Web3-Streaming: a decentralized audio and video streaming platform implementing Conditions-Based Decryption (CBD)

Tesi di Laurea Magistrale in
Computer Science and Engineering -
Ingegneria Informatica

Author: **Andrea Razza**

Student ID: 968483
Advisor: Prof. Francesco Bruschi
Co-advisors: Manuel Tumiati
Academic Year: 2022-2023

Abstract

In recent years, technological innovations gradually invaded everyone's life, suffice it to think the quantity of tasks that are performed everyday thanks to a smartphone and mobile applications. In addition to the positive aspects brought by this evolution it can be found a big issue arising from it: the data privacy and security one. This problem comes from the fact that people's sensitive data became precious for cybercriminals who attack companies causing sensitive data leaks for economic purposes.

This project idea was born from a collaboration with the company KNOBS, with whom it was thought about how to solve the problem of data privacy for the development of a decentralized application that offers on-demand streaming services of audio and video files. The work that anticipated the platform developing consisted of studying different cryptographic technologies and protocols in order to find the best solution for the problem addressed. The choice fell on *Conditions-Based Decryption (CBD)*, a technology that leverages the decentralized *Threshold network* and allows, through a strategy deploy on the blockchain, its member nodes to use mutual cooperation to encrypt and decrypt data. During the encryption phase some settings are specified by the data owner, the most important are the threshold, that is the minimum number of *Threshold network* nodes whose cooperation is required for the file decryption, and the decryption condition set that indicates the requirements to be fulfilled by the data requester's *MetaMask wallet* to gain access to the file.

The platform is built to offer different services and interfaces to data owners and data requesters: the data owners, through the encryption page, can create new strategies or use previously deployed ones to encrypt and save new files, while the data requesters, through the decryption page, can visualize the encrypted files previews with the related decryption conditions and try to decrypt them. CBD technology was chosen since it offers a good data security level while allowing the data owner to potentially share his files with many users with only a single strategy deployment.

Keywords: blockchain, streaming app, cryptography, decryption conditions

Abstract in lingua italiana

Negli ultimi anni le innovazioni tecnologiche hanno gradualmente invaso la vita di tutti, basti pensare alla quantità di attività svolte ogni giorno grazie ad uno smartphone. Oltre agli aspetti positivi portati da questa evoluzione si riscontra un grande problema che ne deriva: quello della privacy e della sicurezza dei dati. Questo problema deriva dal fatto che i dati personali sono diventati preziosi per i cyber-criminali che attaccano le aziende causando fughe di dati per motivi economici.

L'idea di questo progetto nasce da una collaborazione con l'azienda KNOBS, con la quale si è pensato a come risolvere questo problema per lo sviluppo di un'applicazione decentralizzata che offrisse servizi on-demand di streaming di file audio e video. Il lavoro preliminare è consistito nello studio di diverse tecnologie crittografiche e protocolli al fine di trovare la miglior soluzione per il problema affrontato. La scelta è ricaduta su *Conditions-Based Decryption (CBD)*, una tecnologia che sfrutta la rete decentralizzata di *Threshold* che consente, tramite il deployment di una strategia sulla blockchain, di usare i suoi nodi per criptare e decriptare dati grazie alla loro cooperazione. Durante la fase di cifratura alcune impostazioni sono specificate dal proprietario dei dati, le più importanti sono il threshold: numero minimo di nodi della rete *Threshold* la cui cooperazione è richiesta per la decifratura del file, e il set di condizioni di decifratura: requisiti che il *wallet MetaMask* dell'utente deve soddisfare per avere accesso al file.

La piattaforma è costruita per offrire servizi e interfacce differenti ai proprietari e ai richiedenti dei dati: il proprietario, tramite la pagina di cifratura, può creare nuove strategie o usare quelle già create per criptare e salvare nuovi file, mentre il richiedente, tramite la pagina di decifratura, può visualizzare l'anteprima dei file criptati con le relative condizioni di decifratura e provare a decifrarli. La tecnologia CBD è stata scelta in quanto offre un buon livello di sicurezza consentendo al contempo al proprietario dei dati di condividere potenzialmente i propri file con molti utenti effettuando il deployment di una singola strategia.

Parole chiave: blockchain, app di streaming, crittografia, condizioni di decifratura

Contents

1 | Introduction

Privacy issues related to sensitive data is a very serious concern in nowadays world where technologies resources keep on increasing their fundamental role in everyone's lives. Persona data, such as Personally identifiable information (PII), Biometric data, Protected health information (PHI), but also positions, personal interests and many others are very important information about all of us that are collected everyday and given to companies which stores them for various purposes. These data are really precious and if they gets into the wrong hands user's privacy could be violated and this can result to very dangerous consequences for the individuals. [18]

Everyday all of us accept privacy-usability trade-offs in order to exploit all the services that the technology offers without thinking that by simply booking a taxi transfer location data, information about financial details and many other data are communicated. Cybercriminals have stolen in these years millions of credentials and sensitive data to sell them or to exploit them for various purposes, such as identity theft and financial frauds.[14] These cybercriminals are very powerful computer engineers that uses their knowledge to commit malicious activities on information systems or networks to steal companies or private sensitive data in order to generate profit. They attack these kind of systems in various ways, the main and most frequent are:

- Unauthorized access: this can happen through hacking, weak passwords, social engineering, or inadequate access controls.
- Malware and ransomware: malicious software, such as Trojans, viruses, worms, can infect information systems for multiple purposes, including senstitive data theft.
- Phishing and social engineering: the first one consists of impersonating legitimate entities and tricking individuals into revealing personal information through emails, websites and many others. Social engineering tricks instead exploit human psychology to manipulate individuals into disclosing confidential information.
- Lack of encryption: the data encryption process makes the stored or sent sensitive data inaccessible to unauthorized individuals, weak or nonexistent encryption can

lead attackers to easily access these data.

- Data retention and privacy policies: often organizations collect and retain more data than the strictly necessary, leading to increased privacy risks. In these cases, data breaches can be caused by inadequate privacy policies or improper handling of personal information.
- Internet of Things (IoT) vulnerabilities: the increasing presence of smart interconnected devices in every area introduces security risks because these devices can be compromised and data breaches danger is very high.

Another important aspect of this issue is the artificial intelligence and machine learning techniques proliferation: some companies to which we legally yield our sensitive data exploit these technologies to train algorithms that everyday learns new things about all of us and have become capable of influencing human opinions and ideas.[14]

The Web3 platforms based on blockchain, that is the main argument of this thesis, aim to decentralize and provide a more secure control to users over their data and transactions, but also these kind of applications present privacy and security issues, among which we can find:

- Pseudonymity and privacy: while blockchain offers pseudonimity by using cryptographic keys instead of real-world identities, it is still possible to trace transactions and associate them to a specific address.
- Smart contract vulnerabilities: smart contracts are self-executing agreement on the blockchain and they can contain coding errors and vulnerabilities that can lead to cyber attacks. By attacks it can be meant financial losses, unauthorized access and many other dangerous interventions for personal privacy.
- Data privacy: since the blockchain is a transparent entity by nature, that ensures immutability and public transaction visibility, storing personally identifiable and sensitive information directly on it can be translated in unlimited access to them from anyone on the network
- Off-chain data storage and oracles: Web3 applications may rely on off-chain or oracle storage services to save and access data. These systems can introduce security issues if they're not properly secured or if the data stored has been tampered by external entities
- Malicious nodes: it can happen that a spiteful individual joins the network and tries to damage it. This kind of attack is performed by flooding the network with

transactions or by trying to reverse valid ones. [24]

Addressing these security problems that, despite introducing more restrictive measures, the blockchain still present, requires a combination of technical solutions, such as strong data encryption with authentication techniques, secure smart contract coding practices and users education about risks, otherwise it will be always faced the privacy/usability trade-off in the technology tools everyday exploitation.

Threshold Network aims to solve this compromise by using cryptography to unlock the potential of digital assets without intervention or trust in a centralized authority. The Threshold idea consists of distributing the data decryption operations on many distributed and independent entities represented by the Threshold Network nodes. In order to succeed in this decryption operation it is necessary to reach a minimum number of entities cooperating with each other.[14] This technique is the basis of the so called Threshold cryptosystem that aims to address the issue of single point of failure in traditional cryptographic systems because even if some of the threshold agents are compromised, the system can still maintain its security as long as the threshold number of agents remain uncompromised. Threshold Access Control enables end-to-end encrypted data sharing and communication aiming to trust-minimization via cryptography and decentralization and it offers two protocols: Proxy Re-Encryption (PRE) and Conditions-Based Decryption (CBD). The idea behind the first one is to offer an end-to-end cryptography protocol that allows proxy entities, the nodes on the Threshold Network also known as Ursulas, to re-encrypt the encrypted data to transform the decryption key from one to another without ever having access to the plaintext. The PRE protocol on-chain policy is per-Bob: each encrypted message has one and only recipient, specified during the policy creation phase, that is able to decrypt it. The idea from which the second protocol was born, instead, consists of giving access to the encrypted message only to users that satisfies the decryption conditions related to the message. In this scenario, the Threshold Networks nodes job is to check whether the data requester's Metamask account satisfies the required Conditions and, if so, provide the key to decrypt the message. The message encrypter can build a condition set from three different condition types: the first one is the so called *timelock based condition*, which make the message decryptable if and only if the specified time condition is satisfied. The second one is the *EVM based condition* that is a condition type regarding blockchain token possession of the data requester. And the last condition type that the encrypter can specify is the *RPC based condition* that is a money condition, here the message is decryptable if the data requester's Metamask account satisfies the money condition specified by the encrypter.

The topic of study of which this thesis concerns consists of the developing of a Web3 audio and video streaming platform built on blockchain that take advantage of the underlined Conditions-Based Decryption service; moreover, for the faced use case it was carried out also the analysis of the strengths and weaknesses of the two underlined protocols in order to evaluate the best one to use from different points of view. During the platform development it was also created a ERC721 Non-Fungible-Token (NFT) called IrpiniaNFT in order to relate messages decryption to this NFT possession.

The aim of this platform is to provide to users a streaming service that, through this cryptographic protocol, has a very high level of data privacy but at the same time is as much as scalable and usable as possible in order to reach any user without any security issue. In this way the usability-security trade-off which was mentioned above is completely erased.

1.1. Thesis outline

The thesis is organized as follows:

- Chapter 2 provides an overview of the used technologies: an introduction of the blockchain technology with all the exploited decentralized tools and services. Subsequently a deep and careful analysis of the cryptographic technologies offered by Threshold Network will be done.
- Chapter 3 presents the developed decentralized streaming platform, every nuance of the software is properly analyzed and explained in order to fully understand the application workflow.
- Chapter 4 describes the testing techniques implemented both during the development process and at the end of it to validate and verify the platform functionality and evaluate the reached performances.
- Chapter 5 consists of the thesis conclusion which contains final considerations about the platform and an analysis of prospects for improvements and future developments.

2 | Literature

In order to fully understand the developing process of the web3 streaming platform it is necessary to analyze and deepen the tools and the technologies used. It will be firstly presented an overview about the blockchain and all the related services that are based on this technology, exploited to build the platform. It will be explained and clarified what is a decentralized application (*dApp*), a wallet and also some decentralized tools such as *IPFS* and the increasingly famous *NFTs*, in short all the platform developing tools that rely on these kind of networks: the decentralized ones. Subsequently they will be presented and compared, from the theoretical point of view, the cryptographic technologies, offered by Threshold Network, evaluated for the addressed use case: *Proxy Re-Encryption (PRE)* and *Conditions-Based Decryption (CBD)* that are apparently similar each other but hide some deep differences that pushed to choose the second one in order to manage the platform cryptographic operations.

2.1. Blockchain

In these last years it is increasingly common to hear about blockchain and cryptocurrencies, but too often there is confusion about what they really are. The aim from where the blockchain idea was born was to create a network without a central entity, made up of multiple components that own the same data in order to ensure integrity of the data stored in the structures.[16] Blockchain, as we intend it nowadays, is a decentralized and distributed digital ledger technology that enables secure and transparent recording of transaction across multiple computers or nodes. Its *"data structure can be defined by a list of records, also called blocks, that can contain data from any type and that are linked by means of cryptographic references."*[16] The key features of blockchain are:

- Decentralization: instead of relying on a central authority, as traditional information systems do, blockchain relies on a network of nodes whose tasks are to validate and record transactions. This feature is useful to make the system more reliable and attack-resistant.

- Transparency: the blockchain's ledger is visible to any participant in order to guarantee the transparency and enable anyone to check for the transactions integrity.
- Security: blockchain takes advantage of cryptographic tools to secure transactions and keep the data integrity: when a block is added to the network, it is extremely difficult to modify its content.
- Immutability: the blockchain's name comes from the fact that each block is connected, by containing a reference, to the previous one thus forming a chain. This mechanism provides an immutable and verifiable transactions record and ensures the immutability feature because once a new block is added to the chain it can't be easily modified.
- Smart Contracts: they're self-executing contracts that contain terms of agreement within their codes. Their main task is to automatically execute code blocks when certain conditions are met. In this way they automate and provide programmability to blockchain systems.

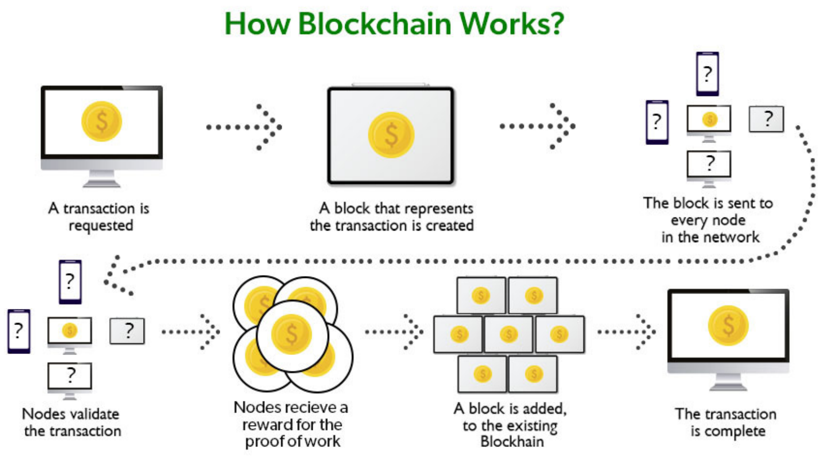

How Blockchain Works?
A transaction is requested
A block that represents the transaction is created
The block is sent to every node in the network
Nodes validate the transaction
Nodes recieve a reward for the proof of work
A block is added, to the existing Blockchain
The transaction is complete

Figure 2.1: Blockchain flow. Source: Geeksforgeeks

2.1.1. The data structure

The blockchain's data structure is based on a decentralized and distributed ledger model that is made up of several nodes connected each other in order to build a chain. In this subsection it will be deeply analyzed the blocks structure. It is necessary to underline that the block structure and the relative components that it is made up of may have variation basing on the analyzed blockchain implementation. However, the essential properties that unite each block variations are the following:

- Block header: the block header contains essential metadata about the block such as its unique identifier (hash), the previous block's one and other technical information like timestamp. The hash represents a signature of the data contained in the block, calculated by applying a cryptographic hash function to the contents of the block, and it is used as digital fingerprint in order to help check the blocks' data integrity.[16]
- Transactions and data: this block's part contains a collection of transactions or data entries. They can represent any kind of data, such as information needed to be stored on the blockchain for applications use cases or movements of digital assets between users in cryptocurrency contexts.
- Difficulty: the level of mining difficulty (Proof of Work) regarding the block itself.[16]
- Nonce: the nonce is a random number useful during the mining process and it is used to modify the generated hash.
- Merkle Tree Root: in several blockchain implementations, the transaction and data of a block are recorded using this kind of data structre. A Merkle tree is a hierarchical layout of hashes constructed by gradually combining individual transactions into pairs until a single root hash, that is called Merkle root, is obtained.

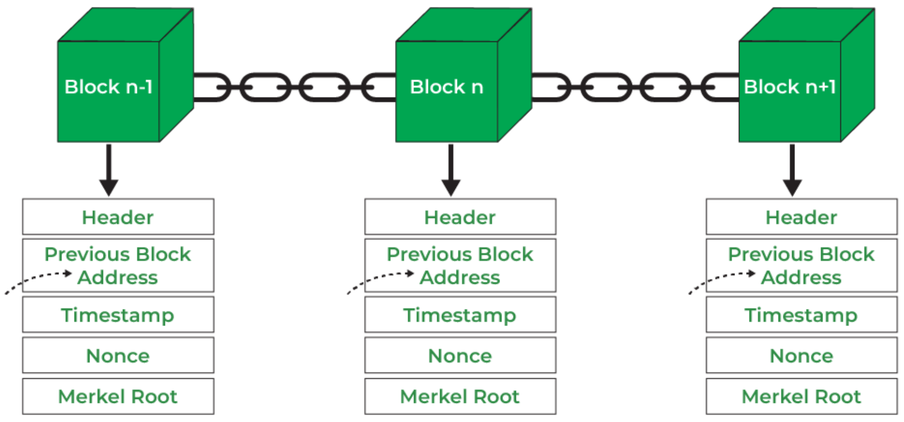


Figure 2.2: Blocks structure. Source: Geeksforgeeks

2.1.2. The network

The blockchain decentralized network is made up of nodes that collaborate with each other in order to guarantee its availability and integrity. Blockchain typically utilize a *Peer-To-Peer (P2P)* network architecture in which the nodes that compose it communicate directly with each other, thus avoiding the presence of a central entity that manages the network. All the actors of this P2P network, the nodes, have the same status and can act both as clients and servers, this design choice allows them to interact, share information and also to implement a mechanism of propagation of transactions and blocks to all the nodes in order to guarantee the blockchain's state synchronization. Thanks to this mechanism each node keep an always updated copy of the entire ledger, ensuring redundancy and increasing the network's failure tolerance, because in case a node fails and goes offline, all the others can remedy and keep the blockchain working without any issue.

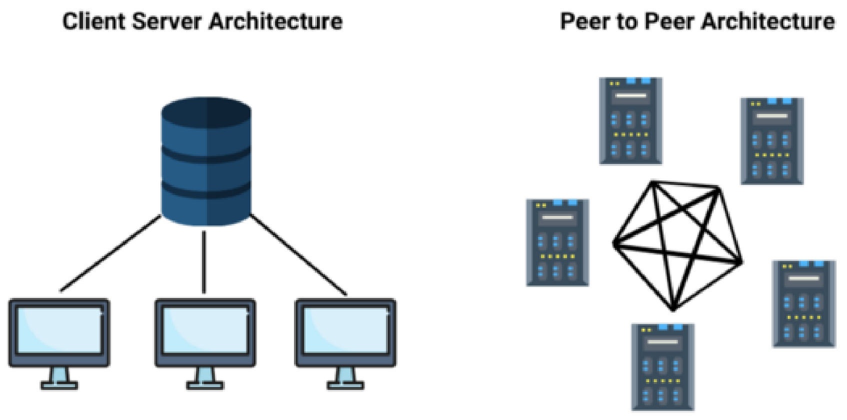


Figure 2.3: Traditional Client-Server System vs. Peer-To-Peer Network. Source: Bybit Learn

Another very important aspect regarding the blockchain network is the *Consensus Mechanism*: a group of protocols, algorithms, incentives and ideas that the blockchain network takes advantage of in order to achieve agreement on its state: transaction validity and blocks order.[10] This mechanisms determine and decide how network nodes agree on the blockchain's state and prevent malicious actors from tampering it thanks to the need to have the majority of nodes in the network reach a consensus on the blockchain state agreement. The most common consensus mechanisms are *Proof-Of-Work (PoW)* based one, used by *Bitcoin*, and *Proof-Of-Stake (PoS)* based one, used nowadays by *Ethereum*.

