for future improvements LPoS is highly inexpensive in terms of computational resources Fairly low gas fees	Very high throughput and number of transactions Extremely low fees Highly scalable with PoH, Rust popular language	Very promising and high potential Developer friendly with low complexity Convenient and smart contracts are upgradable	High security and true decentralization from Ethereum High scalability No gas fees
Cons	Very high transaction cost Bad environmental impact Very low scalability, too few transactions per seconds	Extremely secure BUT too specific, too complex for normal usages Lack of stability, hard to predict the fees Relatively low scalability	Still mostly in Beta Remains niche and lacks validating nodes Complexity to make smart contracts (in low languages)	Very immature blockchain Too few users Medium tps, not worth changing compared to others	No control over the entire process Dependent on the work of others

Table 2: Comparison between different blockchain platforms (Layer-1 and Layer-2)

Amongst the multitude of blockchain platforms, we can distinguish three categories of blockchain sharing common properties. Those categories are known under the names layer-1 (L-1), layer-2 (L-2), and layer-3 (L-3). The Layer-1 blockchain composes most of table 2 and would refer to the underlying blockchain architecture like ETH for Ethereum, BTC for Bitcoin, or Solana. Layer-1 blockchains are the principal networks within their ecosystem.

Polygon is the only Layer-2 blockchain platform present in Table 2 and is an off-chain built on top of Ethereum. Layer-2 blockchains are protocols that are built on top of layer 1 to improve the original blockchain's functionality and fix some weaknesses. Usually, layer 2 keeps all the advantages from layer 1, usually keeping the security properties but solves the high gas fees and increases the maximum number of transactions per second. In our example in table 2, Polygon takes the advantage of Ethereum's security and infrastructure but increases the number of transactions per second and eliminates the gas fees. Both Layer-1 and Layer-2 blockchains are used by NFT marketplaces even though Layer-1 remains more common. There are a lot of other layer-2 blockchains like Ethereum Plasma, and Bitcoin Lightning Network, however, Polygon remains the most popular, even being available as a blockchain on OpenSea.

Finally, Layer-3 are representing applications based on blockchain, like Defi apps, and games, often regarded as a pair with the perspective of a web 3.0. NFT marketplaces aren't yet using those blockchain platforms to trade the tokens.

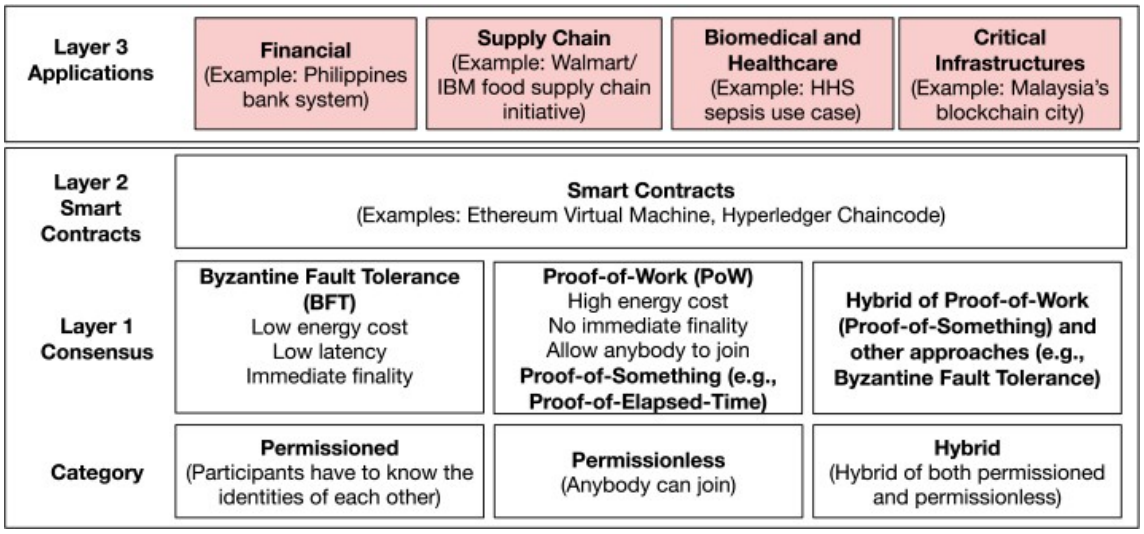


Figure 2: Blockchain layers [8]

2.2.3 NFT Marketplaces

The final component of the NFT environment is the NFT platforms, whose goal is to connect artists and re-sellers with potential clients. These platforms are under the form of specialized marketplaces like OpenSea, Rarible, SuperRare, etc. where the tokens are presented to customers to be exchanged with money in the form of cryptocurrencies. As the NFT tokens are supposedly unique, NFT marketplaces have the responsibility of providing verified unique content to the customers. Thus, it becomes necessary to leverage blockchain technology to verify the origin of the digital content presented on the market. Verifying that the assets proposed are unique is one of the main tasks of these platforms as well as verifying that the artist selling the NFT is in fact, the right artist.

Similarly as for traditional art marketplaces, verifying the authenticity of the products displayed is crucial. For each NFT minted, theoretically, the marketplace should verify the provenance reading the smart contract, verify that the artist selling the art is the one owning the rights to it and verify that it isn't a replica of another NFT posted by another artist. While the marketplaces should do their best to block fraudulent content from spreading on their platforms, they must not act as a centralized authority, or else the decentralized concept of NFTs would be ruined. Finding the right spot to not cross the boundaries is still a topic of reflection and nowadays, platforms are differentiating themselves in their approach to the question.

OpenSea, the leader of the market, chose to stay lax and affirmed that more than 80 percent of the NFTs on their platforms were either spam or plagiarised artwork [9]. On the other side of the spectrum, the NFT marketplace Foundation chose to imitate a classical art gallery where the platform plays the role of the specialist assuring the authenticity of the product and requires the approbation of a member to enter the marketplace. Foundation chose the centralized approach to the marketplace when OpenSea embraced fully the decentralized approach.

As the stakes are getting greater with the gain in interest and capitalization, these marketplaces faced new responsibilities to go against fraud. The entire credibility of the market relies on the security of the marketplace to ensure that the basic properties of NFTs are ensured, meaning that the assets sold are unique and that verified artists shouldn't see their art stolen. Furthermore, the development of illegal financing and money laundering through these means is also reducing the reputation of the entire market. Solving these issues became suddenly a critical key to pursue the expansion of the entire market and installing a user-identification process may help to solve partly those issues. Indeed, verifying the identity of the users may help to trace the origin of the NFT having the identity of the seller and matching it with the identity of the artist and could help us to limit money laundering and illegal financing as the process

wouldn't be entirely anonymous anymore. Yet, this goes against the current decentralization of the platforms and joins the debate of security against anonymity.

