Republic of Tunisia
Ministry of Higher Education and Scientific Research
University of Tunis El Manar
Faculty of Sciences of Tunis

Computer Science Department

Project End of Year

Proposal for an evolution of the Oauth 2.0 protocol based on blockchain and IPFS

Realized by

Ben Salah Nabil

Classe : Second Year Computer Science Engineering

Framed by

Dr. KAFFEL BEN AYED Hella

Ms. KHOUALDI Jihene

Presented in 02 Jun 2022

jury :

membre : Dr ESSID Chaker

College year 2021/2022

Thanks

In a general way, I would like to thank all those who have helped me, from near and far, during the elaboration of this project. Special Thank to Dr. KAFFEL BEN AYED Hella and Ms. KHOUALDI Jihene for helping to go forward in this work and enhance my knowledge of different aspects.

Contents

List of Figures

General Introduction

With the rapid development of smart devices and broadband networks, the Internet of objects (IoT) has gradually moved closer to our lives. It is necessary to remotely control a large number of IoT devices to perform required functions and share information. However, the security and privacy issues will result in economic losses for users and will hinder the development of the Internet of Things. Therefore, access control is considered as one of the most important techniques to ensure the security and privacy of the Internet of Things.

The design of the access control system should adapt to the global trend of the Internet of Things to help users achieve better security and operations simplified. In particular, the access control system in the Internet of Things is designed with the following elements in mind: interoperability, integration of physical and logical access and experience user. Oauth2.0 is one of the access control protocols used in the IoT field. Oauth2.0 is above all a protocol that authorizes the distribution of access tokens in order to acquire authorizations.

The main purpose of an access token is to replace the traditional username and password, and to include different authorizations understandable by the resource server. It's a standard which is now adopted by the world leaders of the web (Google, Facebook, Amazon, Twitter, etc.) and which has proven itself. The set of specifications suggested by the "framework" offers the necessary tools for managing and reassuring communications in a network of resources protected (our IoT in this case)

Chapter 1

Study of the existing

1.1 Introduction

Today healthcare industries are maintaining COVID-19 patients' information electronically which includes patients' diagnostic reports, patients' private information, and doctor prescriptions. However, the COVID-19, patient sensitive information is currently stored in centralized or third-party storage model.

One of the key challenge of centralized storage model is the preserving privacy of patient information and transparency in the system. The privacy risk include illegitimate access to sensitive information of patient such as identification details access and misutilization of patient information and their clinical records.

To overcome this challenge, we proposed a distributed on chain and off-chain storage model using consortium blockchain and interplanetary file systems (IPFS). The proposed framework though maintaining patient privacy makes it easier for legitimate entities like healthcare providers (e.g., physicians and clinical staffs) to access clinical data of COVID-19 patients'.

Terms—Blockchain, off-chain storage, Interplanetary File Systems (IPFS), Local Mining, peer-to-peer (P2P) storage and sharing model.

1.2 Related work

This work is partly based on the result of KHOUALDI Jihene's research project "Proposal for an evolution of the Oauth protocol according to the SSI model and blockchain-based" presented in February 2022. To make data sharing be more efficiently, SSI architecture is proposed.

1.3 Blockchain

1.3.1 Definition

A blockchain is defined as a distributed database of records, or ledger public, of all transactions or digital events that have been executed and shared between participating parties. Every public ledger transaction is verified by majority consensus participants in the system. Once entered, information can never be erased.

Blockchain contains a certain and verifiable record of every single transaction ever made. The three main keywords are Transaction, Block and Consensus. A blockchain is a database of transactions, these transactions are grouped into blocks, whether or not the newly accepted block is generated to be added to the blockchain or not is evaluated using a consensus algorithm.

1.3.2 Blockchain blocks

A transaction is the smallest unit of a block. A transaction can be a financial transaction (in the case of cryptocurrencies) or simply describe an event such as the publication of data in a stream in the case of multichain.

The content of a transaction differs from a blockchain platform to each other, but usually they share a set of attributes such as its id and timestamp. For a transaction to be accepted by the blockchain network, it must be digitally signed in using the privacy of the issuer of this transaction.