Blockchain can be moreover categorized in two main groups basing on the data secrecy that run on it: *permissionless* and *permissioned* blockchains. The first one is also known as public blockchain because it is available and accessible to every user that wants to use it in order to send and validate transactions or data. It is trustable, decentralized and has an anonymous nature but, since it has huge size, the transaction process speed is very slow and Proof-Of-Work is very high energy-consuming so good hardwares are required to join this kind of network.[13] The second type is the *permissioned* blockchain: they are closed networks reserved to a group of users that are allowed to validate transactions and data, these networks are not truly decentralized as permission operations are required. To permissioned blockchain belong two blockchain types: *private* and *consortium* blockchain. The first ones are controlled by one authority, run in closed networks and only few people are allowed to participate, from these properties it derives that they have very high speed and privacy levels.[13] The second type, the consortium blockchain is controlled by a group, indeed it is also known as *Federated Blockchain*, the main characteristic of this blockchain type is that some part are public and some others are private, in this way, more than one organization can manage the blockchain without any privacy or security issue.

The last blockchain type is the *Hybrid blockchain*: it is a mixed version of private and public ones where some part are under some organization management and some others are visible like the public blockchain. It brings the best aspects from both the starting versions.[13]

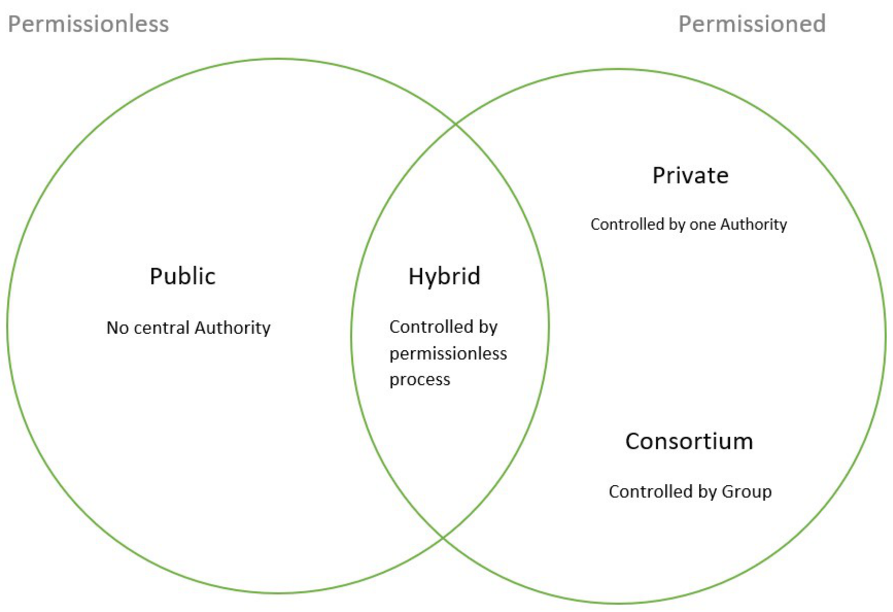


Figure 2.4: Types of blockchain. Source: Geeksforgeeks

2.1.3. Decentralized Applications

Decentralized applications, also known as *DApps*, are software applications that run on a decentralized network, in case of this thesis work on the blockchain, instead of on a single authority like traditional applications that rely on a central server for data storage and processing.[19] DApps leverage the decentralized nature of blockchain to offer to the app users several benefits like transparency and security. DApps run on the blockchain network which offer an environment that is public, open-source and decentralized and, in this way, they won't ever be under a single central entity control and dependency.[19].

A very important role in the majority of DApps is played by *Smart Contracts* that are self-executing contracts that automate processes and allow trustless interactions between participants. Moreover, *Smart contracts* guarantee DApps data integrity, since the data stored within a smart contract is transparent and immutable, as it is written on-chain, and data security, since they also take advantage of cryptographic techniques while storing data and transactions. Another functionality that *smart contracts* offers to DApps is the interoperability, as they can interact and communicate with other smart contracts and DApps allowing exchange of assets and data between different DApps. Among all the shown advantages that decentralized applications introduce we can also find resistance to censorship, as no participant is able to delete or block contents, but also disadvantages can be found such as the fact that their use is still in the early stages and it is prone to some problems and unknowns or the difficult that can be found to make needed code modification to fix some bugs or software updates, as the blockchain data is immutable by definition.[19]

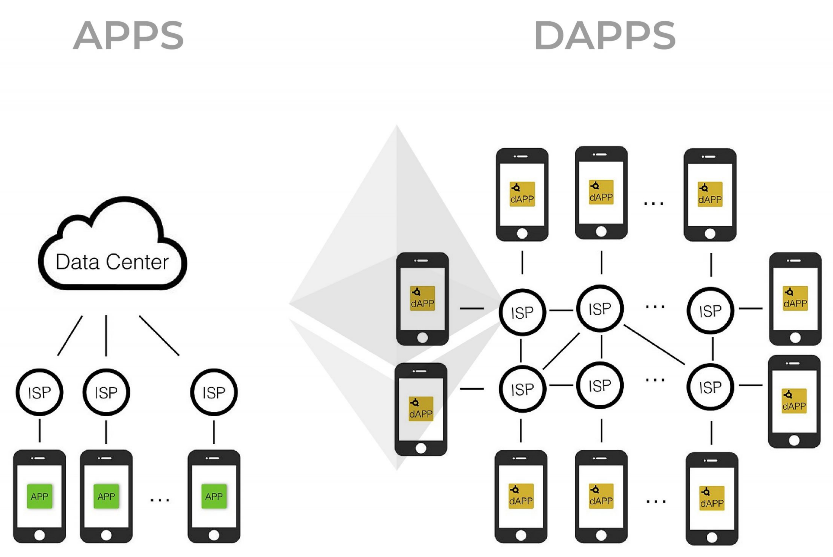


Figure 2.5: Traditional Apps vs. Decentralized Apps. Source: Linkedin

2.1.4. Wallet

In order to interact with the developed decentralize streaming application and with the blockchain in general each user need to own a blockchain wallet that is a user interface that allows users to manage their cryptocurrencies: they allows to transfer funds between users through writing transactions to the blockchain.[2] Crypto wallets also manage cryptography thanks to a public-key cryptography which they're based on, the wallet's key features are a public/private key couple and the address of the wallet. The private key is a 256-bit binary number, generally represented as a 64 length alphanumeric string, used to offer the access to blockchain network to the owner of the relative wallet. The public key is instead used to encrypt information sent on-chain by the owner. Lastly, the address is an alphanumeric string derived from the public key that specifies the crypto wallet location on the blockchain.[2]

In order to interact with the developed web3 audio and video streaming platform the popualar browser extension that act as cryptocurrencies wallet *MetaMask* is needed. The wallet enables the users that wants to upload and encrypt files to deploy the smart contract by paying the required fee on the *Mumbai Testnet* and enables the encrypted data requesters to try to access the data by provide their wallet information to nodes in order to check the decryption conditions fulfillment.

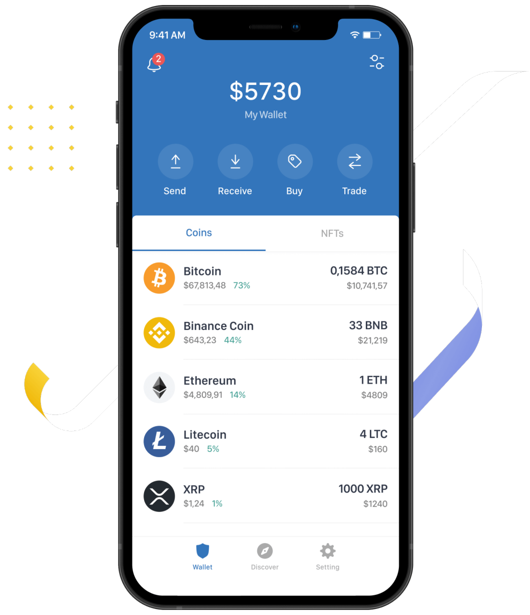

9:41 AM
$5730
My Wallet
Send
Receive
Buy
Trade
Coins
NFTs
Bitcoin $67,813,48 73% 0,1584 BTC $10,741,57
Binance Coin $643,23 44% 33 BNB $21,219
Ethereum $4,809,91 14% 1 ETH $4809
Litecoin $40 5% 4 LTC $160
XRP $1,24 1% 1000 XRP $1240
Wallet
Discover
Setting

Figure 2.6: Crypto wallet illustration. Source: Trustwallet

2.1.5. InterPlanetary File System (IPFS)

When developing a distributed application (DApp) it obviously necessary to adopt a strategy to store and retrieve data, *InterPlanetary File System*, most commonly known as *IPFS*, is a Peer-To-Peer hypermedia protocol designed to offer to DApps a decentralized solution for this task. IPFS allow users to store and access any kind of data, like files but also softwares identified and addressed using *content-based addressing*, indeed each file is assigned a unique cryptographic hash, computed from its content, that is used as its identifier. The contents hosted by IPFS can be of a large variety of types including, among the others, databases, documents, files, websites and each uploaded content is retrivable by simply searching for the relative link.[22] When a content is uploaded on IPFS network, it is chopped up into *blocks* and distributed across several nodes that the network is made up of. Similarly to how blockchain networks employs the nodes to verify the transactions, IPFS employs its nodes, which are hundreds of thousand, to store the data exploiting their storage bandwidth. Basically, IPFS offers a service that also centralized platforms offers, with the very important difference that it does not require any centralized data storage and this feature pushes dApps to exploit this service.[22] For the decentralized platform developing *Pinata* and *ThirdWeb Storage* were used, which are services built on top of InterPlanetary File System, in order to take advantage of their decentralization feature to store the encrypted files, the relative metadata and encryption strategy and conditions to be reused. This choice was born from the need to have a fast, reliable, secure and distributed entity to store the encrypted data and all the necessary platform data.

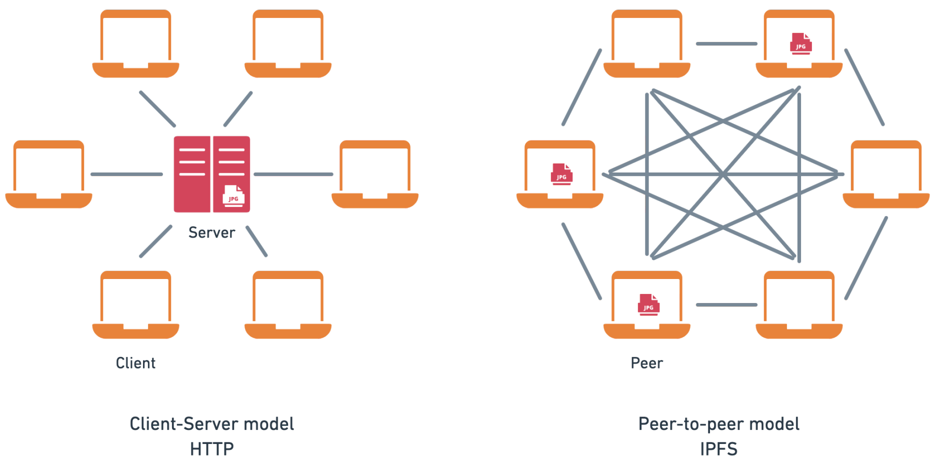


Figure 2.7: IPFS vs. Centralized data storage. Source: Blog.IPFS

2.1.6. Fungible and Non-Fungible Tokens

An important blockchain aspect that is becoming very popular nowadays is the token concept. Tokens are digital assets that run on blockchain, use smart contracts to define the ownership rules and provide a transparent and verifiable transaction history, they can be stored in wallets and they're used in a lot of blockchain use cases basing on the token type: some examples are governance, utility, security, platform tokens and so on.[20] Tokens can be divided in two different groups: *Fungible tokens (FT)* and *Non-Fungible Tokens (NFT)*, the latter have sparked enormous interest and innovation in art, gaming and other worlds because provide monetization opportunity to creators by certificating their ownership.

- Fungible Tokens (FTs): they are interchangeable token, each token is identical to another of the same type. Example of fungible tokens are the cryptocurrencies like Bitcoin (BTC) and Ethereum (ETH) since each crypto unit is interchangeable with another of the same crypto unit. A standard for Fungible tokens is introduced by *ERC-20*, that stands for *Ethereum Request for Comment-20*, which is a technical specification or set of rules that defines the functionality and behavior of Fungible tokens.[20]

- Non-Fungible Tokens (NFTs): differently from FTs, non-fungible tokens are non-interchangeable, unique, indivisible and irreplaceable digital assets.[20] Each NFT is characterized by distinct properties from other tokens of the same type and for this reason they are different from each other and not exchangeable. NFT's uniqueness is achieved by assigning an identifier, called *Token ID*, to each NFT, this feature is responsible for the popularity of this technology since it allows, for example, artists to tokenize and sell their works while retaining control and receivig royalties on subsequent sales. Another field in which NFTs have acquired a great popularity is *Digital Collectibles and Blockchain Gaming* because they can represent various assets and unique items into games, such as characters, avatars and so on. Some of the most popular NFT collections are *Crypto Punks, Doodles, Azuki* and others that are purchasable on different online *marketplaces* like *OpenSea and Binance.*[20] The standard for non-fungible tokens creation is the ERC-721 one, it allows creators to issue these unique assets via Smart Contracts.

While developing the audio and video streaming platform which this thesis is based on, an NFT was created by the process known as *minting*. The created ERC-721 token is called *IrpiniaNFT*, with *contract address: 0x21739b933261efF4bDE8d375796Bd40Eca8713A3 and TokenID: 0*, and it was created to add conditions for files decryption based on the ownership of this token. For the smart contract creation process they were used the *ERC-721 standard and the OpenZeppelin wizard*[1]. The underlined contract deployment and the successive NFT minting were done on a testnet called *Mumbai Testnet* with *Chain ID* = 80001 and in the next chapters it will be shown in detail the decryption process conditioned by its ownership.

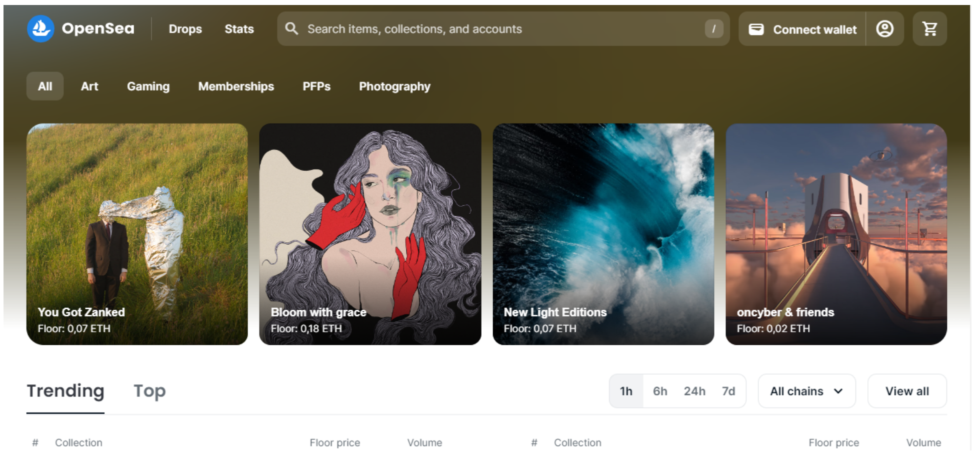


Figure 2.8: A screenshot of the famous NFT marketplace *OpenSea*. Source: OpenSea

2.2. Threshold Network

Threshold Network is a decentralized ecosystem that was born from the merge between two networks and community: *NuCypher* and *Keep*. It offers a suite of threshold cryptography services for web3 decentralized applications that power user sovereignty on the blockchain.[7] The aim of threshold network is to distribute sensitive operations across multiple independent entities represented by the nodes on its network and then return the successful operation result if and only if a minimum number of nodes addressed to operate, known as *threshold*, cooperate with each other in order to compute the response. This process increases security and availability platforms levels and moreover protects against the single point of failure and against malicious nodes presence because a single individual is not able to corrupt the system.[8]

The idea is based on a famous cryptography techinque known as *Threshold Cryptosystem* that is one of the most secure and and reliable one. Threshold Cryptosystem's work consists of encrypting information and distributing secrets among a cluster of independent entities which guarantee *fault-tolerance* thanks to the system's ablity to keep on working also when some of them fails. The general idea that makes this cryptosystem very strong is to consider the participants as susceptible to compromise and, thanks to the threshold concept, even if some of the participant collude the encrypted secret remains safe.[6]

Threshold network, as said, provides to web3 developers an *Access Control* service that is a suite of cryptographic services based on its decentralized network which enables end-to-end, secure and fully decentralized communication and data sharing. The Access Control service is basically divided in two distinct technologies: *Proxy Re-Encryption* (PRE) and *Conditions-Based Decryption* (CBD) both of which provide *end-to-end-encryption-as-a-service*.[12] These technologies are a fundamental topic of this thesis because the developed decentralized streaming platform idea was born from the need to obtain at the end of the work an application that offers sophisticated cryptographic technologies. Both the offered protocols were evaluated and analyzed with respect to the desired final product and, finally, *Conditions-Based Decryption* was exploited for the platform development. In the fist developing phases a web3 streaming *Proof-Of-Concept* implementing *Proxy Re-Encryption* was developed but weaknesses and scalability limits with respect to the goal influenced the choice of using CBD. In this chapter PRE and CBD workflows will be analyzed in detail in order to fully understand the thesis project work.

2.2.1. Proxy Re-Encryption (PRE)

Threshold Proxy Re-Encryption technology is a cryptographic middleware that aims to preserve applications users' privacy, it is an end-to-end encryption protocol that works with a cluster of proxy entities whose task is to receive an encrypted data and perform a re-encryption operation on it in order to transform the decryption key from one to another without revealing the plaintext, these proxy entities are the Threshold Network nodes and they, always basing on the threshold cryptography, cooperatively re-encrypt the received data if and only if the data requester coincides with the one specified by the data owner.[8]

In order to properly analyze and understand how this technology works from the cryptographical point of view, it is necessary to split the cryptographic process in four different steps, each of which is associated to a principal actor:

- Alice: the data owner, the one who wants to share her data through a PRE application.
- Enrico: the encrypter, an entity that performs data encryption on behalf of Alice.
- Bob: the data recipient, the actor designed for the message decryption.
- Ursula: a PRE node on the Threshold Network designed for the re-encryption operation.