3 Identity Verification and User Identification

3.1 Know Your Customer

The gain in popularity of NFTs gained the interest of investors and regulators but also the attention of criminals who saw an opportunity to use the pseudo-anonymity offered by the blockchain to pursue money laundering or finance illegal activities. The risks of seeing money laundering or financing for illegal activities raised the attention of the financial regulators who saw a breach in the system. The same issue was discussed in traditional banking and led to the enforcement of a Know Your Customer (KYC) policy, forcing professionals to verify the identity and the suitability of the person they're having business with.

KYC refers to the obligation to perform identity and background checks on the customers before allowing the user to access the platform. As anonymity is a major element of what makes the blockchain popular, the concept of declining your identity to an entity makes reluctant a lot of users. Therefore, most marketplaces aren't implementing a direct KYC since they aren't forced by regulatory entities. Yet, the anti-laundering entities have an eye on the question and could strike at any moment, thus, acting pre-emptively could ensure that the process is well implemented. By preventing money laundering and limiting the chance of intellectual property theft, the NFT marketplaces would also gain trust amongst their user base and the traditional financial system.

3.2 Identification approaches

There are many different ways to implement a user identification process for an application. In the next part, we're going to develop all the possible approaches to perform a direct KYC, where the platform does the identity verification itself. There exists also two other approaches to doing the KYC that is implemented by a few NFT platforms: the indirect and the hybrid. The indirect KYC corresponds to the situation in which the platform lets another entity perform the registering onto the blockchain and doesn't store or get any information about the user using its platform when the hybrid is the case in which KYC is implemented by the platform but isn't necessary and remains the last case which are prototypes of decentralized KYC. Table 3 shows examples of platforms with different KYC approaches, proving that the market is free to choose the solution they prefer. While some like OpenSea and Rarible prefer to not implement any user identification process themselves, platforms like Polkadot try to innovate by developing a decentralized KYC.

	Examples
Direct	Binance (after Nov. 2021) , Foundation...
Indirect	OpenSea, Rarible...
Hybrid	Binance (before Nov. 2021)
Decentralized	Polkadot

Table 3: Examples of platforms with different KYC approaches

3.3 KYC Types and Applications

Focusing on direct KYC as it is the focus of this project, we can study the different KYC types[7] and enumerate their applications before comparing the strengths and the drawbacks of each type.

3.3.1 KYC Types

- **Paper-based KYC:** In-person form of verification, show physical copies of documents
- **Central KYC:** Having a central repository of KYC records that could be used across different platforms
- **Video KYC:** Paperless and presence-less KYC process recording a video, then match ID with video
- **Biometric and e-ID KYC:** In the case where e-ID and biometric exists, could do a KYC using biometric and the e-ID, paperless.
- **Digital KYC:** Online form of verification, matching live photo of the customer and an identity card

3.3.2 Drawbacks and Strengths

In order to find the optimal user identification solution to start from, we should compare the different KYC types and find the strengths that match our needs. The table 4 presents the drawbacks and the strengths of each type of KYC.

	Drawbacks	Strengths
Paper-based	Cost-intensive, need many employees Vulnerable to operational delays Extremely inefficient	Most traditional Customers familiar Attractive for areas without access to technology
Central	Need to have a central authority Keep an eye on the central database in case of updates Access granted by the authority else impossible	Very fast to access all documents required Data stored in electronic format, no space issue Time-efficient for the users and for the companies
Video	Require heavy investment for companies Need access to device with camera	Almost error-free process Cost and time-efficient for the user Can be automatized, technology can verify if genuine documents
Biometric & e-ID	Need to have e-ID in the country and biometrics Expensive method	No contact with customer needed Using state papers, more secure Fast and efficient
Digital	Verifying the authenticity of a document not manually is very complex In case of manual intervention, inefficient + not error-proof	Paperless Very cheap through automated process Good user experience

Table 4: Drawbacks and Strengths of each KYC type[7]

NFT Platforms constraints In order to choose which KYC type we should implement for our user identification solution, we need to understand the constraints we should take into consideration. Our solution should implement a KYC specialized for NFT platforms, which enforces a certain set of rules. First, the KYC should be an automated process as we want to perform the KYC optimally, eliminating human errors from the process. Secondly, we want to keep a decentralized process that doesn't require keeping all the pieces of information from the users in a big central database managed by a central authority. Finally, we want the implementation to be working globally, by not having country-specific constraints or needing to have a physical encounter. Thus, we want to have a process done entirely digitally that could be implemented no matter the region of the world, only requiring the use of a device able to take pictures and run code.

Using table 4 and knowing the constraints we have for implementing a KYC for platforms based on non-fungible tokens, we can decide which KYC type is the best fit, using its strengths at its full potential while limiting the negative impact from its drawbacks. For the automated process, we can already eliminate the paper-based KYC which required human intervention and also needed physical presence. The decentralized property eliminates the central KYC type as

it forces the creation of a central authority managing the database for the documents. The last property of needing a global solution eliminates the biometric and e-ID KYC type as it works solely in countries where e-IDs and biometrics are available.

It leaves us with the video and the digital KYC that both correspond to what we look for: error-free, efficient, and can be automatized. However, since the video KYC required heavy investments, we decided to mix it with the digital KYC which is way more simple to implement and use for the customers. The only problem with the digital was the possible involvement of a human in the process, however in our implementation, no humans are involved in the process, thus the digital KYC appeared perfect for the project. Furthermore, by doubling the layers of KYC, we are making the process of frauding more complex. Hence why we chose to add another layer of video KYC to accompany the digital KYC as the optimal KYC for our project.

3.4 User Identification Solution

Having all the components to propose a working user identification solution, we need to split the solution into multiple sub-tasks. These subtasks are explicited in Figure 3 which describes the workflow of the solution. To implement a user identification protocol using the digital KYC model, we need to verify that the live photo corresponds to the same person as his official document. To do so, we will need to implement an executable able to record the user and take a picture of his official document. Having the two pictures stored, the goal is to verify whether the faces match the same person. This requires multiple steps first, we need to implement a function whose goal is to detect the faces from the two pictures and return cropped pictures containing uniquely the faces. Then, we need to implement a face matching protocol whose goal will be to take the two faces and return a factor of similarity. Above a certain threshold, we can affirm that the two faces are similar and that the user has been identified. Otherwise, if the factor is under the threshold, then we consider that the two faces are different and the KYC failed.