Alice

When Alice wants to share some data she firstly specify a label to categorize the encrypted data, from this label an asymmetric key pair is created: the public one is shared to Enrico for the encryption process, while the private one is used with the recipient's (Bob) specified public key to create a re-encryption key. Then Alice must determine n Ursulas from the Threshold Network employed for the re-encryption process and configure the policy conditions, that are *expiration time, shares value and threshold value (m-of-n values)*, in order to deploy the policy to the blockchain and pay the associated policy fees. In the end the re-encryption key is split into 'n' key fragments, called *kFrags* in order to create an *encrypted treasure map* that contains the list of the Ursula nodes with the relative re-encryption keys (*kFrags*) encrypted for each Ursulas in the policy. Alice finally passes to Enrico the asymmetric encryption key in order to allow him to encrypt the message to be subsequently sent to Bob with the *encrypted treasure map.* [3]

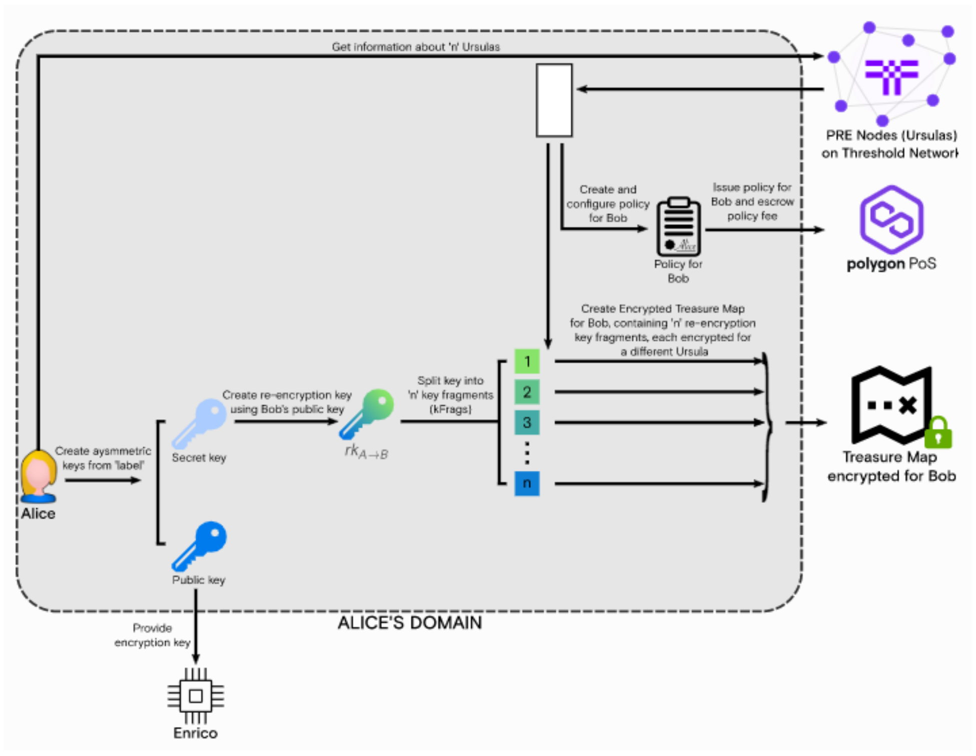

Get information about 'n' Ursulas
PRE Nodes (Ursulas) on Threshold Network
Create and configure policy for Bob
Issue policy for Bob and escrow policy fee
Policy for Bob
polygon PoS
Create Encrypted Treasure Map for Bob, containing 'n' re-encryption key fragments, each encrypted for a different Ursula
Split key into 'n' key fragments (kFrags)
Create re-encryption key using Bob's public key
$rk_{A \to B}$
Secret key
Create asymmetric keys from 'label'
Alice
Public key
Treasure Map encrypted for Bob
ALICE'S DOMAIN
Provide encryption key
Enrico

Figure 2.9: Alice's actions. Source: NuCypher

Enrico

Threshold Network nodes, *Ursulas*, works by following the *Umbral threshold proxy re-encryption scheme*[21] which essentially uses two mechanism: *Key encapsulation mechanism (KEM)* and *Data encapsulation mechanism (DEM)*. Enrico, the encrypter, first of all receives from Alice the data to be encrypted and the asymmetric encryption key generated from the label, subsequently it is created an ephemeral random symmetric key, used to encrypt the data, that is then encrypted with the asymmetric encryption key provided by Alice. In this way the encrypted data (DEM part) and the encrypted symmetric key (KEM part) also called *capsule* are stored together and are sent also with the encrypted treasure map to Bob in order to allow him to receive the necessary tools to retrieve the plaintext. These data can be stored anywhere since they're encrypted and inaccessible from anyone but Bob. [3]

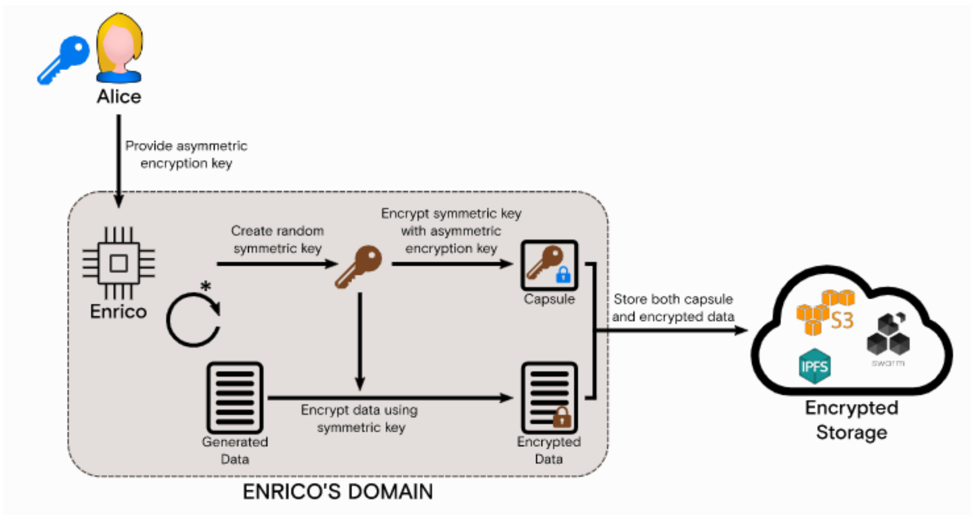

Alice
Provide asymmetric encryption key
Create random symmetric key
Encrypt symmetric key with asymmetric encryption key
Capsule
Enrico
Store both capsule and encrypted data
S3
IPFS
swarm
Encrypted Storage
Encrypt data using symmetric key
Generated Data
Encrypted Data
ENRICO'S DOMAIN

Figure 2.10: Enrico's actions. Source: NuCypher

Bob

In order to obtain the plaintext sent by Alice, Bob must firstly retrieve the encrypted message (that contains the key capsule and the ciphertext) and the encrypted *treasure map* in order to obtain, once decrypted with his private key, the Ursula nodes associated with the policy and the relative re-encryption key fragments encrypted for each of them (*kFrags*). Once obtained this information, Bob sends the capsule and the kFrags to the various proxy entities which returns to Bob the re-encrypted kFrag version, known as *cFrag*. Bob collect these cFrags until he obtains a number of cFrags equal to the *threshold* (m) fixed into the policy; when he reaches this goal, Bob attaches these m received cFrags in order to obtain the symmetric encryption key encrypted under his public key. After the symmetric key decryption process Bob can finally decrypt the message sent by Alice thanks to the result of the previous decryption operation.[3]

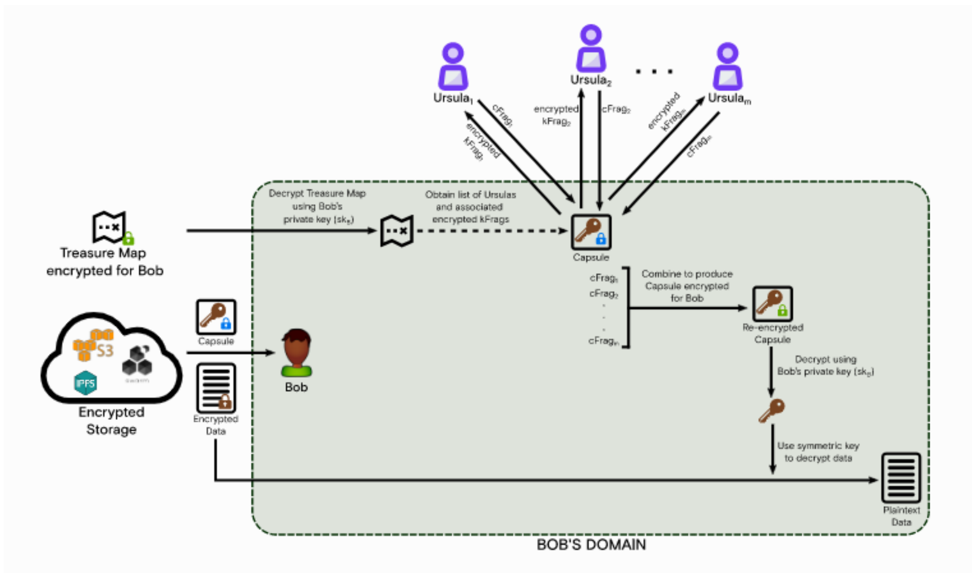

Ursula$_1$
Ursula$_2$
Ursula$_m$
cFrag$_1$
encrypted kFrag$_1$
encrypted kFrag$_2$
cFrag$_2$
encrypted kFrag$_m$
cFrag$_m$
Treasure Map encrypted for Bob
Decrypt Treasure Map using Bob's private key (sk$_B$)
Obtain list of Ursulas and associated encrypted kFrags
Capsule
cFrag$_1$
cFrag$_2$
cFrag$_m$
Combine to produce Capsule encrypted for Bob
Re-encrypted Capsule
Decrypt using Bob's private key (sk$_B$)
Use symmetric key to decrypt data
Plaintext Data
Encrypted Storage
Capsule
Encrypted Data
Bob
BOB'S DOMAIN

Figure 2.11: Bob's actions. Source: NuCypher

Ursula

When Bob receives the key encapsulation and the treasure map, as said, he sends to each Ursula in the trasure map the capsule and the relative encrypted re-encryption key (kFrag). Each Ursula's task is to decrypt the kFrag encrypted under its public key and successively use it to perform the re-encryption operation on the capsule, the related result will be returned to Bob, which requires m of these interactions with m different Ursulas in order to retrieve the fully re-encrypted *capsule*. [3]

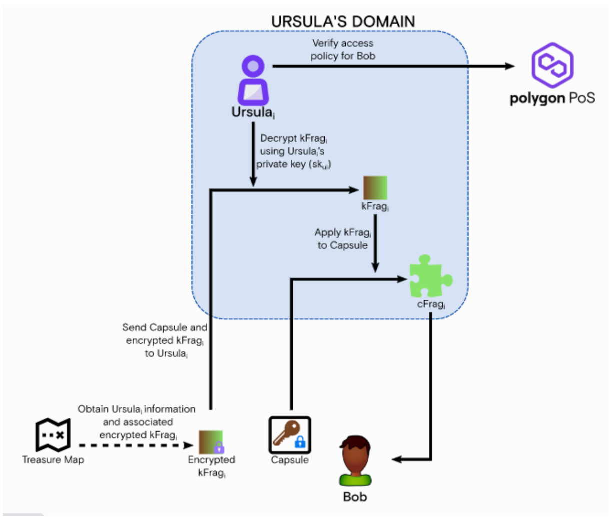

URSULA'S DOMAIN
Verify access policy for Bob
polygon PoS
Ursula$_i$
Decrypt kFrag$_i$ using Ursula$_i$'s private key (sk$_{u_i}$)
kFrag$_i$
Apply kFrag$_i$ to Capsule
cFrag$_i$
Send Capsule and encrypted kFrag$_i$ to Ursula$_i$
Obtain Ursula$_i$ information and associated encrypted kFrag$_i$
Treasure Map
Encrypted kFrag$_i$
Capsule
Bob

Figure 2.12: Ursula's actions. Source: NuCypher

2.2.2. Conditions-Based Decryption (CBD)

For the project development the other analyzed technology, presented by Threshold Network on 6th of January 2023, is *Conditions-Based Decryption*. It is a plug-in service that offers the possibility to share any private or sensitive data within web3 dApps. Identically to PRE service, the encrypted data flowing on the internet remains secret until it reaches an authorized recipient device. This technology enables users to attach decryption condition to the encrypted messages in order to make the messages available in clear only for users that fulfills the specified conditions.[9] Conditions-based decryption adopts the threshold cryptosystem idea too in order to reach an high reliability and security level and remove the single point of failure. The access conditions, that are related to the ciphertext, can be combined in any logical sequence or decision tree and can be of three different types: *time-based* condition type enables the message decryption if and only if the specified temporal condition is satisfied; *EVM-based* condition type regards the recipient's ownership of a token specified by the data owner; *RPC-driven* condition type is based on the recipient ownership of a specified amount of a given token in the wallet.[9] The conditions fulfillment verification is one of the specified *cohort* of Threshold Network nodes jobs ensemble with the key fragments decryption to be sent to the requester that is able to decrypt the encrypted message only when he receives an amount of fragments equal to the threshold specified by the data owner in the strategy deployment phase.[9]

For the developed web3 streaming platform *Conditions-Based Decryption* was the technology used to manage the files encryption and decryption. Nevertheless the above technology explanation, since the technology has been launched only few months ago, it is actually build on top of Proxy Re-Encryption so in the actual state the Ursula nodes that belong to the strategy cohort still perform the re-encryption operation while in the next Threshold Conditions-Based Decryption release the technology is built using a Decentralized Key Generation ceremony, so the Ursulas will not perform any re-encryption, just check the conditions and decrypt the keys fragments.

CBD Encryption process

When a data owner wants to encrypt some data the first step consists of configuring a cohort of nodes from the Threshold Network, with the relative *shares* and *threshold* values, in order to employee them to participate to the data sharing process. When this cohort is chosen, a *Decentralized Key Generation* ceremony is performed in order to create a key pair for the encryption and decryption phase, the derived public key is sent to the encryptor to successively perform the message encryption operation while the private one is broken into segments and sent to each cohort node in order to return each fragment to any conditions-allowed arbitrary recipient during the decryption phase. Contextually, the *strategy*, built from the nodes cohort ensemble with a label specified from the data owner, is deployed onto the blockchain with the relative fee payment. Successively, the data owner must specify a decryption conditions tree and the message he wants to encrypt in order to enable the *encryptor* entity to use these data with the previously deployed strategy for the message encryption. Finally, the data owner can send to any arbitrary recipient the block that contains the encrypted data and the decryption condition and all the information about the cohort of Threshold Nodes which refer to in order to try to decrypt the message if the message specified conditions are fulfilled.

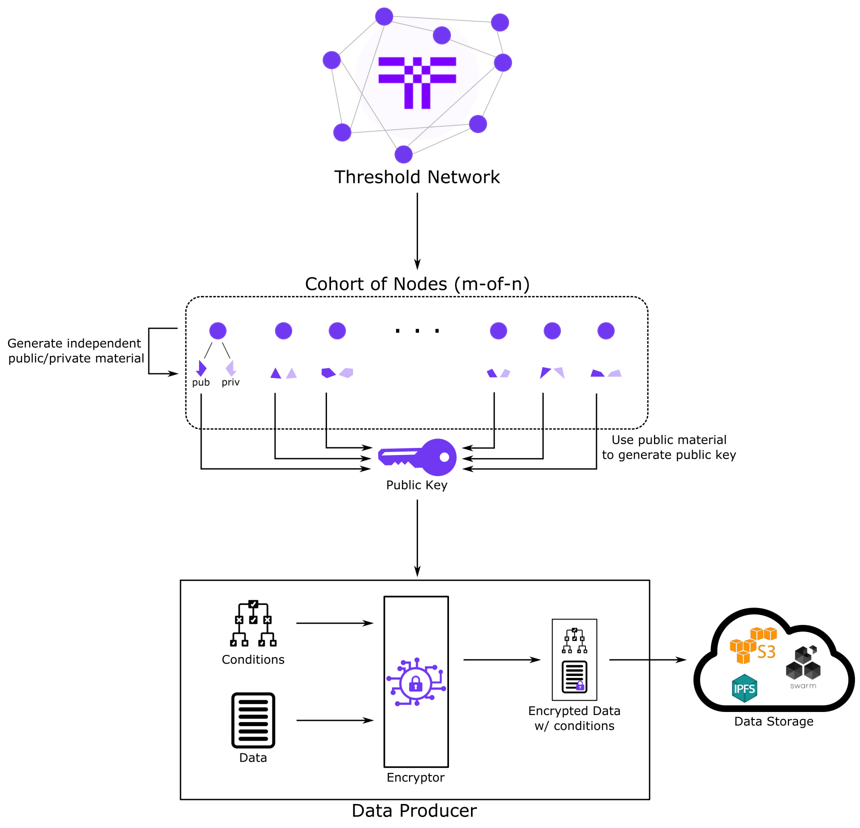


Figure 2.13: CBD encryption flow. Source: GitHub

CBD Decryption process

The Conditions-Based Decryption process starts by retrieving the data produced in the encryption phase by the data owner from the data storage which consist of a block that contains the encrypted data and the decryption condition set. When this recipient obtains these information it is sent a request to the nodes cohort that also contains the previously retrieved block. In order to check the requester's conditions fulfillment an interactive step is now required: since the cohort nodes need to verify certain conditions with respect to a certain Ethereum Wallet, they want a wallet ownership proof performed from the user by a message signature. This signature is then cached so it is asked only the first time. At this time the Ursulas check through the passed Web3 Provider if the wallet satisfy or not the passed condition set, in case of positive answer each of them provides to the user the decryption key fragment related to its node. When the recipient collect a number of fragment equal to the threshold set during the strategy deployment phase he assemble all of them together in order to fully reconstruct the decryption key and finally obtain the plaintext. In case the recipient's wallet doesn't fulfill the message decryption conditions, each cohort node answer with an error message and the plaintext remains obviously inaccessible.

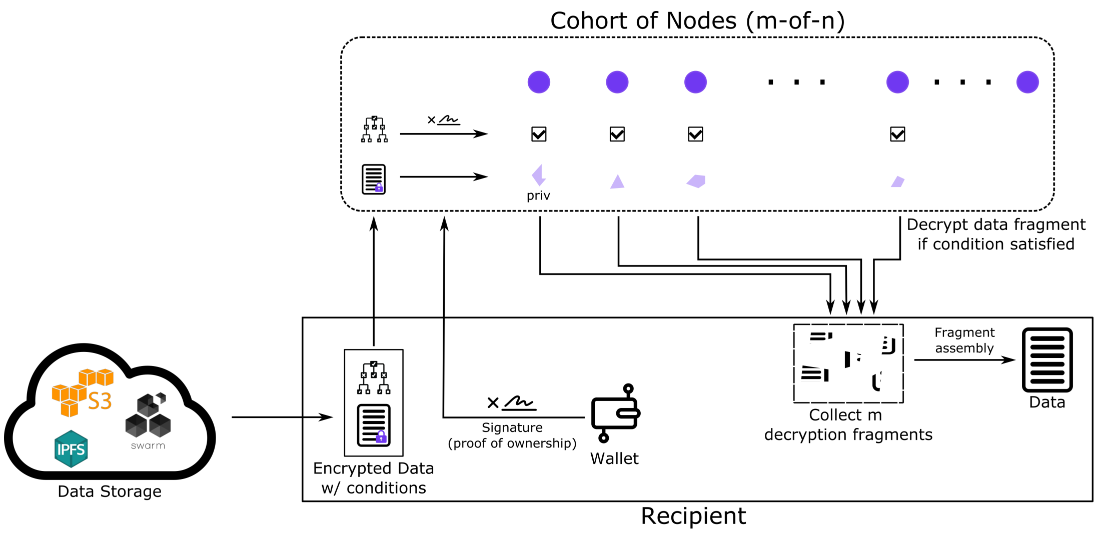

Cohort of Nodes (m-of-n)
priv
Decrypt data fragment if condition satisfied
Fragment assembly
Data
Collect m decryption fragments
Signature (proof of ownership)
Wallet
Encrypted Data w/ conditions
S3
IPFS
swarm
Data Storage
Recipient

Figure 2.14: CBD decryption flow. Source: GitHub

2.2.3. PRE vs. CBD

The two analyzed technologies designed by Threshold Network practically offers the same service to the users that is a secure data end-to-end sharing, but, comparing them, they reveal very important and deep differences, this subsection purpose is to properly explore these differences from different points of view also with respect to the developed application use case.

The most important difference between these two cryptographic technique, that is also the reason for what Conditions-Based Decryption was devised, is that Proxy Re-Encryption's encrypted message can be decrypted and readable by one and only one user, that is specified during the policy deployment phase, this means that if a user wants to encrypt a message to be read from more than one user by using the Proxy Re-Encryption technology he will be forced to create and deploy a number of policy that is equal to the number of recipients he wants to share that message with. This aspect is really important both in terms of usability, because if it is needed to create a very large number of policies the process would take a long time and maybe even be unfeasible, and also economics because when a policy is created the user must pay a fee for the writing operation onto the blockchain, and so paying for a large number of policies would cost too much. Conditions-Based Decrytion, instead, was designed to potentially allow the message access to an infinite number of users by creating and deploying simply one strategy onto the blockchain. As said in the above section, the encrypted message with CBD is related to a decryption condition set from which derive if a user is allowed to access that message or not. So it can be said that in PRE each policy is *per-Bob* and in CBD the conditionSet is *per-ciphertext*. During the study phase of the developed decentralized application it was realized that this PRE's feature was a too big limit for the fixed goal, i.e. an audio and video streaming platform containing files potentially decryptable by any user. CBD offers a very scalable and cheap solution in order to reach this goal ensemble with the possibility to specify a decryption condition set that is a perfect access control method for the encrypted files inside the platform.

Another very important difference can be found on the security aspect, in particular the Threshold Network nodes' jobs are different as we said in the analysis conduced in the previous sections: in PRE applications the Ursula nodes performs proxy re-encryption operations, which means that they change the decryption key from one to another by applying a new encryption to the encrypted data, so they will never be able to access the plaintext. This means that in the case in which malicious nodes take part to the process, also if in number that is greater then or equal to the policy threshold, they will never be able to access the plaintext because they do not acquire the message decryption key. The only attack that they can carry out is a *denial-of-service* attack but the encrypted data will always remain secret and inaccessible. CBD cohort nodes's job, instead, consists of return to the arbitrary recipient a decryption key fragment, that he will only need to put all of them together in order to fully reconstruct the key. In this case, if malicious nodes that take part to the process are at least equal to the strategy's threshold they will be able also to access the plaintext simply by putting together the key fragments decrypted by them.

We can finally conclude that Proxy Re-Encryption employs a more secure and reliable service from the data privacy point of view but Conditions-Based Decryption offers a more scalable solution for the data distribution point of view, still offering a quite high security level.

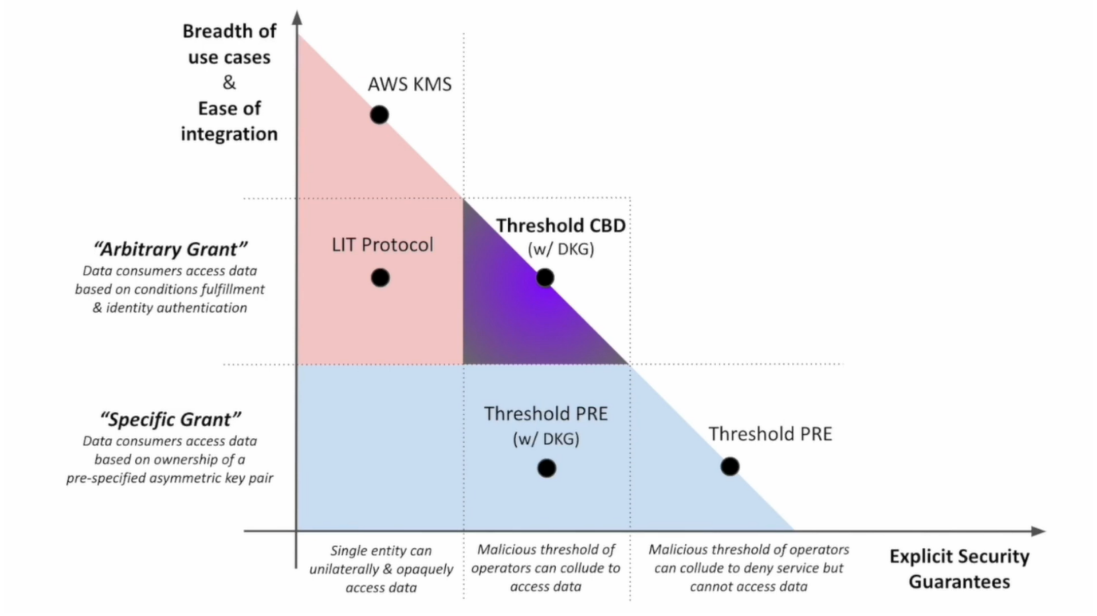

Breadth of use cases & Ease of integration
"Arbitrary Grant"
Data consumers access data based on conditions fulfillment & identity authentication
"Specific Grant"
Data consumers access data based on ownership of a pre-specified asymmetric key pair
AWS KMS
LIT Protocol
Threshold CBD (w/ DKG)
Threshold PRE (w/ DKG)
Threshold PRE
Single entity can unilaterally & opaquely access data
Malicious threshold of operators can collude to access data
Malicious threshold of operators can collude to deny service but cannot access data
Explicit Security Guarantees

Figure 2.15: Technologies security and scalability comparison diagram. Source: YouTube

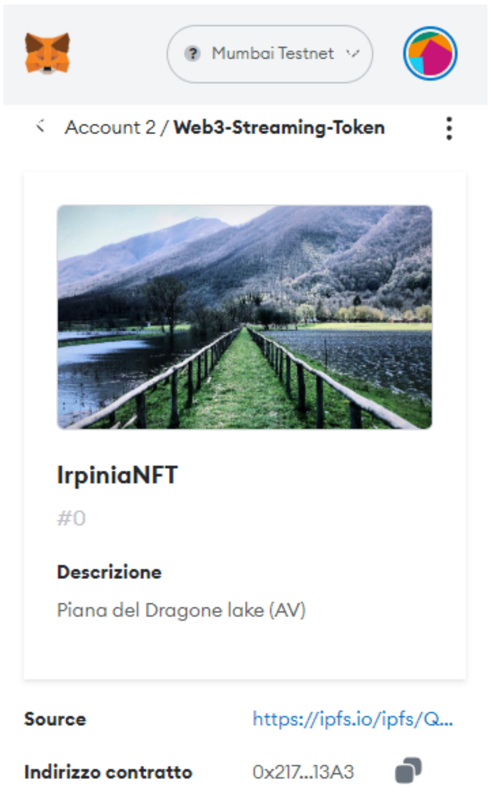
3 | Final project and achieved results

As introduced in the previous chapters, the project consists of a web3 decentralized platform that allows authorized users to access on-demand audio and video contents while using a very secure and reliable cryptographic technology: *Conditions-Based Decryption (CBD)*. The decentralized application, born from a collaboration with the company KNOBS, is a *React App* that take advantage of the *Webpack* module bundler to manage the project dependencies and it was developed with *Typescript* as programming language. The application is made up of two pages: the first page is dedicated to the encryption phase, the second one instead to the decryption phase, every nuance regarding these pages will be thoroughly analyzed and explained in this chapter. CBD consists of allowing the access to the files available on the platform only to users whose wallet fulfills the required decryption conditions, starting from this assumption the main idea was to enable decryption of files based on whether the data requester's wallet owns a certain *Non-Fungible Token (NFT)* or not, so one was created: the *IrpiniaNFT*. The application moreover offers other decryption condition types, as it will be seen in this chapter in which it will be firstly described the token creation and minting process and successively a focus on the application will be done.

3.1. IrpiniaNFT

As stated above, this NFT has been created for the *Proof-Of-Concept*, in order to relate some files decryption to the user's wallet ownership of this token. The smart contract creation was done thanks to the *Contract Wizard* of *OpenZeppelin* that allows users to write all kinds of contracts, among which *ERC-721* standard one, used to create the contract to which IrpiniaNFT belongs, whose name is *Web3-Streaming-Token* and symbol is *W3ST*. Successively, to exploit the contract deployment and token minting processes the *Remix IDE* from *Ethereum* in collaboration with *IPFS* where the token's *baseURI* and metadata are stored. The *Web3-Streaming-Token (W3ST)* smart contract was deployed on the *Mumbai Network* which is a testnet with *Chain ID* = 80001, the choice of using this network comes from the fact that, for which concerns the developing and testing phase, the smart contract with whom the application's interactions take place in order to write the CBD strategy details and pay the related fees runs on this network.

The Web3-Streaming-Token (W3ST) ERC-721 contract's address is 0x21739b933261efF4-bDE8d375796Bd40Eca8713A3 and the IrpiniaNFT's tokenID is 0. Contract details can be found on Mumbai Polygonscan

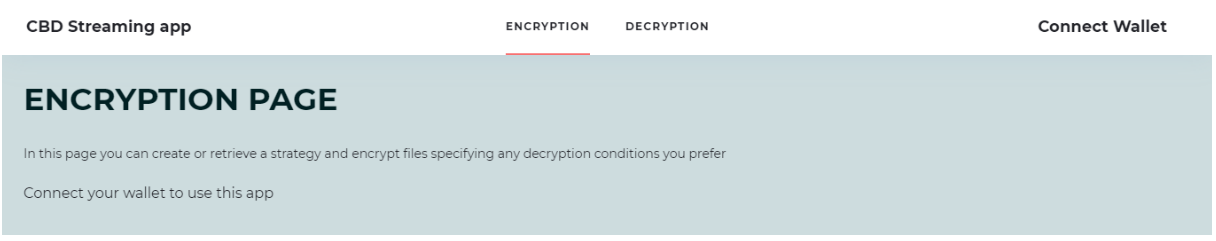


IrpiniaNFT
#0
Descrizione
Piana del Dragone lake (AV)
Source https://ipfs.io/ipfs/Q...
Indirizzo contratto 0x217...13A3

Figure 3.1: IrpiniaNFT MetaMask screenshot

3.2. Files encryption phase

The first step of the application workflow consists of the file encryption phase, in order to interact with this page each user need to connect his MetaMask account to the website by clicking the proper button on the right in the TopBar.



ENCRYPTION PAGE

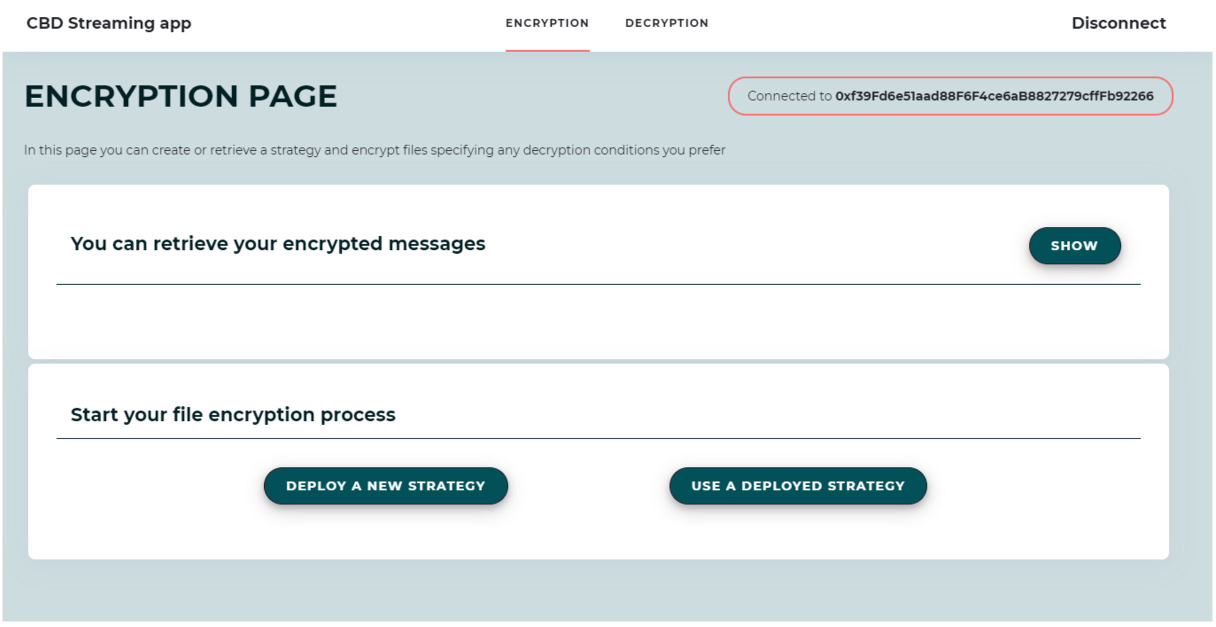
In this page you can create or retrieve a strategy and encrypt files specifying any decryption conditions you prefer

Connect your wallet to use this app

Figure 3.2: Wallet connection phase

3.2.1. CBD Strategy building

Once the MetaMask wallet connection is correctly established the user sees the main page, which contains two sections, the first one allows the user to retrieve from Pinata, the used IPFS service used as data storage, the files previously encrypted with the currently connected account, but this section will be later analyzed. The second section, instead, consists of the first step of the file encryption process that is the choice between deploying a new Conditions-Based Decryption strategy on-chain and selecting one among the user's already deployed ones saved on Pinata. This latter feature is very useful in order to use the same strategy for multiple files encryption without the need of deploying a new one and consequently pay the writing fee every time.



ENCRYPTION PAGE

Connected to 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

In this page you can create or retrieve a strategy and encrypt files specifying any decryption conditions you prefer

You can retrieve your encrypted messages

Start your file encryption process

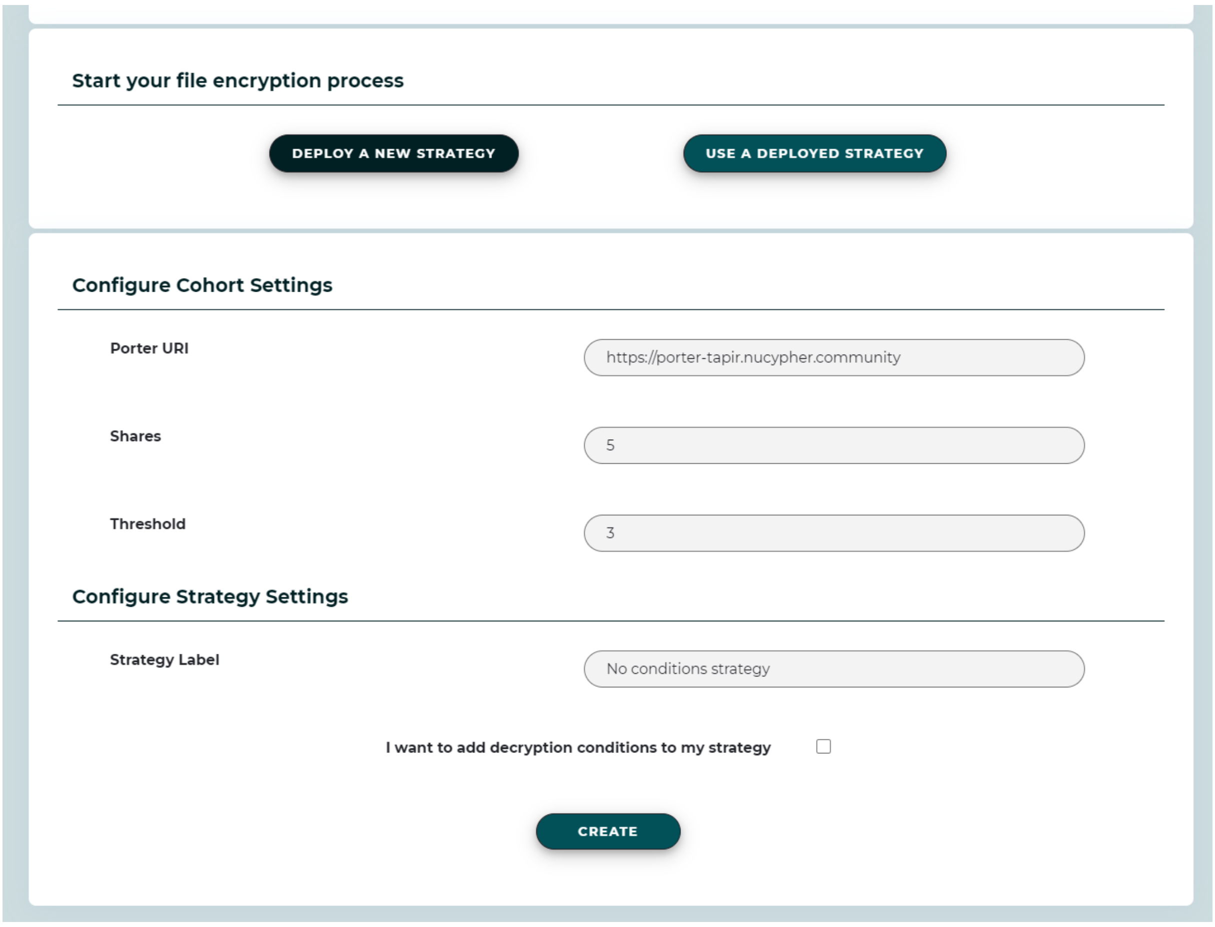
Figure 3.3: Encryption process starting

Stategy deployment

When the user decides to create and deploy on-chain a new strategy it will be necessary to configure two kind of settings: the *Cohort* and the *Strategy* settings. The cohort, as explained in the section 2.2.2, is the set of Threshold Network nodes responsible for the *Decentralized Key Generation* process in order to construct the asymmetric encryption key to be used subsequently, and for the management of the conditions checking during the decryption process. The Cohort settings definition consists of specifying the URI of the *Porter*, the number of nodes to be considered in the cohort and the threshold of nodes' responses needed to decrypt the message. The *Porter* is a sort of gateway for the Threshold Network, it is a web-based service which performs *nucypher-based* protocol operations on behalf of applications, its goal is to simplify the interaction with the nucypher protocol operations and to avoid that the application need to interact with this protocol through a python client. Moreover, the Porter entity allows cross-platform applications, such as web and mobile ones, to exploit the nucypher functionality.[5] By default, the Porter URI inserted in the input field is a public one provided by *Tapir*: "https://porter-tapir.nucypher.community".

The strategy settings configuration, instead, consist of simply specifying a label to be associated with the strategy and, optionally, a decryption condition set to be attached to all files that will be encrypted with the strategy being created. This feature is very useful in the case in which a user needs to encrypt a lot of files with the same decryption condition set. Since the conditions setting phase is a crucial point and the chosen ones are attached to the encrypted file, we will analyze and explain it later in the section.

The below example, the figure 3.4, shows an example of a new strategy creation, built with the default *Tapir Porter URI*, the amount of randomly chosen Threshold nodes involved in the *Decentralized Key Generation* equal to 5 and the minimum number of nodes to decrypt files related to this strategy equal to 3, for which concerns the cohort. For which concerns the strategy settings, the entered label is "No conditions strategy" because no decryption conditions are specified for this strategy, as can be seen from the unchecked checkbox, it will be necessary to add decryption conditions in the next step.