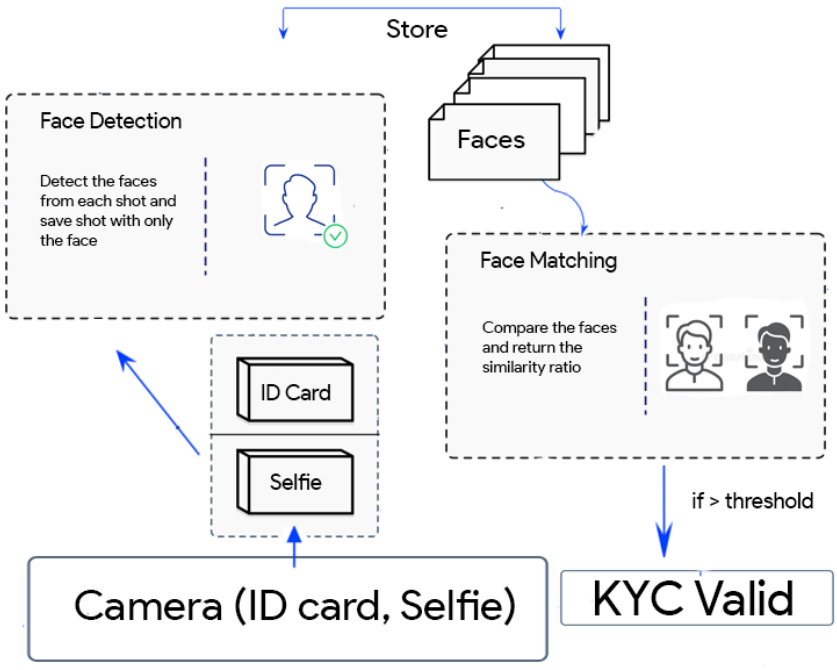


Figure 3: User Identification Workflow

Figure 3 displayed distinctly the three main steps that needed to be implemented: I/O management with the camera creating the two images used as input, then, storing the faces after

their detection, and finally handling the result of the KYC, valid or invalid depending on the result of the face matching phase. The two other steps are face detection and face matching, responsible for the computation of the data and the identification of the user.

4 I/O Management

For the management of the input and the output, we needed to implement few elements making the workflow working.

4.1 Camera

The digital KYC relies on the comparison between facial information from official documents taken from the camera and a selfie taken from a live camera recording. For our implementation, we decided to use the video and the digital KYC at two different moments of the process. The video KYC will intervene at the liveness check while the digital KYC comes at the user identification stage. The user identification step needs to capture a live photo of the user and a picture of one of his official documents.

The initial idea was to create a pseudo camera setup where we could capture multiple shots using the camera of the device. We wanted to make the user able to have a view of the shot and let him take multiple pictures before letting the user choose the one to keep. We used the Python libraries OpenCV and Tkinter to embody this imitation of a camera.

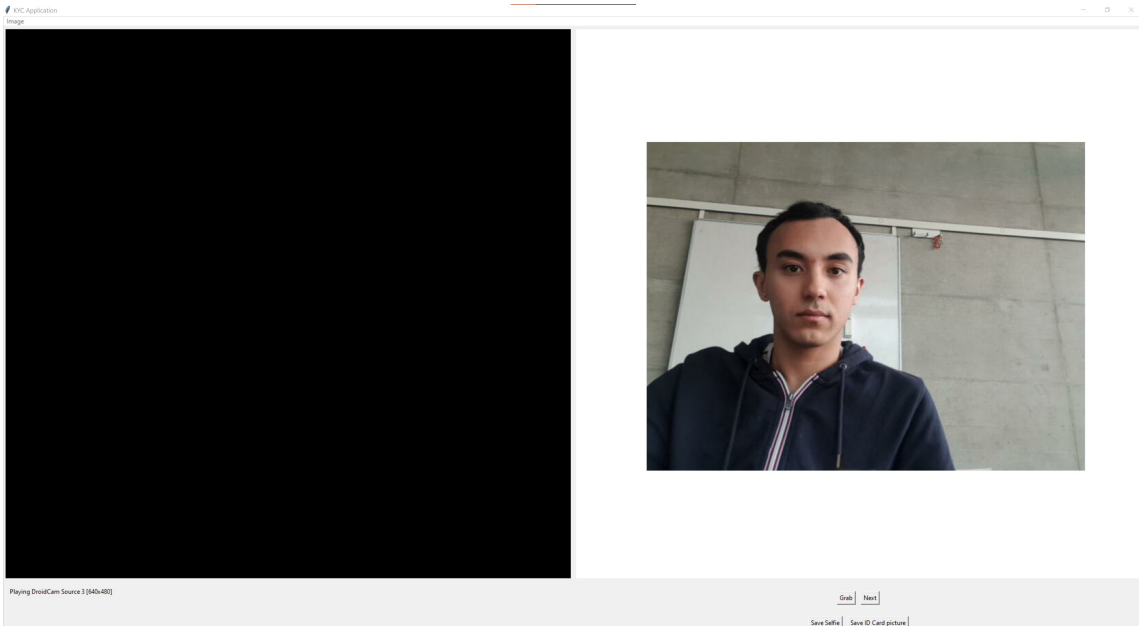


Figure 4: Camera GUI

4.2 Faces and output

The second and third I/O management part of implementation applies when we have the two pictures using the camera we implemented in ***gui/MainWindow.py***. Once these two shots were saved in the ***images*** folder, we then ran the face detection algorithm from which we got the coordinates of the faces in each photo. Then, the function *detectFaces()*, makes us store the faces cropped from the pictures using the coordinates that we computed in the folder ***modified***.

For the output, we simply retrieve the similarity ratio and output True if the similarity ratio is above the threshold and False otherwise.

5 Face Detection

The first major element of the user identification protocol is the face detection algorithm which is a crucial step for the project, that will be used to detect the faces from the two selfies and the official document.

5.1 Cascade Classifier

Given an image, our goal is to retrieve a cropped version of the input image only containing the face. To do so, we need to use a framework able to detect the faces from a shot and return the coordinates of the face in question. Then, having the coordinates and the initial image, we could crop the image at the given coordinates and recover the face as the final image. The choice of the framework to use for our project depends on the properties we want to defend.

For our project, the goal was to achieve a relatively fast image process to deliver a good user experience and give a product that requires a relatively low amount of resources to make it run on all types of devices. The OpenCV's solution for handling face detection seemed to fit the requirements we gave, therefore we decided to use it as our cascade classifier.

5.1.1 Haar-cascade Detection

OpenCV's classifier is using a Haar-feature-based cascade classifier to detect the faces. The Haar-based cascade classifier is real-time face detection that was theorized by Paul Viola and Michael Jones in "Rapid Object Detection using a Boosted Cascade of Simple Features" published in 2001[10]. The method consists of a machine learning process that will train a cascade function using a large dataset of images with faces and others without faces. The Haar-cascade model works for all types of elements, it requires a lot of positive and negative samples to train the model. For every case, once the training phase is complete, the model can be used to detect objects in other images.

Therefore, in our case for face recognition, the classifier will be trained by a large dataset of images with faces and others without from which the classifier will extract features. The extraction is completed using Haar features that take three different forms (see figure 5). All of those features are based on the separation into two areas: the white surface and the black surface. The feature is then the value obtained by subtracting the sum of pixels under the white rectangle from the sum of the ones under the black rectangle.

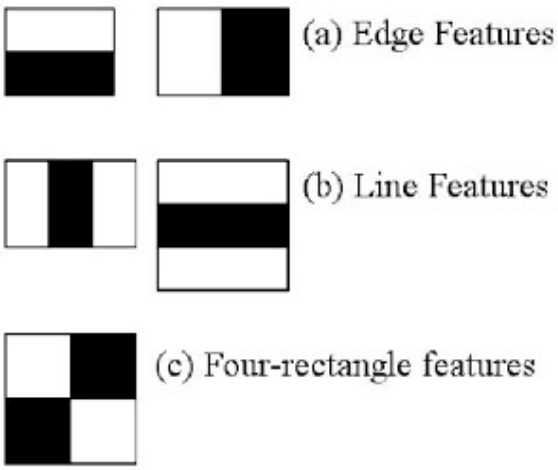


Figure 5: Haar features from the OpenCV documentation [10],[11]

When training the model, we compute these substractions for rectangles of all sizes and at all possible locations to detect patterns of features appearing on faces. Most of the features that

will be calculated will be irrelevant, however, when training over a large dataset of positive and negative images, we will recognize patterns in positive samples. For example, the region of the eyes is notably darker than the bridge of the nose and the cheeks (see figure 6).

Once we have detected such fundamental patterns that make faces recognizable, we need to find the optimal threshold for these features to classify the faces as positive to limit the error rate. After having trained the model, when we will send a picture to detect the faces, the classifier will try to detect such features and return the result consequently.

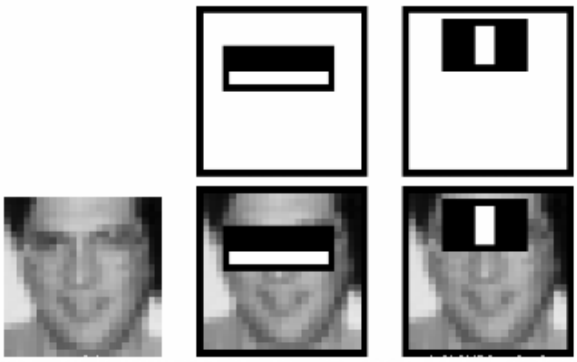


Figure 6: Examples of relevant Haar features [10], [11]

5.1.2 Detecting faces

For our project, we thus used the implementation developed by the OpenCV library retaking the theoretical work from Viola and Jones. The function *detect_faces()* relies on the method *detectMultiScale()* which computes the face detection using the Haar-cascade method. Since the Haar-method is used over images in levels of grey, we needed to convert our image using the method *cvtColor()*. We then used the function detectMultiScale initialized with a cascade classifier trained on the dataset of faces. We played with the parameters to only keep the main faces while eliminating faces that could potentially appear the background. We also tried limiting the possible errors from the models by requiring a minimal number of 5 neighbours for each candidate rectangle, which requires to have five neighbours having a verifiable feature, eliminating possible misreadings. Then for all the faces we get from the picture, we store their faces inside the folder ***modified***.

5.2 Evaluation of the results

To evaluate the results from the face detector, we can use the testing results from testing sets that affirm to reach a success rate of around 90 to 95%. Moreover, for the speed of the detector, the initial creators estimated that the time to detect a face was less than a tenth of a second which is more than acceptable for our requirements.

With our personal experience, the face detection works extremely well and it worked fine under low exposure, without the entire face and yet it arrived to detect the face almost every time. However, errors still happened with faces not detected while they were present but it was very marginal. The case of high exposure was more complex as the classifier based itself on the difference of exposure but with extreme lighting, these differences of shadows could be erased by the too high exposure. As such, these edge cases could be a limitation to the user identification process. However, the security of the process remain clear as in case no faces are detected, the identification failed, thus blocking the user from reaching the market. This remains an issue that could penalize normal users to pass the user identification process, yet it isn't a critical issue, therefore we kept this implementation.

(a) Original shot (b) Face extracted

Figure 7: Original shot side by side with the face extracted

6 Face Matching

The second major step of the implementation of a user identification process is face matching, where we will use stored faces to compare them and verify that they belong to the same person. To perform such a task, the end goal would be to compare two images by simply having a similarity ratio that would mathematically return how close two faces from each other. We could imagine being able to subtract two images using Euclidean distance and find a value corresponding to the difference between two images. However, to implement such an idea, we would need to transform the images under something that could be compared using Euclidean subtraction.

More generally, in order to do the face comparison, our implementation relies on having one image as a reference from which we will register the faces. Then, we take the second image and retrieve all the faces from it to compare them to the ones stored as references. Therefore, in case the picture contains multiple faces, the program will do the comparison for all faces within the shot. The comparison will be completed using the Face Recognition API from PyPI, which contains a function to compare faces and return the distance between the two faces. This step of the process is managed in the class ***compare_image.py***.

6.1 Face Registration

The first step in order to make that Face Recognition API work is by having the two inputs ready and then we first input the first image into the registration step. During this step, we need to call the function *scan_known_people()* where from there we will add the faces to the database associating the name of the file in with the faces that we retrieved from it. The face registration phase is trivial, we initialize a list of names where we'll add the name of the file and a list of faces that we're going to fill with the faces that we retrieved from the image, only keeping the biggest face from the image as we assume that we compare the main subject from the original picture.

6.1.1 PyPI Face Comparison

The comparison of the faces done by the PyPI implementation requires converting an image into a 128-dimension face encoding for each face in the given image. Before comparing the faces, we, therefore, need to convert our images to this encoding using the *face_encodings()* method. With this encoding, we can calculate directly the norm of the difference between the encoding of the registered face and the encodings of the faces from the image to compare it.

This comparison is possible thanks to the pre-trained convolutional neural network that worked on recognizing facial elements in order to generate 128 measurements for each face at

set positions. These positions include measurements for the eyes, the chin, the mouth, etc. and are pre-trained thanks to PyPI's implementation. Their training method used a Deep Convolutional Neural Network, used to train it to generate those 128 measurements for each face. This neural network was trained by looking at three face images at the same time, two images of the same person and one of a totally different person. The algorithm will first generate ranom measurements for each of the three images and will tweaks the neural network so that the measurements generated should be close between the two pictures of the same person and different for the other face [12]. This step will be repeated for a large dataset of images of thousands of different people so that the neural network can reliably generate those 128 measurements for each person.

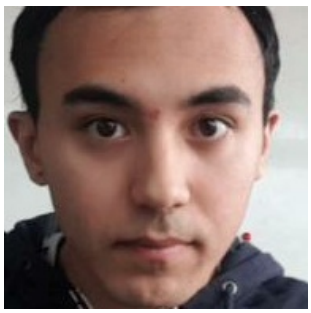
Finally, having those 128 measurements for both of our pictures, we can verify compute the Euclidean distance between each of those measurements and if the distance remain under a given threshold, then we consider that the two images relate to the same person.