Start your file encryption process

Configure Cohort Settings

Porter URI: https://porter-tapir.nucypher.community

Shares: 5

Threshold: 3

Configure Strategy Settings

Strategy Label: No conditions strategy

I want to add decryption conditions to my strategy ☐

Figure 3.4: Strategy creation example

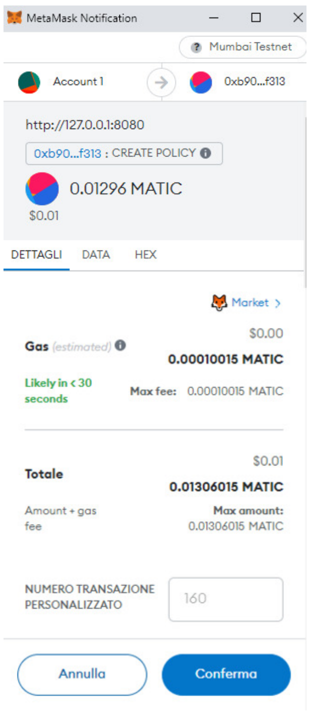
Once the settings are established, the "Create" button click make the strategy deployment process start: the first step consists of making a post request to the Porter URI in order to retrieve the addresses of the requested nodes and start with the Decentralized Key Generation. Successively, the strategy is written onto the blockchain (Mumbai Testnet) thanks to the method *createPolicy(bytes16, address, uint16, uint32, uint32)* of the smart contract *SubscriptionManager* specifying the off-chain generated policyID, the policy owner i.e. the user itself, the number of shares and a start date and end date (in the form of timestamp). The last two parameters are useless and will be removed in the next CBD release. The fee that the user is called to pay is computed by a simple calculation:

$$transactionCost = feeRate \times duration \times numberOfNodes \tag{3.1a}$$

where duration is endTimestamp - startTimestamp.[4]

The fee payment is made through MetaMask and the relative notification shown to the user can be found in figure 3.5. The underlined smart contract details can be explored on Mumbai Polygonscan.

Moreover, when the strategy deployment ends successfully, as we can see in figure 3.6 the application asks the user to sign a message with his wallet in order to get the relative secret key to store on Pinata, under his MetaMask account, the newly deployed strategy.


MetaMask Notification

Mumbai Testnet

Account 1 → 0xb90...f313

http://127.0.0.1:8080

0xb90...f313 : CREATE POLICY

0.01296 MATIC

$0.01

DETTAGLI DATA HEX

Market >

Gas (estimated) $0.00 0.00010015 MATIC

Likely in < 30 seconds Max fee: 0.00010015 MATIC

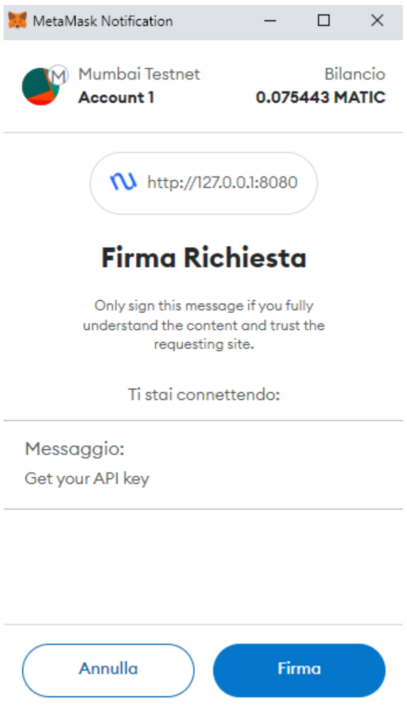
Totale $0.01 0.01306015 MATIC

Amount + gas fee Max amount: 0.01306015 MATIC

NUMERO TRANSAZIONE PERSONALIZZATO 160

Annulla Conferma

Figure 3.5: Transaction's fee payment


MetaMask Notification

Mumbai Testnet Bilancio

Account 1 0.075443 MATIC

http://127.0.0.1:8080

Firma Richiesta

Only sign this message if you fully understand the content and trust the requesting site.

Ti stai connettendo:

Messaggio:

Get your API key

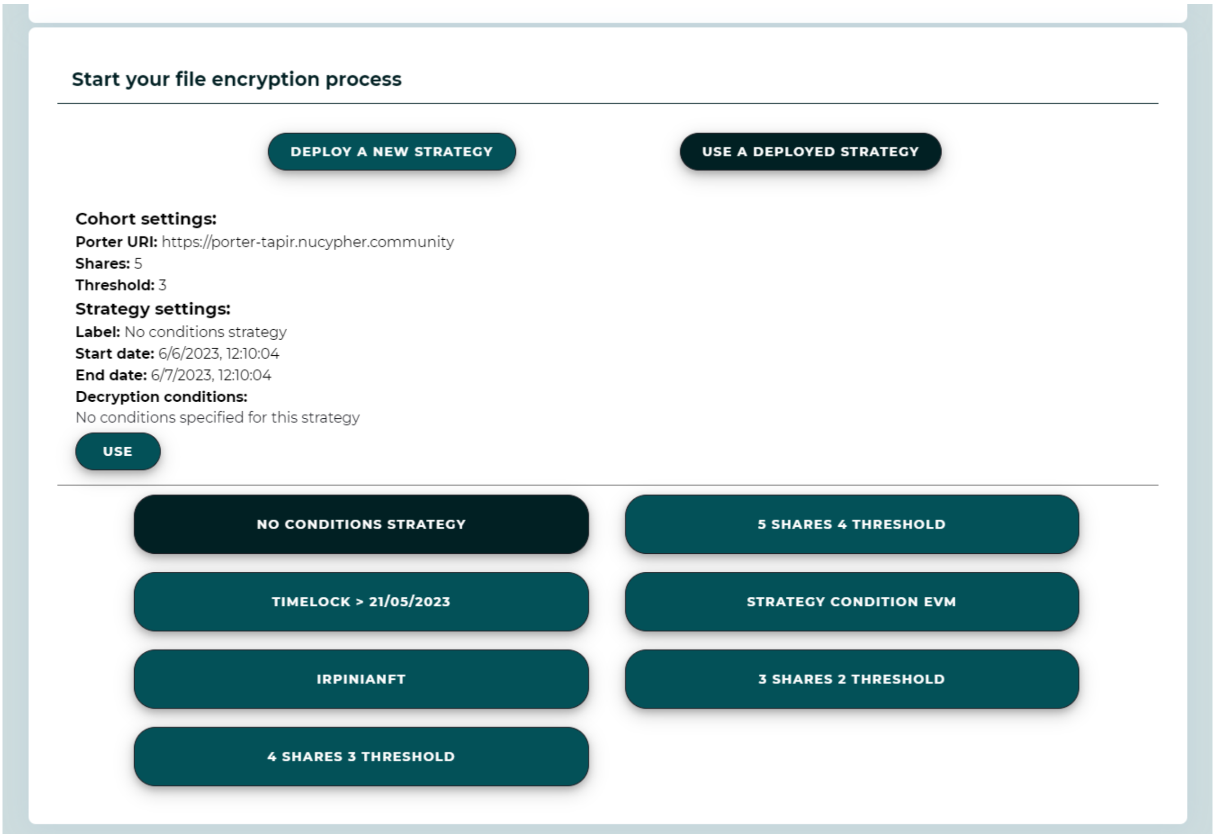
Annulla Firma

Figure 3.6: Message signature for the Pinata key generation

Stategy selection from IPFS

Before moving to the conditions set insertion step it is good to highlight also the other method that can be used in order to move to the encryption phase: using an already deployed strategy chosen from IPFS, if present. In order to retrieve the currently connected user's encryption strategies from Pinata, as seen above, it is necessary to provide the wallet signature to generate the Pinata secret key to retrieve the files. This interactive step is necessary only at the first request, after which the key is cached. Once the user clicks the right button "Use a deployed strategy", shown in figure 3.3, all the user's strategies are retrieved and the user can also visualize all the details regarding each of them before deciding which use. As said, this application feature is useful to avoid paying a transaction fee for each message to be encrypted and also to reuse the same decryption condition set for different files without the needing of entering it every time.

In the below example, figure 3.7, are shown seven buttons related to the strategies already deployed on-chain with the currently connected account, each of them is represented by the label inserted during the creation phase. As can be noticed, the just deployed one with label "No conditions strategy" has been clicked and the relative details are shown, among which we can find the decryption conditions inserted. This is the strategy that will be used in order to go ahead with the encryption process.


Start your file encryption process

Cohort settings:
Porter URI: https://porter-tapir.nucypher.community
Shares: 5
Threshold: 3
Strategy settings:
Label: No conditions strategy
Start date: 6/6/2023, 12:10:04
End date: 6/7/2023, 12:10:04
Decryption conditions:
No conditions specified for this strategy

Figure 3.7: Deployed strategies retrieved from Pinata

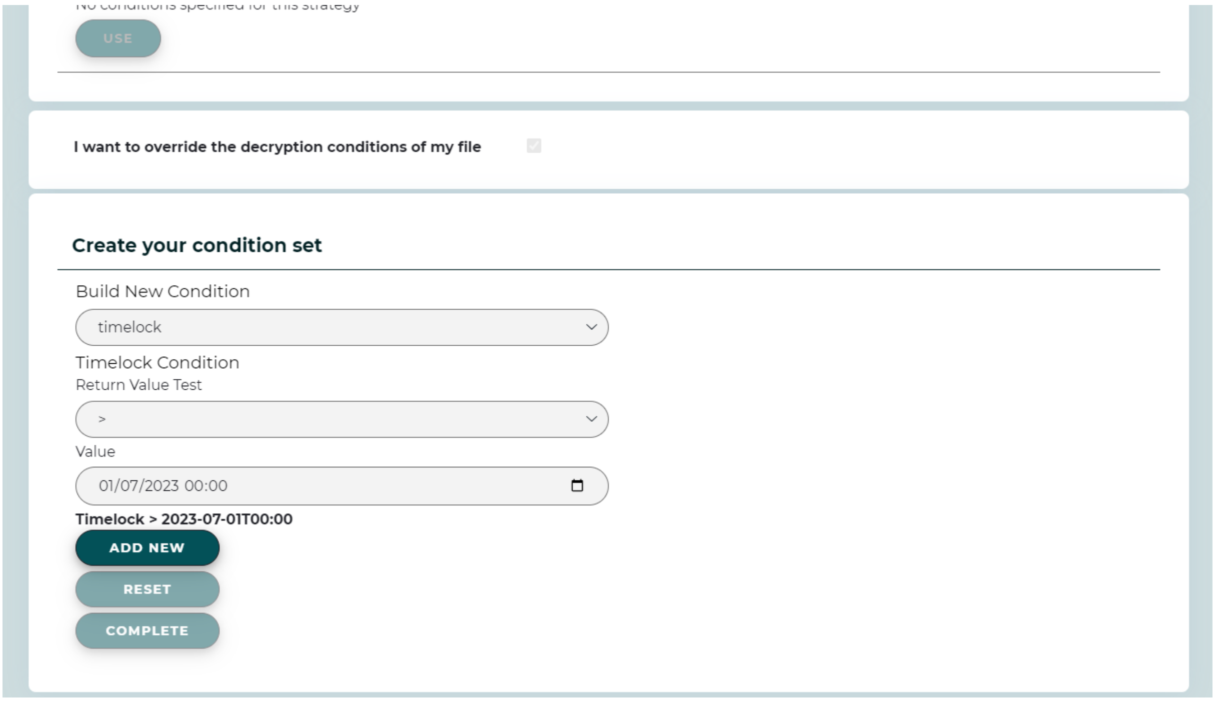
3.2.2. Condition set entry

The key concept of the project comes when it is needed to specify the decryption conditions to be then attached to the encrypted file. If in the previous step the selected strategy to be used contains an empty condition set, it is mandatory to specify one before moving to the file selection. Otherwise, if the used strategy contains a condition set it is possible to use it or also to overwrite it by checking the related checkbox and inserting a new one for a specific file, still using that strategy. This is possible because the conditions are *per-ciphertext*: they are encapsulated within the encrypted message, so it is possible also to overwrite them before the file encryption.

The condition set can be made up of three different condition types that can be also concatenated in a hierarchy tree thanks to the two logical operators *AND/OR* that allows to put more conditions together.

Timelock-based condition

The first available condition type is the *Timelock-based* one, it allows to make a file decryption available or unavailable basing on temporal conditions, as can be seen in figure 3.8 the user can insert an operator to compare the return value, that for this condition type is the current timestamp, to the date and hour value that can be inserted through the calendar. For the working example it is inserted a timelock condition that makes the file undecryptable until the 1st of July 2023 at 00:00.



No conditions specified for this strategy

I want to override the decryption conditions of my file

Create your condition set

Build New Condition: timelock

Timelock Condition

Return Value Test: >

Value: 01/07/2023 00:00

Timelock > 2023-07-01T00:00

Figure 3.8: Timelock-based condition.

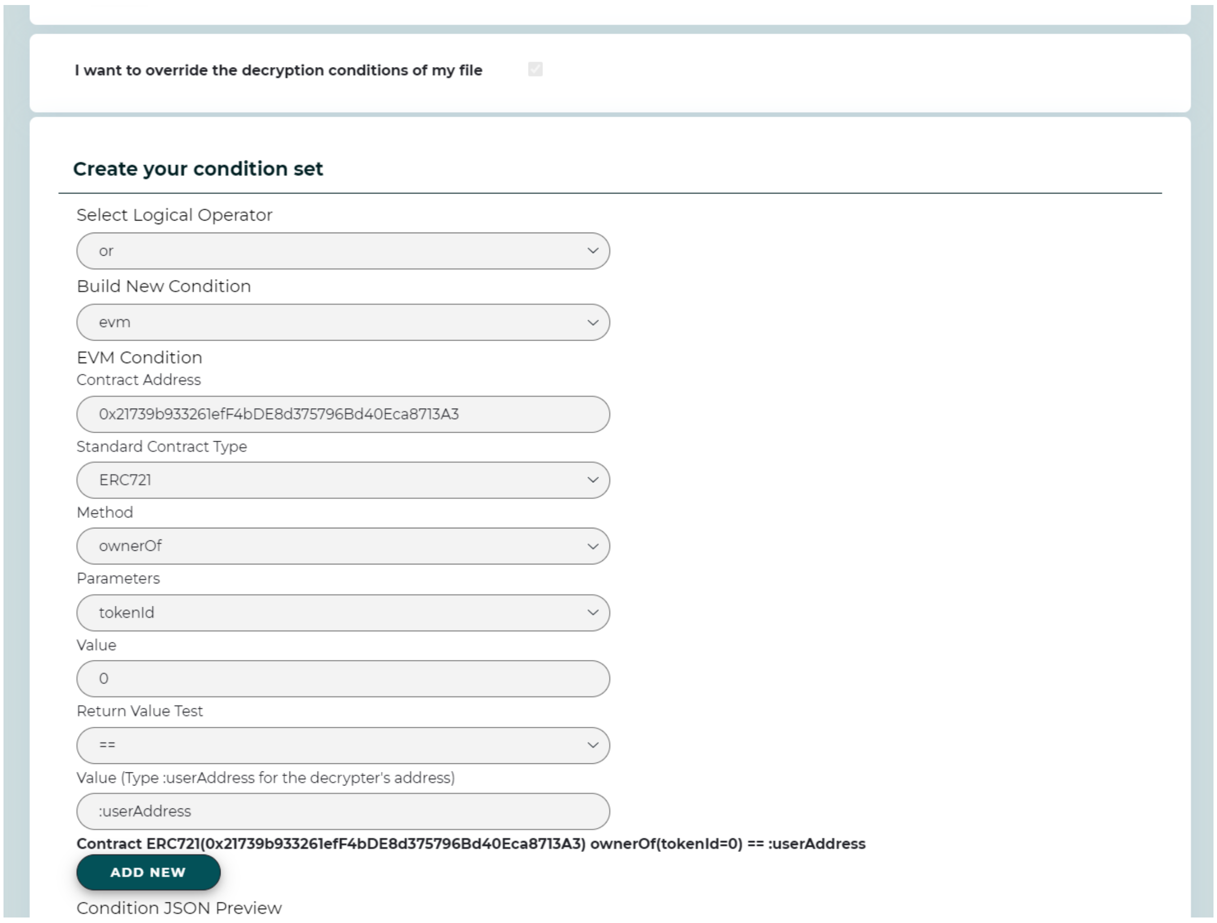
EVM-based condition

The second available condition type is the *EVM-based* one: it allows users to enable files decryption basing on token ownership. It is possible to specify this condition for the two most famous token standards: *ERC-20* for fungible tokens and *ERC-721* for non-fungible tokens. Basing on the selected standard different methods are offered, it is good to analyze all the possibility the user can face. If *ERC-20* is selected the user, after the smart contract address specification to which the token refers, can call the method *balanceOf()*, which is a method of ERC-20 contracts that need to receive a wallet address as parameter in order to return the amount of tokens owned by it. The address parameter must be specified in the proper input field and if the user wants to pass to the method the decrypting user's address it is necessary to type ":userAddress". And then he can compare the return value with the any value and operator.

Instead, when *ERC-721* is selected the user can choose between two methods: *balanceOf()* and *ownerOf()*. The first method is a ERC-721 method which, by receiving as parameter a wallet address, returns the number of NFTs belonging to the smart contract on which the method is called owned by that wallet. The second method, instead, receives as parameter a *TokenID* and returns the owner address of that tokenID belonging to the smart contract on which the method is called. So the application can call these two methods on the inserted smart contract address and passing the selected parameter. As before, the user can compare the return value with any value he wants.

It is good to underline that the, since the platform works with the *Mumbai Testnet*, this method works with smart contracts and tokens running on it.

In the below example, the figure 3.9, it is inserted an EVM-based condition that makes the file inaccessible if the user that will require the file doesn't own the IpriniaNFT, that belongs to the Web3-Streaming-Token smart contract: address 0x21739b933261efF4bDE8d3-75796Bd40Eca8713A3 and has TokenID = 0. This condition is added to the previously created condition set with the OR operator.


I want to override the decryption conditions of my file

Create your condition set

Select Logical Operator: or

Build New Condition: evm

EVM Condition

Contract Address: 0x21739b933261efF4bDE8d375796Bd40Eca8713A3

Standard Contract Type: ERC721

Method: ownerOf

Parameters: tokenId

Value: 0

Return Value Test: ==

Value (Type :userAddress for the decrypter's address): :userAddress

Contract ERC721(0x21739b933261efF4bDE8d375796Bd40Eca8713A3) ownerOf(tokenId=0) == :userAddress

ADD NEW

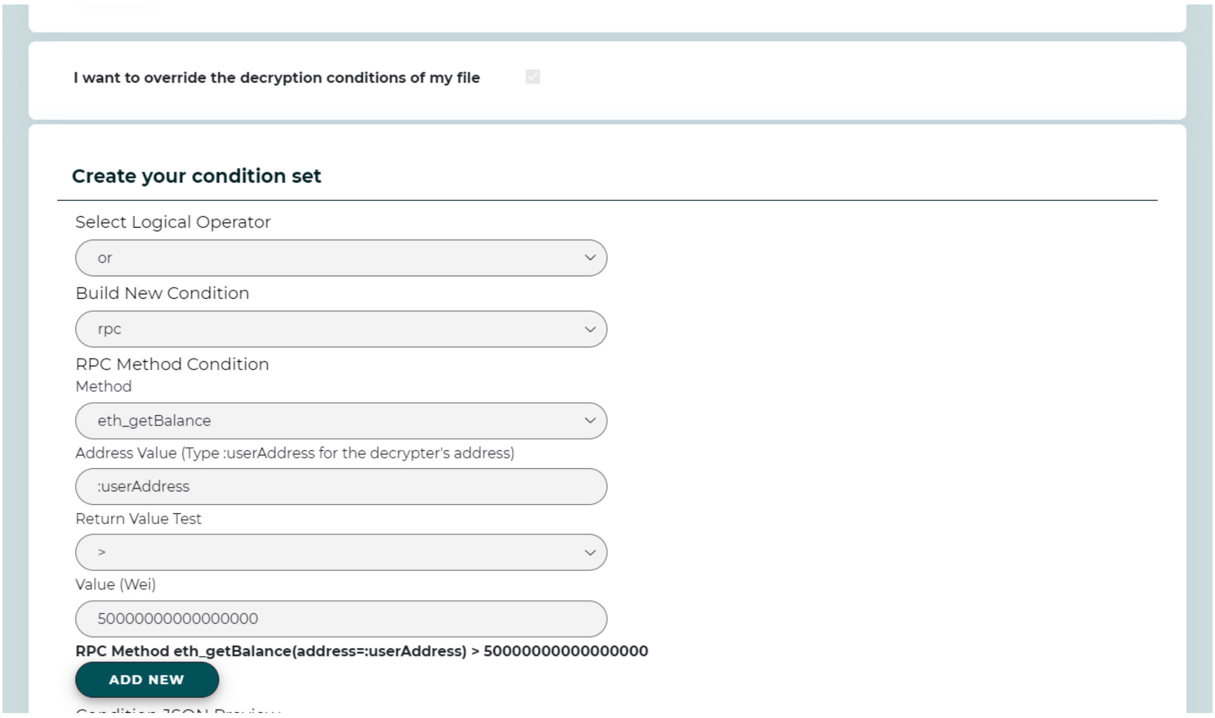
Condition JSON Preview

Figure 3.9: EVM-based condition.

RPC-driven condition

The last condition type provided by CBD is the *RPC-driven* one that allows to make the file decryptable or not basing on money conditions. This kind of condition can be used with two different methods: `eth_getBalance()` and balanceOf(). The first one requires a wallet address as parameter and returns the account balance related to the chainID token required. The second method, instead, requires a smart contract address as parameter and returns its balance. After choosing the method to use and entering the relative paramter to pass the user can insert the return value comparison in order to build the condition, as can be seen in the platform the comparing value must be specified in *Wei* unit of measurement.

As the previous method, also this one works on *Mumbai Testnet*, therefore the cryptocurrency considered is *Matic*. In the below example, figure 3.10, the RPC-driven specified condition calls the `eth_getBalance()` method on the data requester's address and checks if its balance is greater than 0.05 MATIC, that becomes 50000000000000000 once converted to Wei. As before, this condition has been added to the condition set with the OR operator.



I want to override the decryption conditions of my file

Create your condition set

Select Logical Operator: or

Build New Condition: rpc

RPC Method Condition

Method: eth_getBalance

Address Value (Type :userAddress for the decrypter's address): :userAddress

Return Value Test: >

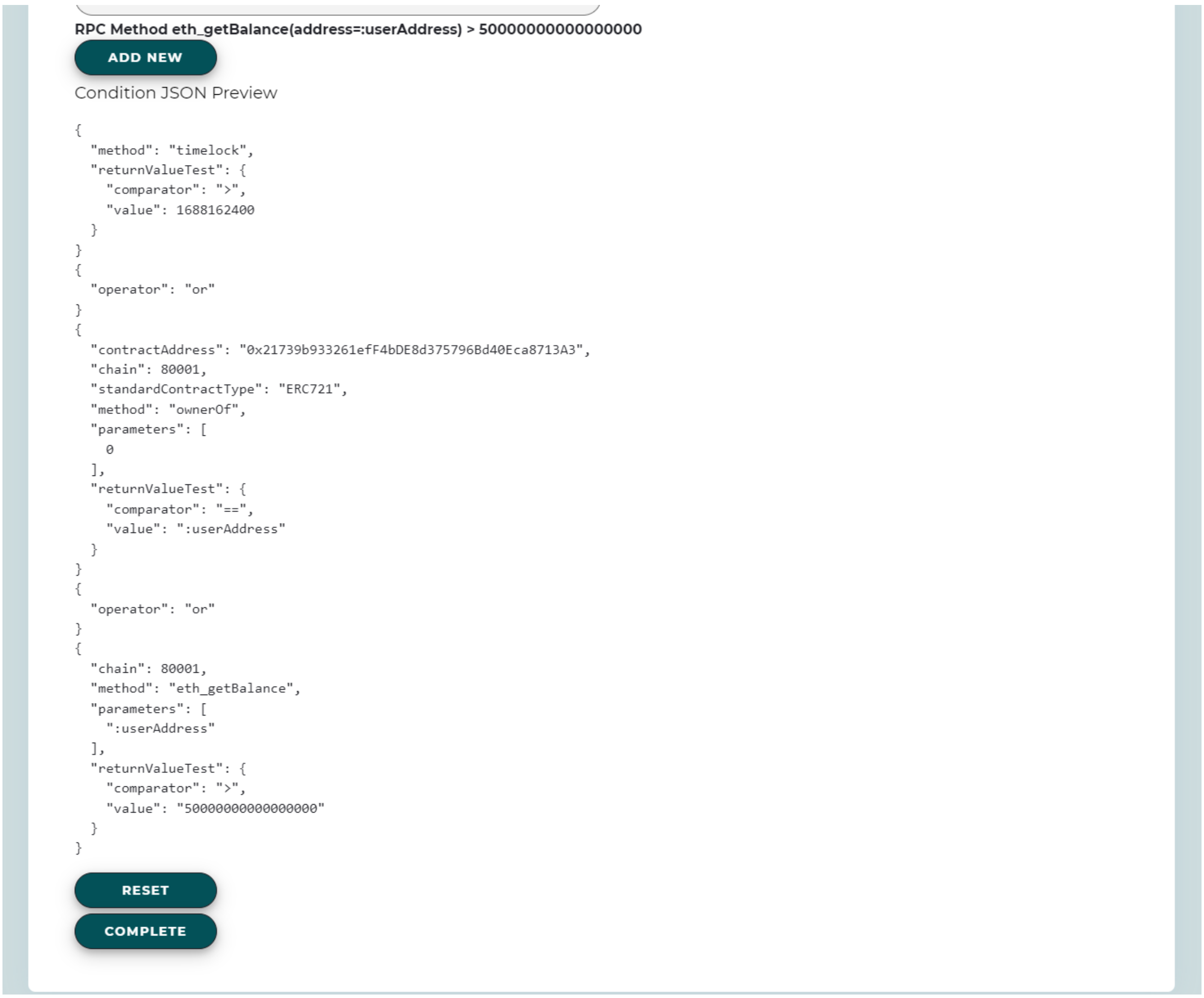
Value (Wei): 50000000000000000

RPC Method eth_getBalance(address=:userAddress) > 50000000000000000

Figure 3.10: RPC-based condition.

When each condition is added to the condition set it is shown a JSON preview that recaps the inserted condition set, in order to double check the decryption rules before moving to the file encryption phase. From the screenshot below, the figure 3.11, it can be seen the JSON related to the inserted condition set, in this case they can be found three conditions, one for each type, concatenated with OR operators. The below condition set meaning is that the file that will be associated to it will be decryptable only after the 1st of July 2023 at 00:00 unless the data requester owns the IrpiniaNFT in his wallet or his wallet's balance on the Mumbai Testnet is greater than 0.05 MATIC.

It is good to underline that the just outlined process is identical if the user decides to add a condition set to the strategy during the deployment phase. In case the strategy is already associated with a condition set this step can be skipped in order to directly pass to the file encryption process or, if the user wants to change it, he can overwrite it thanks to the special checkbox in order to make some modification to conditions for a specific file.


RPC Method eth_getBalance(address=:userAddress) > 50000000000000000

Condition JSON Preview

```
{
  "method": "timelock",
  "returnValueTest": {
    "comparator": ">",
    "value": 1688162400
  }
}
{
  "operator": "or"
}
{
  "contractAddress": "0x21739b933261efF4bDE8d375796Bd40Eca8713A3",
  "chain": 80001,
  "standardContractType": "ERC721",
  "method": "ownerOf",
  "parameters": [
    0
  ],
  "returnValueTest": {
    "comparator": "==",
    "value": ":userAddress"
  }
}
{
  "operator": "or"
}
{
  "chain": 80001,
  "method": "eth_getBalance",
  "parameters": [
    ":userAddress"
  ],
  "returnValueTest": {
    "comparator": ">",
    "value": "50000000000000000"
  }
}
```

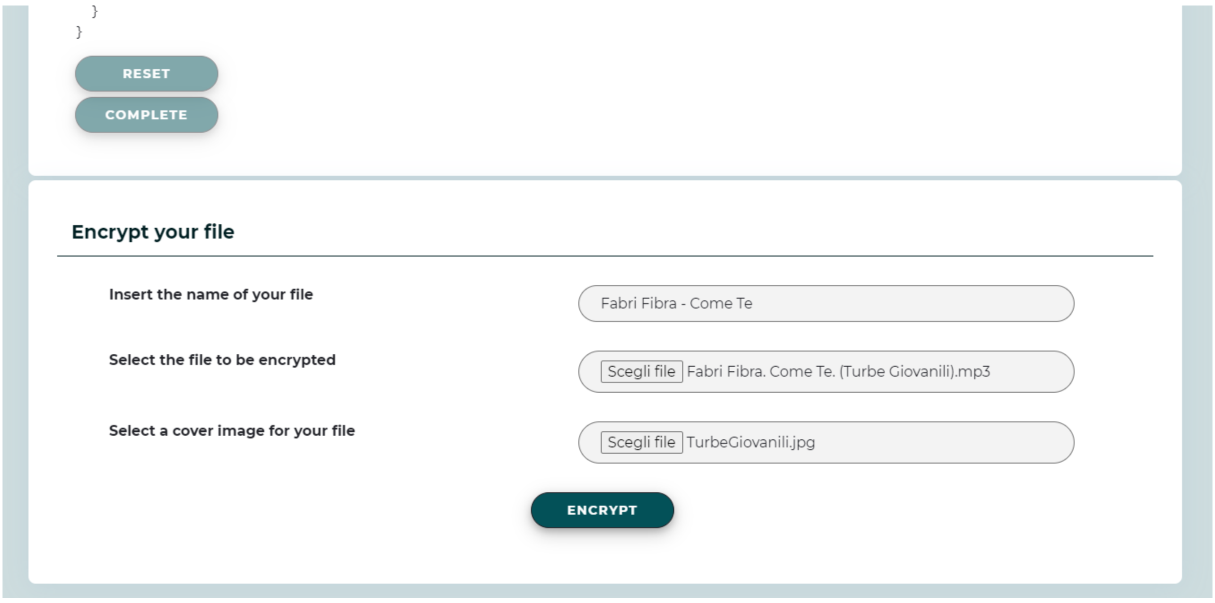
Figure 3.11: Decryption condition set building.

3.2.3. File encryption

When the decryption conditions are successfully established, in order to go ahead to the file encryption phase, the user needs to click the "Complete" button that will be enable the file upload. If the user notices that between his condition set there is something wrong he can, thanks to the "Reset" button, go back and repeat the condition entering process. When going ahead another content block appears on the page, this content block contains three input fields: the first and the third ones are needed to specify the name and the image related to the file he wants to encrypt that the user wants displayed on the decryption page, while the second one is the audio or video file itself.

Once these input field are properly filled the decryption process can be run through the "Encrypt" button: when this action is performed it happens that the file is sent to the encrypter entity ensemble with the chosen condition set that will encrypt the plaintext and encapsulate it with the conditions. When this operation is successfully completed the encrypted message, encoded in a *base64* string, is uploaded on IPFS Pinata service divided in different chunks with other data that will be needed during the decryption phase, among these we can obviously find the file's title and image, a reference to the strategy used and also the decryption condition set. When uploading the file to Pinata, in the file metadata it is also specified a reference to the MetaMask wallet that performed this operation in order to, as it will be explained in the next subsection, retrieve the files encrypted and uploaded by the currently connected user.

In the example in figure 3.12 it has been selected an audio file to be encrypted, the related title, "Fabri Fibra - Come Te", and cover image to be shown inside the decryption page.



Encrypt your file

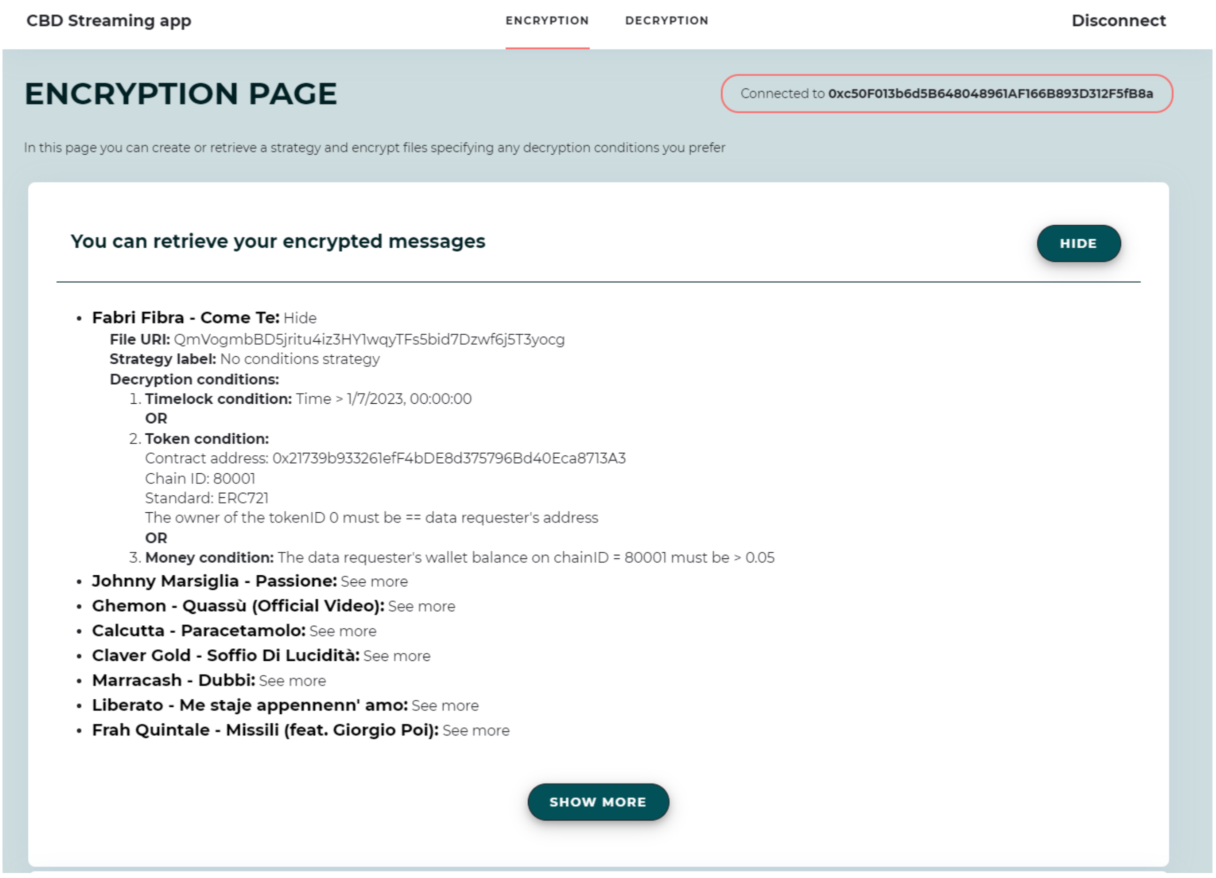
Insert the name of your file	Fabri Fibra - Come Te
Select the file to be encrypted	Scegli file Fabri Fibra. Come Te. (Turbe Giovanili).mp3
Select a cover image for your file	Scegli file TurbeGiovanili.jpg

Figure 3.12: Filled file encryption input fields.

3.2.4. Encrypted and uploaded files visualization

At the beginning of this section it was mentioned the first content block that is dedicated to the uploaded encrypted messages retrieving. This section allows users to browse between the files encrypted and uploaded on Pinata by the currently connected wallet with the related details. If the user is new to the platform and no file has been uploaded by him the message "Sorry, zero files found!" will be shown. Otherwise, the menu displays an unordered list with the title of each file and an hypertext that allows users to retrieve details related to the selected file: the IPFS CID, the strategy label thanks to which the file was encrypted, and the detailed decryption condition. If the user wants to access the strategy details he can do it thanks to the second content block, as said in the above subsections.

This feature is useful in order to allow any user to track and consult any file encrypted and uploaded to the streaming platform. The figure 3.13 shows an example of the explained process, it can be noticed that, among the other files encrypted with the currently connected wallet, it can be found the file encrypted during this workflow example, of which details are also shown.



ENCRYPTION PAGE

Connected to **0xc50F013b6d5B648048961AF166B893D312F5fB8a**

In this page you can create or retrieve a strategy and encrypt files specifying any decryption conditions you prefer

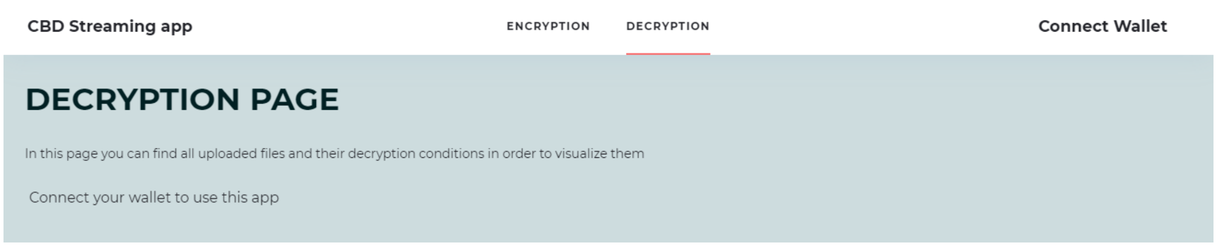
You can retrieve your encrypted messages

- **Fabri Fibra - Come Te:** Hide
  File URI: QmVogmbBD5jritu4iz3HY1wqyTFs5bid7Dzwf6j5T3yocg
  Strategy label: No conditions strategy
  Decryption conditions:
  1. **Timelock condition:** Time > 1/7/2023, 00:00:00
     OR
  2. **Token condition:**
     Contract address: 0x21739b933261efF4bDE8d375796Bd40Eca8713A3
     Chain ID: 80001
     Standard: ERC721
     The owner of the tokenID 0 must be == data requester's address
     OR
  3. **Money condition:** The data requester's wallet balance on chainID = 80001 must be > 0.05
- **Johnny Marsiglia - Passione:** See more
- **Ghemon - Quassù (Official Video):** See more
- **Calcutta - Paracetamolo:** See more
- **Claver Gold - Soffio Di Lucidità:** See more
- **Marracash - Dubbi:** See more
- **Liberato - Me staje appennenn' amo:** See more
- **Frah Quintale - Missili (feat. Giorgio Poi):** See more

Figure 3.13: Encrypted files retrieved from IPFS.

3.3. Files decryption phase

After outlining in depth the platform encryption side, it must be properly analyzed and explained the files decryption page. Equally to the encryption page, when a user enters this page no content can be displayed until his MetaMask wallet is successfully connected to the platform. As the decentralized applications majority, in order to interact with the platform this connection operation is necessary, in detail in this dApp's decryption page, as widely said, to decrypt files the conditions must be met by the MetaMask wallet. The figure 3.14, shows the page the user faces when enters the app.