6.1.2 Evaluation of the results

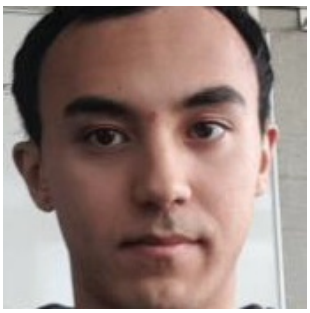
The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark, which shows that the implementation is very efficient. Similarly, every time I used twice the same face, the values were under the threshold, making it valid. When using different faces, the test also outputted that the similarity ratio is above the threshold, confirming that the two faces are different.

However, we need to nuance as both shots were always taken in the same context, having the same camera, the same timing, lighting for the ID card shot and the two selfies. Moreover, after more manual testing, it seemed obvious that the shots needed to be relatively close in terms of angle with the camera as the implementation do not take into account the possibility for the user to be profiled. As such, when doing a profile shot with the face shot from the ID card, the result is usually false, as the 128 measurements cannot always be matched on the image.To counter the issue of the profile shot that could be taken, we could implement a verification for the shot where the user is facing the camera to confirm that the process detects a face before saving the picture.

(c) Face from ID document



(d) Face from selfie



(e) Face extracted from video shot

Figure 8: All three faces that need to be compared

```
Processing..... C:\Users\sanhc\KYC-NFT-Semester-Project\images\card.jpg
Processing..... C:\Users\sanhc\KYC-NFT-Semester-Project\images\face.jpg
Processing..... C:\Users\sanhc\KYC-NFT-Semester-Project\images\selfie.png
0.24310896757225703
0.5932303836825736
True
```

Figure 7: Results for the comparison between the Selfie and the video shot, Results for the

comparison between the ID document and the selfie

7 Secure Identification

7.1 Background

NFT platforms' anonymity is at the same time a blessing and a curse. In fact, it provides the freedom for everybody to purchase or sell art equally, without needing to be a known artist but it also opens the door for criminals to finance their actions through these marketplaces. Implementing a user identification policy to block the malicious users from being anonymous and thus unreachable is the solution to block such drifts. However, the vast majority of platforms decided to not implement such security, and to convince them to implement it, it would be necessary to have a convincing solution. Some of those platforms are against the idea of losing the notion of anonymity characteristic of blockchain applications and some others are simply reluctant to implement a functionality that isn't mature yet. Indeed, current implementations are extremely vulnerable to multiple attacks where malicious users could enter the system without having to put too much effort while making the process heavier for the rest of the active users.

As such, our proposition is to make the KYC process more robust and secure so that platforms may be more interested to implement it in their applications. This new layer of security tackles the issue of liveness which is a concern in information security as secure transmissions shouldn't be tampered with by altered or replayed information. In the case of a KYC process, it is crucial making sure that the person doing the KYC is a live person and not simply someone else trying to sign up by impersonating someone else or using AI-generated faces. In the simple case of the user identification process that we implemented, a malicious user could bypass the system simply by having a photo of the person he wants to impersonate as he could simply use this photo and tamper with the official card to display the same image on it.

In order to limit the possibilities for attackers to abuse the system as they currently can, we decided to implement a liveness check before the user identification. This liveness check will make sure that the user in front of the camera is live, thus blocking the attacks from the user only having a photo of the person they want to impersonate.

7.2 Liveness Check

Some implementations of the liveness check use a single shot and try to distinguish patterns that would show that the face was live, however, this solution is very sensitive to the use of deep fakes as the attackers could use a single shot of AI-generated face to pass the liveness check and then do the KYC as easily as before. Our goal is to make the process of overpassing the KYC for an attacker as hard as possible which is why we decided to use the benefits of the video KYC for the liveness check.

7.2.1 Theory

Therefore, our implementation works using a live video recording in order to realize the liveness check. To verify that the user performing the liveness check is live in front of the camera, we implemented a challenge requiring the user to perform a certain number of actions. Those actions will be randomly chosen out of a pre-determined set of actions. While the user will perform those tasks, computation will be made in the background to verify that they are realized by the customer. For example, if the user is asked to smile, the program will verify that the live feed of the customer corresponds to the shot of a person smiling. Forcing the user to perform a certain number of actions live in front of the camera blocks malicious users from attacking the application with a single face as the attacker would then need to have faces corresponding to all the different actions asked to pass the liveness check.

The strength of this liveness check comes from the modularity of the implementation as by simply modifying two parameters we can make the verification way harder as we could modify the number of actions needed to be performed to pass the test or modify the time slot available to perform these actions. By playing with these parameters, the authoritative entity can make the process way more complicated to pass for malicious users but to the detriment of the average user experience.

Finding the right middle-ground where malicious users would find the requirements to overpass the security too expensive while the experience remains positive for the customers. To facilitate the understanding of the average users and to solve the main issue advanced by the skeptics of the liveness check, we decided to implement a home page where the users will be presented with explanations on the entire process of the secure KYC. Psychological acceptability on the side of the users is crucial to enforce a secure application as the users will be more understanding of the actions they'll be asked to perform.

7.2.2 Possible security breaches and solutions

However, even though this implementation offers the solution to some problems previously mentioned and may reduce the number of malicious users trespassing the rules, there remain a lot of possible security breaches that we have to mention and deploy solutions to such threats.

- Malicious users could do the liveness check themselves and then complete the user identification using the tampered pair of faces. We thought about this issue very early when developing our theoretical implementation. Fixing this issue was fairly trivial so we decided to keep our theoretical idea. Indeed, to block this attack, we could take a picture from the live video feed to then compare it with one of the photos used for the KYC. As such, we are simply re-using techniques already implemented while making sure that the user doing the liveness check is the one doing the KYC and blocking such an attack. Nevertheless, solving this issue wasn't completed as we needed to discuss the way we chose the moment to take the shot from the video.

  Here, we faced a dilemma, by taking a shot at a random, we were forcing the user to do the liveness check himself as he couldn't know when the picture would be taken.

  However, this comes with the risk of having a picture in which the user isn't facing the camera or not even in the shot. While taking a random shot would remain the best solution to block any possible attack as the user cannot act in consequence, it seemed counter-intuitive to make the output depend on the moment the shot was taken. Moreover, even if we want to offer a secure implementation, we shouldn't sacrifice the global user experience. As such, we decided to add a new state during the liveness check in which which the user will need to stand "normally" in front of the camera. This state can also happen at any moment during the liveness check.

- The user could do the liveness check, then when asked to stand face to the camera, he could use the fake face.