DECRYPTION PAGE

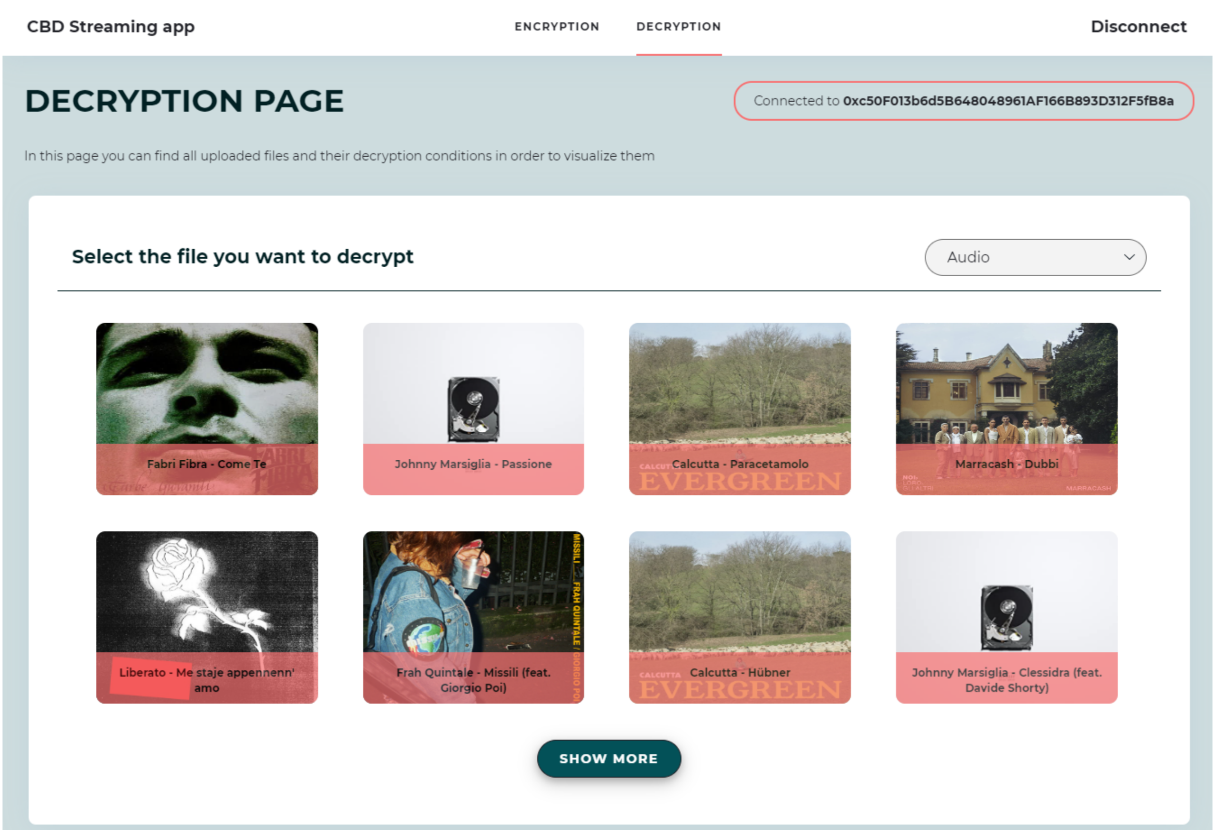
In this page you can find all uploaded files and their decryption conditions in order to visualize them

Connect your wallet to use this app

Figure 3.14: Wallet connection phase.

3.3.1. Encrypted files retrieving

When the wallet connection is successfully completed the encrypted files are retrieved from Pinata, it is initially shown only one row, made up of four files, but the user can download the subsequent ones by clicking the button "Show more". The encrypted files are displayed through the images and file titles specified during the file encryption phase. On the top of the content block, on the right, the user can find a dropdown menu that allows him to filter the files basing on the type: audio files, video files and all files, by default the dropdown value is "Audio", so when the user enters the platform and connects the wallet he browse the uploaded audio files, as shown in figure 3.15, where two file rows have been downloaded.


CBD Streaming app
ENCRYPTION
DECRYPTION
Disconnect
Audio
Fabri Fibra - Come Te
Johnny Marsiglia - Passione
Calcutta - Paracetamolo
Marracash - Dubbi
Liberato - Me staje appennenn' amo
Frah Quintale - Missili (feat. Giorgio Poi)
Calcutta - Hübner
Johnny Marsiglia - Clessidra (feat. Davide Shorty)
SHOW MORE

DECRYPTION PAGE

Connected to 0xc50F013b6d5B648048961AF166B893D312F5fB8a

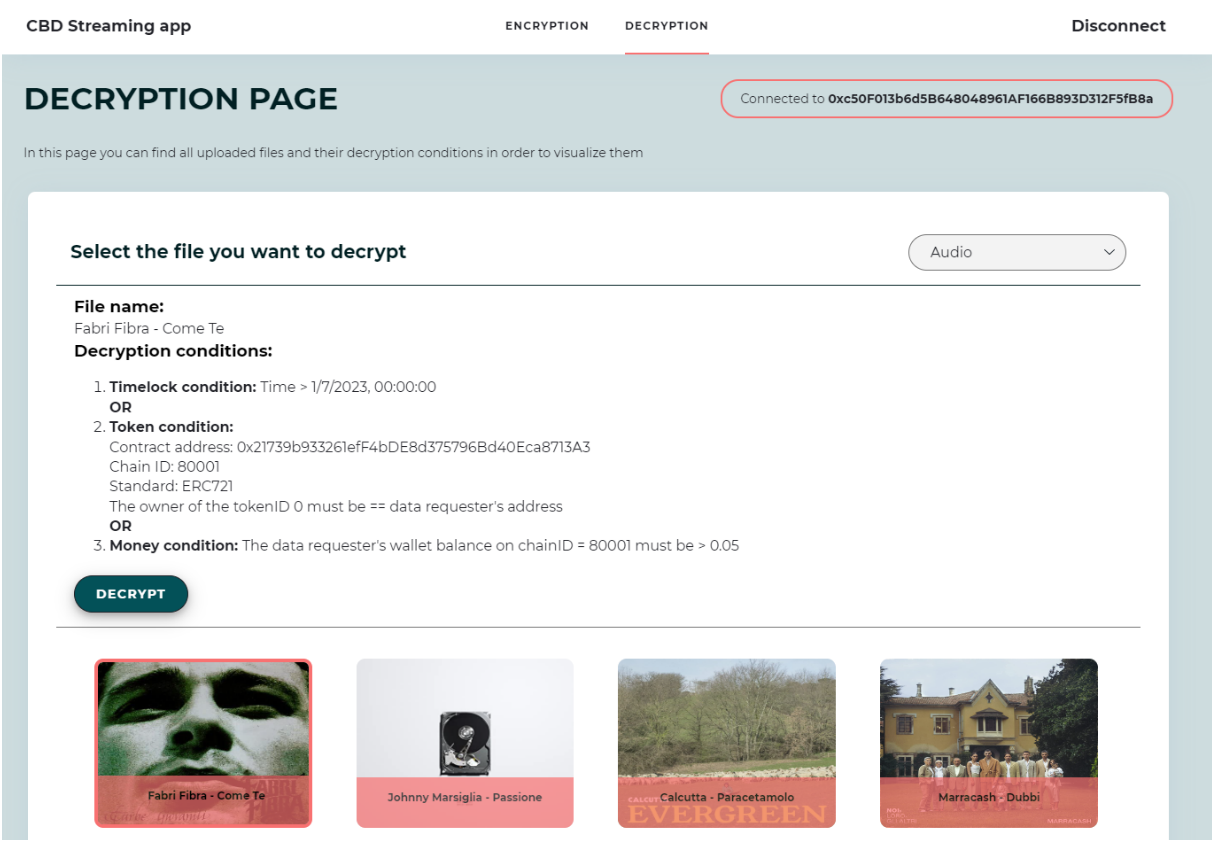
In this page you can find all uploaded files and their decryption conditions in order to visualize them

Select the file you want to decrypt

Figure 3.15: Decryption page showing audio files.

It can be noticed from the above figure that on the decryption homepage the file that was encrypted during the explanation of the first side of the platform in the previous section is also shown. The user, when browsing these files, can look at each file decryption details by clicking onto the relative cover image in order to understand the necessary requirements to gain access to the selected file, the section devoted to this purpose appears above the files' content block.

The figure 3.16 shows the just highlighted section mentioning the decryption details related to the previously encrypted file which presents all three types of decryption conditions.


CBD Streaming app

DECRYPTION PAGE

Connected to **0xc50F013b6d5B648048961AF166B893D312F5fB8a**

In this page you can find all uploaded files and their decryption conditions in order to visualize them

Select the file you want to decrypt

File name:
Fabri Fibra - Come Te
Decryption conditions:

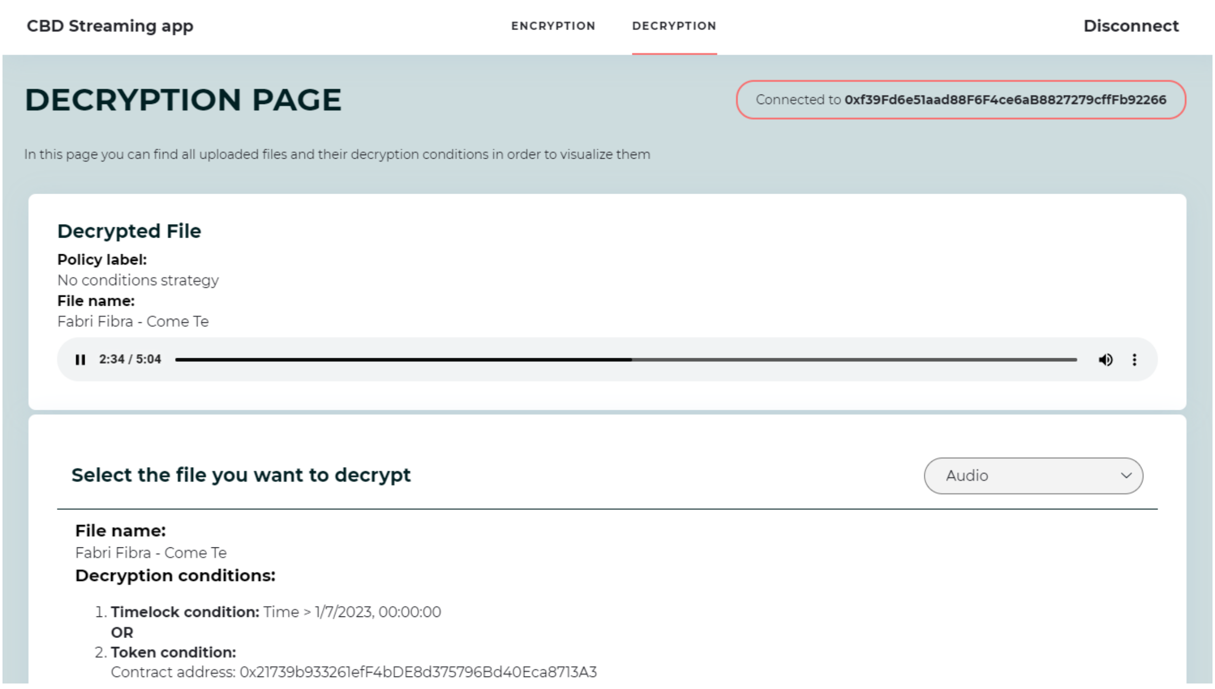
1. **Timelock condition:** Time > 1/7/2023, 00:00:00
   OR
2. **Token condition:**
   Contract address: 0x21739b933261efF4bDE8d375796Bd40Eca8713A3
   Chain ID: 80001
   Standard: ERC721
   The owner of the tokenID 0 must be == data requester's address
   OR
3. **Money condition:** The data requester's wallet balance on chainID = 80001 must be > 0.05

Figure 3.16: File's decryption conditions details.

3.3.2. Files decryption and visualization

When a file's decryption conditions details are shown, that section also present a button "Decrypt" that is useful to send the decryption request to the cohort of nodes established during the related strategy deployment process. The Threshold Network's nodes to interview in order to receive the decryption key fragments are communicated to the data requester through Pinata strategy metadata. When this request takes place they are sent to the cohort nodes the encrypted message encapsulated with the relative condition set and also the currently connected wallet so that the nodes can perform the conditions checking process. In this conditions fulfillment checking phase an interactive step is required: in order to prove the wallet ownership a message signature is required, this happens only the first time the wallet performs a request because this signature is then cached. When this wallet ownership proof correctly ends, the Threshold Network nodes belonging to the cohort proceed by extracting the condition set from the message encapsulation and, subsequently, verifying if that condition set is fulfilled by the provided wallet. If so, each of them returns its decryption fragment to the user, who will be able to decrypt the message only when a number of fragments equal to the strategy threshold value will be collected because only in that case the decryption key can be fully reconstructed. In this case the audio or video file is shown to the data requester without any issue. If, on the other hand, the user's wallet does not meet the decryption requirements, each node returns an error message that is simply shown to the user on the platform.

The below figure 3.17 shows the result of the decryption process of the file shown in figure 3.16 performed with IrpiniaNFT proprietary wallet, that is able to gain access to it because, as said, this file can be freely decrypted by anyone after the 1st of July 2023 at 00:00, but until then the decryption is reserved to the IrpiniaNFT owner and to wallets that own at least 0.05 MATIC on Mumbai Testnet. When the file is provided, they are also displayed the file name and the related deployed strategy label.



DECRYPTION PAGE

Connected to **0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266**

In this page you can find all uploaded files and their decryption conditions in order to visualize them

Decrypted File

Policy label:
No conditions strategy
File name:
Fabri Fibra - Come Te

2:34 / 5:04

Select the file you want to decrypt

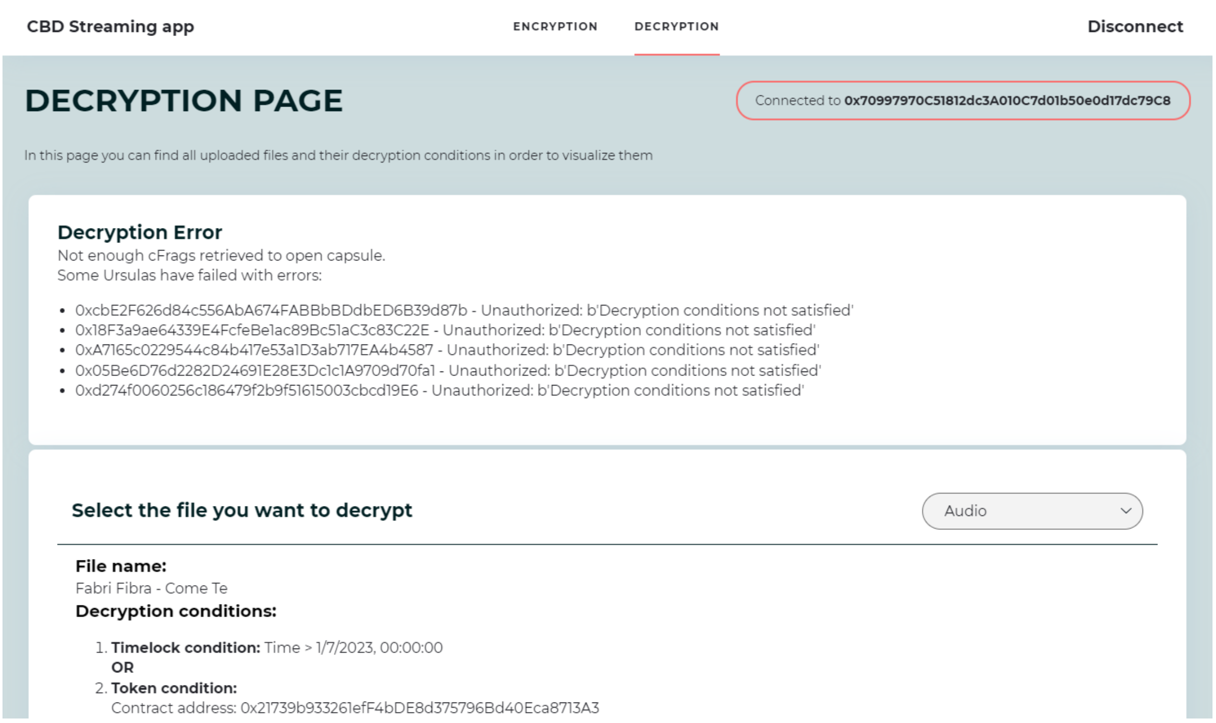
Audio

File name:
Fabri Fibra - Come Te
Decryption conditions:

1. **Timelock condition:** Time > 1/7/2023, 00:00:00
   OR
2. **Token condition:**
   Contract address: 0x21739b933261efF4bDE8d375796Bd40Eca8713A3

Figure 3.17: Successful audio file decryption.

The figure 3.18 instead illustrate the result of the same operation but performed with an empty wallet, which owns neither the IrpiniaNFT nor any MATIC token, therefore a wallet like this will be able to access the file only after the 1st of July 2023. As can be seen they are visualized the addresses of the cohort nodes involved in the strategy with the error message returned by all of them.


CBD Streaming app ENCRYPTION DECRYPTION Disconnect

DECRYPTION PAGE

Connected to **0x70997970C51812dc3A010C7d01b50e0d17dc79C8**

In this page you can find all uploaded files and their decryption conditions in order to visualize them

Decryption Error
Not enough cFrags retrieved to open capsule.
Some Ursulas have failed with errors:

- 0xcbE2F626d84c556AbA674FABBbBDdbED6B39d87b - Unauthorized: b'Decryption conditions not satisfied'
- 0x18F3a9ae64339E4FcfeBe1ac89Bc51aC3c83C22E - Unauthorized: b'Decryption conditions not satisfied'
- 0xA7165c0229544c84b417e53a1D3ab717EA4b4587 - Unauthorized: b'Decryption conditions not satisfied'
- 0x05Be6D76d2282D24691E28E3Dc1c1A9709d70fa1 - Unauthorized: b'Decryption conditions not satisfied'
- 0xd274f0060256c186479f2b9f51615003cbcd19E6 - Unauthorized: b'Decryption conditions not satisfied'

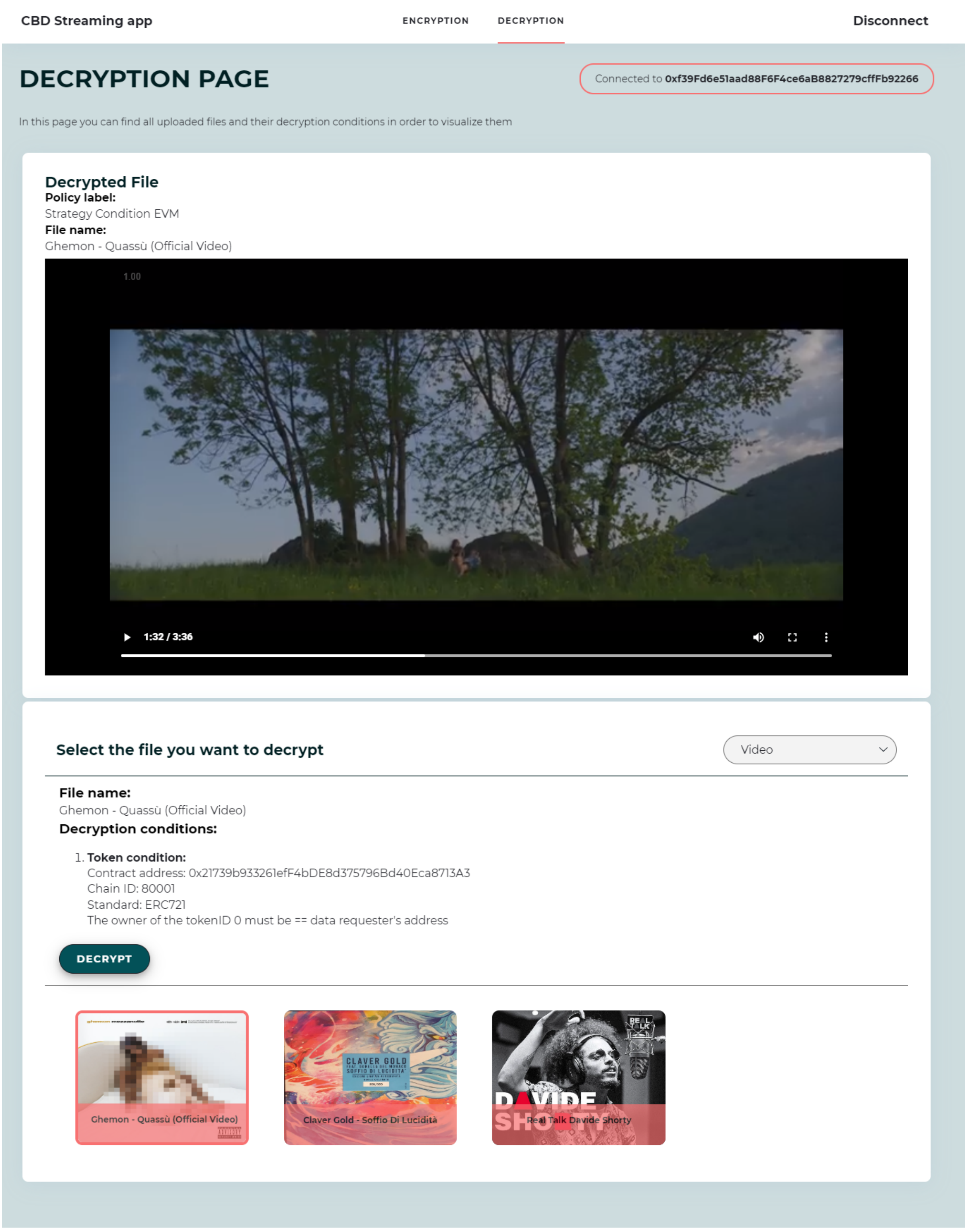
Select the file you want to decrypt Audio

File name:
Fabri Fibra - Come Te
Decryption conditions:

1. **Timelock condition:** Time > 1/7/2023, 00:00:00
   OR
2. **Token condition:**
   Contract address: 0x21739b933261efF4bDE8d375796Bd40Eca8713A3

Figure 3.18: Result of decryption attempt by a non-authorized user.