  With a new solution comes new issues, the attacker could wait until the moment of the "normal face" to use the same fake face that he will use to overpass the user identification process and then go through the liveness check himself. Therefore, the attacker would pass the liveness check as himself and still have the right fake face for the final comparison. After a lot of reflection, we didn't find any successful solution to block this security breach optimally. One solution to limit the impact of the breach would be by reducing the time slot allocated to take the normal face so that the malicious user may not have the time to

change its face before the shot. Once again, this may damage the experience for normal users but would make the breach more complex to use.

- The user could use video deep fakes where a mask is applied over the user's face while keeping the face movements.

Unfortunately, deep fakes attacks are getting popular and more and more efficient. Researchers are working hard to find effective solutions to detect those types of attacks. The race between researchers and developers of deep fakes to keep a lead represents well the issue for the KYC implementations.

For example, one great solution to block deep fakes would be to use depth sensors like Apple's Face ID that cannot be fooled by such attacks. However, most webcams and smartphones don't have access to such technologies. As our KYC application should be able to be used on all types of devices with a working camera we aren't allowed to use such techniques. Trying to find solutions to these attacks will be key in the few years to ensure the robustness of the KYC.[13]

7.2.3 Implementation

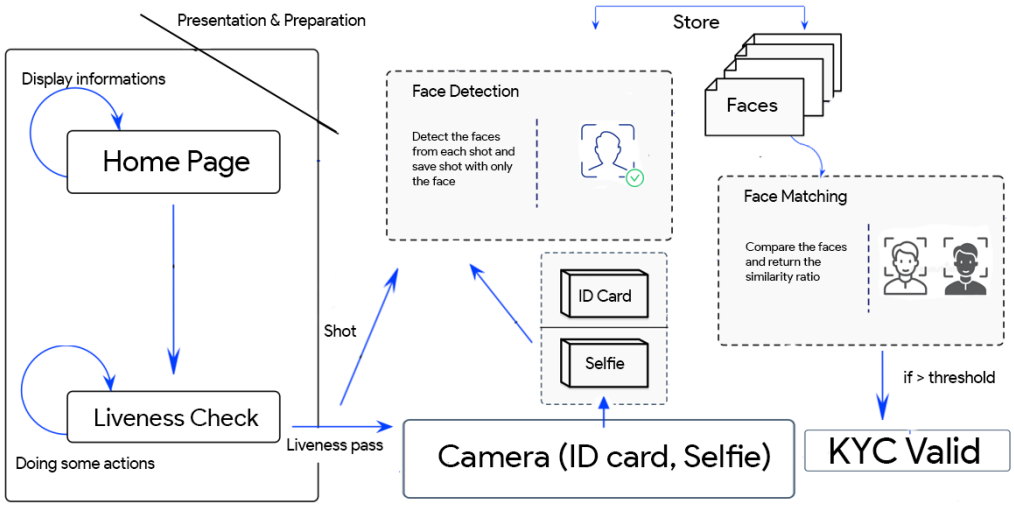


Figure 7: Secure User Identification Workflow

The implementation for the additional security layer corresponds to the addition of two new steps in the process. Consequently, before the previously implemented user identification solution, we implement the liveness check as a necessary condition to pursue to the next step. To implement the liveness check, we again need to access to the live feed of the camera in order to verify that the user is performing the actions we're asking him to do. The code corresponding to the security implementation is managed in the file ***face_ anti _ spoofing.py***.

Randomize questions The first step of the implementation consists of generating the random set of questions that the customer will need to perform. The set of questions is defined in the file ***questions.py*** and is composed of six different actions that can be asked to do.

These actions are smiling, surprise, blinking eyes, being angry, turning face left, and turning face right. During the liveness check, the questions will be selected randomly out of these six possibilities and displayed to the user. For each question, the user will have a determined number of seconds to accomplish the task. The time allowed per question can be modified by changing

a simple parameter. Lowering the value of this parameter may reduce the possibility for the attacker to perform his trick but could also block other users to perform the liveness check normally.

Similarly, the "normal face" status may appear randomly as one of the actions but has a separate timer. Doing as such, offer the possibility for the authoritative entity to modulate the time of the critical step independently of the rest of the steps. Making the timer for this step shorter makes more sense as no background computation is required and reduces the length of the critical state. To validate the liveness check, the user will have to validate *limit_ consecutive* questions in *limit _ questions* number of tries. These parameters are easily modifiable and could again set the level of difficulty required to pass the test.

Detect Liveness For each action, background computation will happen simultaneously with the live feed being displayed. At regular time intervals, we are going to take a capture of the live feed and use this image to compare it to the results from pre-trained models to verify whether this image does correspond to the expected results for the given action.

We used pre-trained models specifically for each action with pictures of users doing the action and others that don't. Each of these images will pass through three predictors that will estimate respectively the orientation of the face—useful to verify the actions of turning the face left or right—, and the emotion of the user—useful for checking if the user is angry, surprised or smiling—, and the final one to detect the eye blinks.

Face Orientation For the first detector, we worked on detecting the orientation of the face. To do so, we first used a cascade classifier from OpenCV using a trained model on profile faces. Using this pre-trained model, we call the function *detectMultiScale()* that will return the position of the faces detected by the trained model for a given shot as well as the level of confidence it has with it. Similarly, to the way we used the same method to detect faces in the case where pictures are facing the camera, we are now using the classifier trained using profile pictures. The dataset used to train the classifier used left-sided profiles.

Consequently, when once the classifier has been trained, if we send a picture to it, it will detect a face in case the user turned his face to the left. If the photo didn't detect any faces, we flip the photo horizontally and restart the same process. If once again, the classifier doesn't find any faces, we could then consider that with a high chance, there is no face in front of the camera or someone is facing straight into the camera, in both cases, the liveness condition hasn't been verified. However, if after being flipped, we recognized a face, then we can conclude that the user was turning his face to the right.

Therefore, depending on the moment we found a face, we can return the direction of the face and return null in case we didn't find any face turning left or right. The trained model was provided by David Bradley from Princeton University. [14].

Emotion Detection For the second detector, we worked on discerning the different possible facial emotions. Here, again, for every shot, we use the coordinates of the faces to extract the face from the shot before using a model trained on faces matching the following emotions: angry, disgust, fear, happy, neutral, sad, and surprise. Once the classifier has been trained, we assume that it can find facial traits unique to each emotion. Thus, when sending an image to the classifier, it would give a level of prediction for each emotion, giving its estimation of the face with his knowledge of each emotion. The label from the enumeration above that had the highest level of prediction is considered to be the emotion shown by the user.

If the emotion detected from the enumeration doesn't correspond to one that we're interested in, we simply consider it as a wrong choice of label. On the other hand, if the emotion with the highest level of prediction matches the one that the user was asked to do, then we consider that the user successfully passed this requirement.

The trained model was provided by Juan Camilo López Montes [15] and the emotion detection had accuracy of around 70% on a given data set.

Eye Blinks The third and final detector relies on detecting the eye blinks of the user from the live feed. Differently, the implementation for the blinks isn't working on a single shot but more in a succession of shots, considering a blink as a succession of consecutive frames under a threshold and a succession of frames above. Once again, we instantiate a predictor pre-trained on a dataset focused on placing landmarks on the faces of users especially to detect the eyes. The goal is to find the coordinates corresponding to the eyes of the users from the picture. Then, having determined the coordinates for the extremities of the eyes using our model, we can convert the landmarks to an array of six (x,y) coordinates used to circle the eye in clockwise order. Studying the distance between these six points is very important to determine whether the user has blinked or not.

Using the work of Tereza Soukupová and Jan Cech [16], we implemented a solution using the relation called the Eye-Aspect-Ratio or EAR which computes the ratio between the distance of the vertical eye landmarks divided by the distance between the horizontal landmarks.

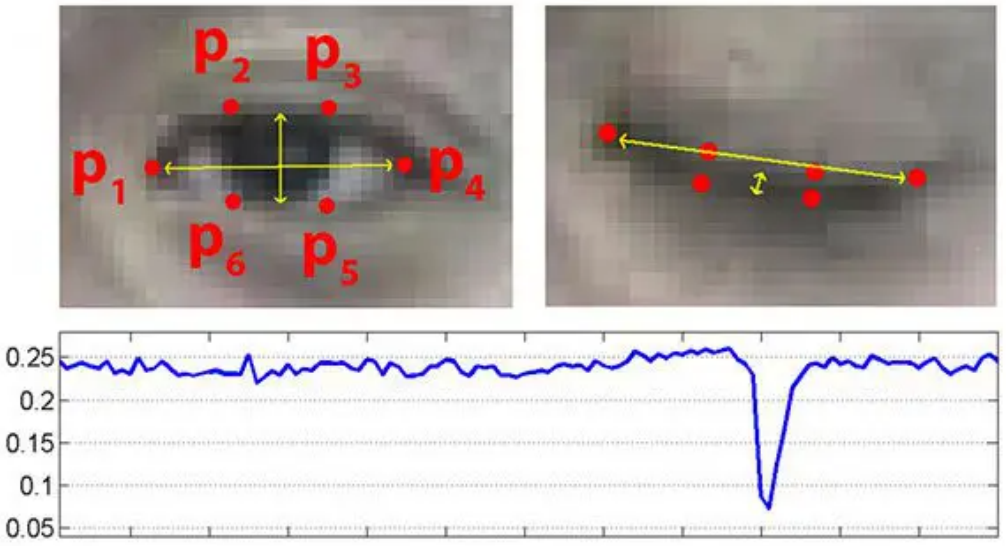


Figure 9: Top: Visualization of the eye landmarks when the eye is open and when it is closed Bottom: Plot of the eye ratio, the dip indicates a blink [16]

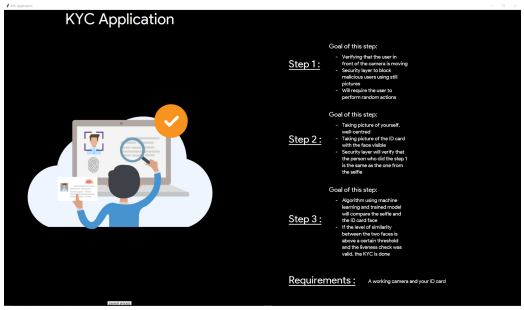
When the ratio drops suddenly to almost reach zero, this is symptomatic of a blink. Indeed, this is visible in the definition of the EAR as the numerator will go to zero, the distance between the vertical points going to 0, with the two eyelids joining themselves. The horizontal distance remains the same, therefore the EAR goes to 0. Furthermore, in case the eyes remain open, the EAR stays consistent, making the process of detecting eye blinks very convenient.

We implemented this protocol, by computing the eye aspect ratio of the two eyes and averaging the values of the two eyes to verify that the user isn't simply closing one eye, and if the EAR goes below a certain threshold, we consider that the eyes were closed. As we ask the user to blink the eyes, we want to ensure that the eye blinking did last a certain number of consecutive frames to eliminate possible camera glitches.

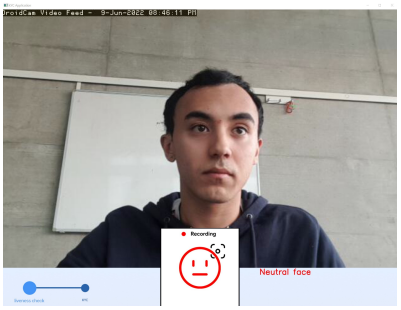
GUI Implementing a user-friendly GUI was an important element we took into consideration during the development of this secure KYC. First, we decided to implement a home page where

we would explain the entire process of our user identification application, to make the process more understandable for all the users.

Furthermore, to pursue our quest to make the process of doing the KYC easy and practical for everyone, we added symbols in the interface along with the text to make the liveness check understandable for everybody and playful. We also added a change of color from red to green to show the user that he performed the task correctly.

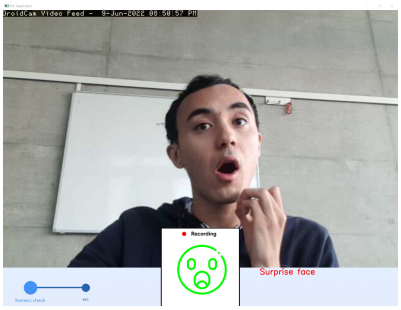


(f) Homepage

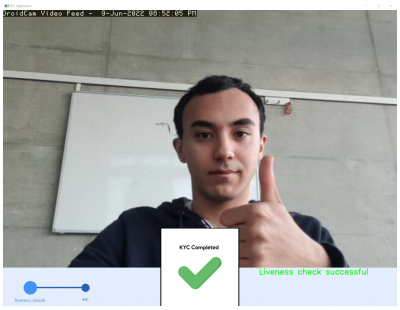


(g) Normal Face liveness

Figure 10: GUI samples of the desktop application



(h) Surprise face valid



(i) Liveness check completed

Figure 11: GUI samples of the desktop application

7.2.4 Limit testing the liveness check

The final part of the project consisted of finding the possible limitations of the liveness check by testing it in extreme conditions. We also tried possible attacking methods to see whether the implementation matched our theoretical hopes. We tried testing the liveness check in different lighting environments, different angles, and face positions. Moreover, we tried hiding parts of the face to see if it changed something to the results. Finally, we placed ourselves in the role of an attacker and we tried using techniques like using pictures of a person doing the poses to see whether that worked. The results of the testing are presented in the table below.