For the sake of completeness, the figure 3.19 illustrates an example of access to a video file, whose encryption and decryption workflows are completely identical to the audio files ones presented during all this chapter. The access to the below video, as can be read in the decryption condition section in the figure, is reserved to the IrpiniaNFT owner wallet.



DECRYPTION PAGE

Connected to 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

In this page you can find all uploaded files and their decryption conditions in order to visualize them

Decrypted File
Policy label:
Strategy Condition EVM
File name:
Ghemon - Quassù (Official Video)

Select the file you want to decrypt

File name:
Ghemon - Quassù (Official Video)
Decryption conditions:

1. **Token condition:**
   Contract address: 0x21739b933261efF4bDE8d375796Bd40Eca8713A3
   Chain ID: 80001
   Standard: ERC721
   The owner of the tokenID 0 must be == data requester's address

DECRYPT
Ghemon - Quassù (Official Video)
Claver Gold - Soffio Di Lucidità
Real Talk Davide Shorty

Figure 3.19: Successful video file decryption.

4 | Application testing

When developing software, whatever kind it is, a very important phase is testing, which is a mandatory process that must be carried out to minimize the presence of any kind of bugs. The huge importance of this developing step resides in the fact that an eventual presence of bugs in a software launched on the market can lead to several risks for the users and for the company itself. Moreover, the testing phase allows software developers to improve their software quality, through various testing techniques they can assess the functionality, performance, security, and reliability of the software, in order to find any weaknesses and improve them to offer to the end-user an always higher-quality product and reducing the risk of problems for the company's business operations. In particular, in the decentralized applications' world the distribution of a platform which contains bugs is even more dangerous because, as outlined in the chapter 2, this type of software, once written onto the blockchain, is impossible to remove or to make changes to the code, this due to the blockchain immutablility property.

In order to perform software testing several techniques and approaches exists but they are typically divided in two main categories:

- Static testing techniques: this group of testing methods consists of searching for defects in the application without executing the code. This approach is used in the early stages of the software development in order to find issues that may cause failures and, therefore, improve the software efficiency by fixing them.[11]
- Dynamic testing techniques: the methods belonging to this second group, instead, test the software under analysis by inspecting its dynamic behaviour through code execution. The testing operation is performed by inserting dynamic input values that, basing on the software requirements, can be allowed (positive testing) or rejected by the platform (negative testing).[11]

The developed streaming platform has undergone numerous tests, both static and dynamic. The purpose of this chapter is to illustrate the used testing techniques and analyze the results obtained.

4.1. Static tests

For what static testing concerns, the *Static code review* technique has been exploited, this method consists of a systematic review of the source code without obviously running it, since it belongs to the static approaches family. The purpose of the static code review of the developed streaming platform was to, through a careful line-by-line review, look for code imperfections that may cause problems. In particular the attention was focused on:

- Code syntax in order to avoid logical errors with respect to the software requirements.
- Design and implementation choices in order to maximize the platform efficiency.
- Compliance with coding standards and best practices like formatting, naming conventions, code structures.
- Code redundancies search, like unused variables, and comments completeness in order to make the code as understandable as possible by third parties.
- Security vulnerability detection in order to, if any, mitigate any risk derived by them by properly implementing the needed security measures.
- Code complexity and maintainability: it is possible to improve the software maintainability and facilitate future platform enhancement by analyzing factors such as code structure, functions size and code duplication.

During the development of the platform the *Static code review* technique was used whenever a task was accomplished to streamline this process and apply it to smaller code blocks so that nothing was left out.

The test carried out returned quite good results but, at the same time, helped a lot in correcting errors and inefficiencies in the platform's source code. In particular, thanks to this method the maintainability level of the software was improved and some redundancies were found like unused variables. Once the dApp developing was completed another code review was done in order to double check the final obtained product, this final test was passed with very good results, no issues were found also thanks to the several intermediate static test done during the developing phase.

4.2. Dynamic tests

When talking about testing approaches, the most common and used are the ones belonging to the Dynamic testing techniques family. As said, the most relevant characteristic of this group's methods is that, differently from static-based ones, the testing process is performed by running the source code of the software and analyzing its behavior in order to search for defects of various nature during runtime, such as functionality and performance issues or the software compliance with its specific requirements.

Among the techniques that belong to the dynamic ones they can be found *Unit testing* and *Integration testing*, that are two of the most common and important testing methods in literature and are the ones used to dynamically test the developed streaming decentralized platform. The first one involves individual components testing by isolating that specific unit of the software with respect to the others, the aim of the *Unit testing* method is to make sure that each unit and component of the software properly works as expected and it can be done manually or by taking advantage of automated tools. The second one is consequential to the first because, when the unit testing is successfully completed, the *Integration testing* method aims to put that units together and verify the interactions with each other.

4.2.1. Unit testing

Unit testing is an integral part of the developing process of the platform because every developer, once completed the task on which he's working on, runs the code in order to verify the correct operation of that specific code unit. This method allows developers to promote code quality, early detection of eventual bugs and defects and maintainability. This method involves isolating the section of code under testing and check that it properly works by entering dynamic input values, also to verify its robustness with values that could cause malfunctioning, this approach is called negative testing.[17]

In order to test the different units of the streaming platform unit testing processes have been performed also at the end of the work, before putting all together and moving to the integration testing. The purpose of this chapter is to highlight all the tests carried out with the expected behavior and the relative test result. The below table 4.2 shows all the performed tests and the relative results.

Encryption Page testing

Table 4.1: Unit testing table for encryption page components

	Test case	Expected output	Result
1.	**Wallet connection**		
	The user clicks on the "Connect wallet" hypertext	The MetaMask connection popup opens	OK
	The user enters the correct wallet password	The page content with the currently connected wallet address is displayed	OK
	The user enters an incorrect wallet password	The MetaMask popup shows the message "Incorrect password"	OK
2.	**User's encrypted files collection**		
	The user clicks on the "Show" button in the first content block	If the user's key is not cached, the MetaMask signature popup opens	OK
	The user correctly sign the message and no files were encrypted by him	The message "Sorry, zero files found!" is displayed	OK
	The user correctly sign the message and some file were encrypted by him	An unordered list with all file titles is displayed	OK
	The user clicks on an hypertext "See more" in order to visualize file details	Correct URI, strategy label and decryption conditions are displayed below the file title	OK
3.	**Strategy deployment**		
	The user clicks on the "Deploy a new strategy" button	The form to enter the cohort and strategy settings is displayed	OK
	The user correctly fills the form and clicks the "Create" button owning funds on the connected wallet	The MetaMask popup to pay the strategy fee opens and, after that, the newly deployed strategy is stored on Pinata and the conditions content block is displayed	OK

	Test case	Expected output	Result
	The user correctly fills the form and clicks the "Create" button without owning funds on the connected wallet	The error message "Error during strategy creation, check the parameters or your wallet funds" is displayed	OK
4.	**Strategy selection from IPFS**		
	The user clicks on the "Use a deployed strategy" button and no strategies were deployed by him	The message "Sorry, zero strategies found!" is displayed	OK
	The user clicks on the "Use a deployed strategy" button and some strategy were deployed by him	The strategy deployed by the currently connected wallet stored on IPFS are displayed in the form of buttons	OK
	The user clicks on a strategy button	The clicked strategy and relative cohort details are shown	OK
	The user clicks on the button "Use" related to the selected deployed strategy	The selection content block is disabled and the file encryption condition content block appears	OK
5.	**Decryption conditions insertion**		
	The user checks the checkbox in order to override the strategy's decryption condition for that specific file	The content block that allows the user to define a condition set appears	OK
	The user correctly fills the single condition form and clicks the button "Add new"	The inserted condition appears in the JSON preview below	OK
	The user fills the single condition form with illegal values and clicks the button "Add new"	The error message "Sorry, error while adding your condition, check your parameters and retry!" appears and the condition set must be rebuilt	OK

	Test case	**Expected output**	**Result**
	The user clicks the button "Reset"	The condition set represented in the JSON preview empties out in order to be rebuilt	OK
	The user inserts at least one condition and clicks the button "Complete"	The condition content block's button become disabled and the encryption content block appears.	OK
6.	**File encryption**		
	The user correctly fills the encryption form	The button "Encrypt" in enabled	OK
	Once filled the form, the user clicks on the button "Encrypt"	The loader is displayed during encryption and uploading operations and, when the process correctly completes, the encrypted file is stored on Pinata and the relative URI is displayed	OK

Decryption Page testing

Table 4.2: Unit testing table for decryption page components

	Test case	Expected output	Result
1.	**Wallet connection**		
	The user clicks on the "Connect wallet" hypertext	The MetaMask connection popup opens	OK
	The user enters the correct wallet password	The page content with the currently connected wallet address is displayed	OK
	The user enters an incorrect wallet password	The MetaMask popup shows the message "Incorrect password"	OK
2.	**Encrypted files downloading**		
	The connected user enters the page	A row made up of four images representing the audio files and a button "Show more" appear	OK
	The user clicks the button "Show more"	Another row of four images appears below	OK
	The user changes the dropdown menu value to "Video"	The previously visualized images are changed with the ones representing the video files	OK
	The user changes the dropdown menu value to "All files"	Both audio and video files related images are displayed in the content	OK
3.	**Files decryption details visualization**		
	The user clicks on an image related to a file	A content block showing the clicked file name and the related decryption conditions to fulfill is displayed	OK

	Test case	Expected output	Result
	The user clicks on another file image	The content of the details section changes, showing the information related to the newly selected file	OK
3.	**File decryption process**		
	The user clicks on the button "Decrypt" related to a file and his connected wallet fulfills the required decryption conditions	The loader is shown during the conditions checking and decryption process and, at the end, the audio or the video file is shown to the user	OK
	The user clicks on the button "Decrypt" related to a file and his connected wallet doesn't fulfill the required decryption conditions	The loader is shown during the conditions checking and decryption process and, at the end, the Threshold Network nodes' error responses are shown: "Unauthorized: b'Decryption conditions not satisfied'"	OK

4.2.2. Integration testing

As said, *Integration testing* is commonly intended as the second phase of a software testing process, consequent to *Unit testing*, because this method in needed to check that the single software units analyzed by unit testing properly work together as designed.[23] The integration testing process is usually performed when all the pieces are merged and it is very useful to verify the compatibility of software modules and components, the correct data flow between different part of the software and many others possible defects that can come out during the components integration.

For the streaming platform integration testing has been performed both during the development process and at the end of it, this because the platform works in a sequential way: each unit work is fundamental to the subsequent ones. In literature many types of integration testing exists, but the focus was directed to the *Big Bang Integration* method, that consists of merging all the modules and components and testing the entire system as a whole. This approach is mainly used for relatively small and simple softwares.[23]

The used testing techniques are only the first two levels of the pyramid of software testing methods that is made up of four floors. The last two methods of this pyramidal hierarchy are designed to be used in the very last developing phases before the product launching. They are, in sequence:

- System testing: in this phase it is checked the system compliance with the requirements in terms of security, performances, speed, reliability and many other parameters. Usually, this test is performed by professional figures trying to emulate the real end-users environment as much as possible in order to search for eventual issues. This step is very important because, as said, it is performed when the product launch to the market is about to happen.[15]
- Acceptance testing: also known as *UAT (User Acceptance Testing)* it the very top of the highlighted pyramid and consists of a distribution of a Beta version of the software to some users in order to allow them to use it, as a simulation of the product launch, in order to receive the last feedbacks before the final release. Usually, the tester's focus is mainly directed to the usability feature of the software, but also to spelling mistakes, clarity of usage and more.[15]

The *System testing* and *Acceptance testing* will be the future testing phases of the decentralized streaming platform. Since they are the last two steps, they will be performed once eventual future platform enhancements will be carried out.

5 | Conclusions and future developments

The preliminary studies that anticipated the decentralized platform developing were made to find the optimal solution for the usability-privacy trade off that everyone meet everyday in the nowadays digital world, as explained in the Chapter 1. The main idea, which came about through collaboration with the company KNOBS, was to find the best way to share files between users without worrying about security issues and taking advantage of decentralized technologies. After several protocols and solutions analysis the choice fell on *Conditions-Based Decryption (CBD)* which has been widely explained and analyzed the previous chapters and allows the achievement of an high usability level while ensuring the encrypted data privacy.

The streaming platform allows users to encrypt audio and video files in order to make their decryption available only to users whose cryptocurrency wallet fulfills certain conditions, from the security point of view the solution adopted isn't obviously too restrictive since the conditions may allow also every users to decrypt the file, but that is a user choice: the user when encrypt his files establishes the decryption conditions basing on the secrecy level of the file. So, the platform offers the possibility of making the decryption available to anyone, for example by inserting a simple timelock condition, or also to one and only user, for example by setting a Non-Fungible Token ownership condition. It is important, however, to underline that when the file runs encrypted onto the web it is extremely secure, practically undecryptable. This is why this technology was chosen, because is flexible to different situations while ensuring a very high data security level.

For what future developments and platform enhancements concern, a smarter decryption process could be implemented: the current platform version handles the encryption and decryption flows by manipulating the entire file, this means that, in case of huge files, the decryption operation could last a long time. A solution to ease this process might be to manage the encryption process by fragmenting the files and encrypting the resulting fragments in order to allow the data requester to access the file without waiting for the file to be fully decrypted, and giving the requester the ability to start accessing the file during download, as classic streaming platforms do.

Another limit of the Web3-Streaming platform is that once a message is encrypted and uploaded on IPFS, its condition set is unmodifiable, since it is written ensemble with the ciphertext on IPFS, that is unchangable by definition. An improvement of the platform under this point of view could be achieved by giving the possibility to encrypt messages and relate their decryption to updateable condition set. This solution can be reached by deploying onto the blockchain a smart contract containing the decryption condition set and giving the data owner the possibility to alter it by sending a transaction to the smart contract. This solution could also help the data owner to revoke access to the file for all users.

Bibliography

[1] What is openzeppelin? developer's guide 2023. URL https://www.alchemy.com/overviews/openzeppelin-developers-guide-2022#:~:text=OpenZeppelin%20is%20a%20crypto%20cybersecurity,largest%20DeFi%20and%20NFT%20projects.

[2] What is a blockchain wallet and how does it work? URL https://kriptomat.io/blockchain/what-is-a-blockchain-wallet/.

[3] Character concepts, 2019. URL https://docs.nucypher.com/en/latest/architecture/character.html.

[4] Ethereum contracts, 2019. URL https://docs.nucypher.com/en/latest/architecture/contracts.html.

[5] Web development, 2019. URL https://docs.nucypher.com/en/latest/application_development/web_development.html.

[6] Threshold cryptography, mpc, and multisigs: A complete overview, 2022. URL https://blog.pantherprotocol.io/threshold-cryptography-an-overview/.

[7] About threshold network, 2022. URL https://threshold.network/about/.

[8] What is threshold network (t)?, 2022. URL https://academy.binance.com/en/articles/what-is-threshold-network-t.

[9] Conditions-based decryption (cbd), 2023. URL https://docs.threshold.network/applications/threshold-access-control/conditions-based-decryption-cbd.

[10] Consensus mechanisms, 2023. URL https://ethereum.org/en/developers/docs/consensus-mechanisms/.

[11] Software testing techniques, 2023. URL https://www.geeksforgeeks.org/software-testing-techniques/.

[12] Threshold access control, 2023. URL https://docs.threshold.network/applications/threshold-access-control.

[13] Types of blockchain, 2023. URL https://www.geeksforgeeks.org/types-of-blockchain/.

[14] C. Boscolo. Che cos'è e come funziona threshold network?, 2023. URL https://it.cryptonews.com/news/che-cose-e-come-funziona-threshold-network.htm.

[15] M. Calvello. The 4 levels of testing in software engineering explained: Examples, challenges, and approaches, 2022. URL https://fellow.app/blog/engineering/the-levels-of-testing-in-software-engineering-explained/.

[16] T. Candido. A technical introduction to blockchain, 2020. URL https://betterprogramming.pub/a-technical-introduction-to-blockchain-22ab05308151.

[17] T. Hamilton. Unit testing tutorial – what is, types test example, 2023. URL https://www.guru99.com/unit-testing-guide.html.

[18] N. Iny. Sensitive data exposure: What is it and how to avoid it?, 2022. URL https://www.polar.security/post/sensitive-data-exposure.

[19] M. L. Jake Frankenfield, Jefreda R. Brown. Decentralized applications (dapps): Definition, uses, pros and cons, 2023. URL https://www.investopedia.com/terms/d/decentralized-applications-dapps.asp.

[20] A. Knezovic. The difference between fungible and non-fungible tokens (nfts), 2022. URL https://medium.com/udonis/the-difference-between-fungible-and-non-fungible-tokens-nfts-123df237b892#:~:text=Fungible%20tokens%20are%20not%20unique,investments%20in%20the%20right%20hands.

[21] D. Nuñez. Umbral: A threshold proxy re-encryption scheme, 2018. URL https://github.com/nucypher/umbral-doc/blob/master/umbral-doc.pdf.

[22] R. N. Shilpa Lama. What is ipfs?, 2022. URL https://beincrypto.com/learn/what-is-ipfs/.

[23] J. Terra. What is integration testing: Examples, challenges, and approaches, 2023. URL https://www.simplilearn.com/what-is-integration-testing-examples-challenges-approaches-article.

[24] D. Zafar. 8 blockchain security issues you are likely to encounter,

2022. URL https://cybersecurity.att.com/blogs/security-essentials/8-blockchain-security-issues-you-are-likely-to-encounter.

List of Figures

List of Tables

Acknowledgements

I want to thank my advisor, the professor Francesco Bruschi, for giving me the opportunity of working on this thesis that I've been so passionate about in this months of hard work, for giving me the chance to expand my knowledge by delving into the blockchain world and for supporting me during the study and development process together with my co-advisor, Manuel Tumiati, and Stefano De Cillis from KNOBS whom I want to thank for all the advices and insights given to me in these last months.