Experiment	Eye Blinking	Angry	Smile	Turn face right	Turn face left	Surprise
Normal conditions	y	y	yyy	yyy	yyy	yyy
Low lighting	∼	yyy	∼	∼	∼	y
High lighting	n	n	n	n	n	n
Very close to the camera	∼	∼	yyy	n	n	yyy
Very far from the camera	yyy	n	yyy	yyy	yyy	yyy
Only the eyes	n	n	n	n	n	n
Without the eyes	n	n	n	∼	∼	n
Printed pictures of poses	∼	y	yyy	yyy	yyy	yyy
Pictures on a screen of poses	∼	∼	y	y	y	y
Only the mouth	n	n	n	n	n	n
With glasses	y	y	yyy	yyy	yyy	yyy
Without the mouth and nose (mask)	y	n	n	∼	∼	n
Nothing	n	n	n	n	n	n

Table 3: Results of the experiments made on the liveness check (n: failed, : failed most of the time, y: passed most of the time, yyy: passed everytime)

These results show that the implementation seems to work as intended in normal conditions and face some issues in a few extreme conditions. For each of these experiments, we ran from five to ten times to study the reactions of the code to these situations. What we observed, is that most of the time, the implementation requires the entire face to appear within the shot to be working. This can be explained by the fact that face comparison uses facial landmarks on specific positions of the face and the absence of some may cause critical issues in this step. Furthermore, we also remarked that the liveness check worked fine in the negative face, where it didn't detect anything when the camera wasn't showing any face which was expected. Another issue that the testing showed is the problem of high exposure for the camera as in the case where the user is facing a strong source of light, the liveness check is failing. This can be explained by the face detection using the Haar-cascade detection, which uses the difference in lighting to detect facial patterns. In the case of high exposure, those differences in lighting become smaller and thus harder to detect. Using another method of face detection could be a solution to solve this issue.

Finally, the tests around adversarial thinking showed concrete limitations of the code. Printing pictures of a user doing the different poses and showing them to the camera actually passes our implementation for the liveness check extremely well. Same for showing the same faces on a screen. However, for the second case, the face matching seem to not work as smoothly and most of the time, it considered it as a different person, not passing the KYC. However, printing the images worked and shows that the solution isn't perfect and requires further reflections. We still have to remind that this solution makes the attack way more complex, requiring the user to have access to multiple shots of the same person doing all the poses.

8 Further Work and Conclusion

Further Work Even if the goal of the project has been achieved, some limitations remain and some question marks are still unanswered. These limitations mostly come from time and resources issues, since for example implementing a verification of the format of the official document was an idea that we explored. Yet, we abandoned this idea as we required a large data set of official documents and a lot of work for possibly no effective results. Implementing a verification on the identity document would however make the process way more secure as we would have layers of security on the two different sets of inputs.

Limitations aside, there were a lot of other ideas that could be explored to secure the user identification, one could supplant a layer of security that would verify that the picture of the official document is a direct picture of the document. The goal would be to eliminate attacks from users not having the actual document but simply a copy and could thus impersonate someone. The development of solutions to detect deep fakes could also help seriously to make the entire

market more secure and would probably convince new platforms to implement such policies. Finally, making the application lighter so that more devices could benefit from this solution like mobile devices would offer a great way to push for an industry more aware of the advantages that secure KYCs may offer.

Conclusion In this project, we began by exploring the world of NFTs and their components as a way to introduce the notion of NFT marketplaces and study their challenges. In less than ten years, NFTs became a massive trend, and all of its actors needed to adapt analogously, forcing extremely rapid progress. Similarly, we had limited time to implement these security principles for my project which forced us to find a solution that would solve the issues in a relatively short amount of time. Thinking about ways to implement a secure user identification process was very challenging, yet a very interesting experience. Moreover, since most of the current KYC implementations require human intervention or are extremely slow and not user-friendly, working on a new solution that could be adapted for all kinds of applications made the project much more exciting. The use of liveness check to perform a KYC could be easily transplanted in other domains than NFT platforms such as traditional banking systems, transportation services...

References

[1] Qin Wang, Rujia Li, Qi Wang, Shiping Chen. Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges, 2021
https://arxiv.org/pdf/2105.07447.pdf

[2] Encyclopedia.com. Token. In Encyclopedia.com dictionary, 2018
https://www.encyclopedia.com/science-and-technology/computers-and-electrical-engineering/computers-and-computing/token

[3] Julien Moretto, CoinHouse What is a token?, 2019
https://www.coinhouse.com/learn/blockchain-technology/what-is-a-token/

[4] Rakesh Sharma. Non-Fungible Token (NFT) Definition, 2022
https://www.investopedia.com/non-fungible-tokens-nft-5115211

[5] Entriken, W., Shirley, D., Evans, J., and Sachs, N., EIP-721: ERC-721 Non-Fungible Token Standard, 2018
https://eips.ethereum.org/EIPS/eip-721

[6] ISO/IEC JTC 1/SC29/WG1 N100158, REQ "Use Cases and Requirements for JPEG NFT v1.0", 2022
https://jpeg.org/jpegnft/documentation.html

[7] The Digital Fifth, Demystifying Different Types of KYC, 2021
https://thedigitalfifth.com/demystifying-different-types-of-kyc-for-financial-institutions/

[8] David McNeal, Crypto 101: The 4 Layers of Blockchain Protocol, 2021
https://thecryptowriter.co/crypto-101-the-4-layers-of-blockchain-protocol-8c2256c3557b

[9] @OpenSea. Tweet. "However, we've recently seen misuse of this feature increase exponentially. Over 80% of the items created with this tool were plagiarized works, fake collections, and spam", 2022
OpenSea's Tweet

[10] Paul Viola, Michael J. Jones, Robust Real-Time Face Detection, 2001
https://link.springer.com/content/pdf/10.1023/B:VISI.0000013087.49260.fb.pdf

[11] OpenCV documentation, Cascade Classifier, 2022
OpenCV documentation

[12] Adam Geitgey, Medium, Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning
https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

[13] James Vincent, The Verge, Liveness tests used by banks to verify ID are 'extremely vulnerable' to deepfake attacks, 2022
https://www.theverge.com/2022/5/18/23092964/deepfake-attack-facial-recognition-liveness-test-banks-sensity-report

[14] Bradley, D.: Profile face detection, 2003
http://www.davidbradley.info/publications/bradley-iurac-03.swf

[15] juan-csv, emotion_detection on GitHub, 2020
https://github.com/juan-csv/emotion_detection

[16] Tereza Soukupová, Jan Cech, Eye-Blink Detection Using Facial Landmarks, 2016 ftp://cmp.felk.cvut.cz/pub/cmp/articles/cech/Soukupova-TR-2016-05.pdf

[17] Lina Shang, Cui Zhang, Haozheng Wu, Eye Focus Detection Based on OpenCV, 2019 https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=9010199

A GitHub Repository

Link to the repository

B Figures

All the figures without any reference to the bibliography were realized by myself.