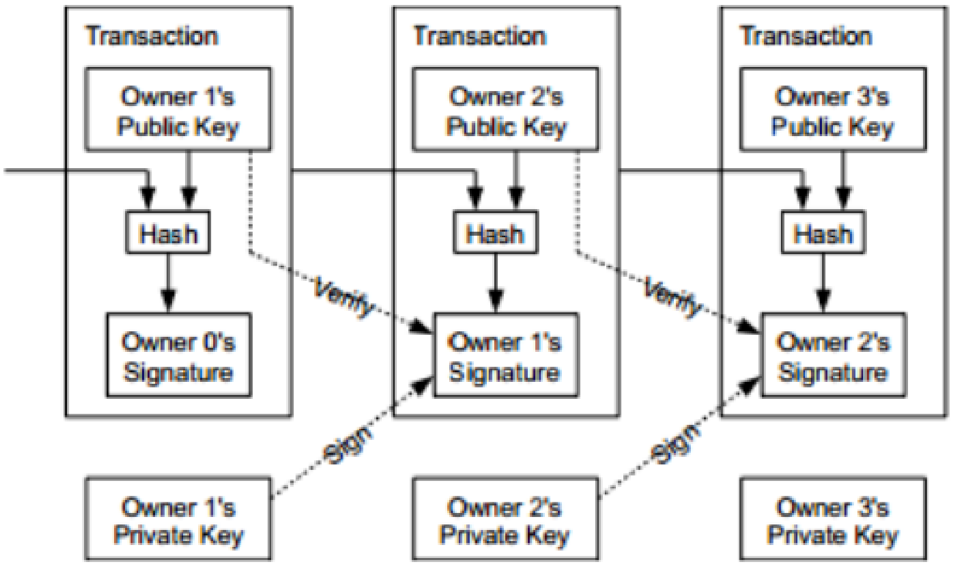


Figure 1.1: Blockchain blocks

1.3.3 Blockchain Transaction

After a blockchain transaction is broadcast on the network, a set of blockchain machines blockchain (nodes) called miners compete to create a block that satisfies the requirements of the transaction. Miners compete to create a block that satisfies the consensus algorithm. As shown in Figure 1.2, a block contains 3 main fields, the Prev Hash field contains the hash of the previous block.

The block also contains a set of Tx transactions and finally, it contains the Nonce which is a series of bytes used to modify the hash value of a block. In the case of Bitcoin, the mining process is in fact an operation which consists in modifying the Nonce and calculate the new block. The Nonce and the calculation of the new hash of the block, until this hash meets a certain condition.

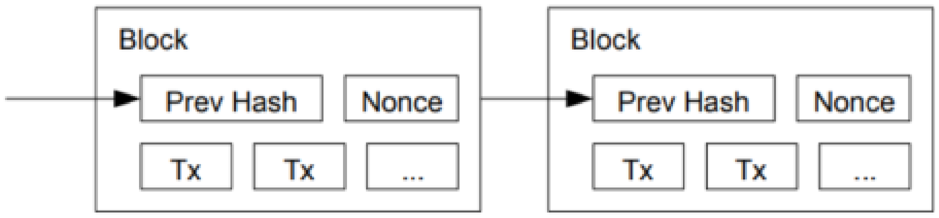


Figure 1.2: Different fields of a block

1.3.4 blockchain consensus

Blockchain consensus is the process by which a blockchain transaction is approved and a new block is created. Blockchain on what is considered a valid chain.

This verification method may be corrupted if more than half of the nodes in your blockchain are controlled by a single party.

There are several consensus algorithms, the most common being The following :

Proof of Work :

Proof of Work consensus is used in Bitcoin and Ethereum platforms. This process requires that the nodes of the blockchain compete with each other to answer an equation mathematical. In the case of bitcoin, nodes must create a block whose hash must include a fixed minimum number of zero. To do this, miners continually create blocks by grouping a set of transactions and generating a Nonce, they then calculate its hash. This operation can be occur up to thousands of times per second. For this reason, this consensus algorithm is very resource consumer, and the amount of resources consumed increases as the number number of blockchain nodes is increasing .

Proof of Stake :

Unlike consensus by “Proof of Work“, consensus by Proof of Stake does not require the same massive computing power, in fact, it requires a lot less. Instead of putting computers in competition, this protocol chooses nodes randomly or based on their holdings, size or their duration of activity, depending on the platform. Many platforms adopt this consensus, such as Peer Coin, ShadowCash and BlackCoin .

1.4 The OAuth2.0 Framework

OAuth is an authorization framework in which a third-party application is delegated limited right to access the user information that is stored in another web service. OAuth 2.0 provides more development convenience and has a simpler authentication process than OAuth 1.0 and OAuth 1.1 . It does this by removing a complicated encrypting process and using the HTTPS protocol.

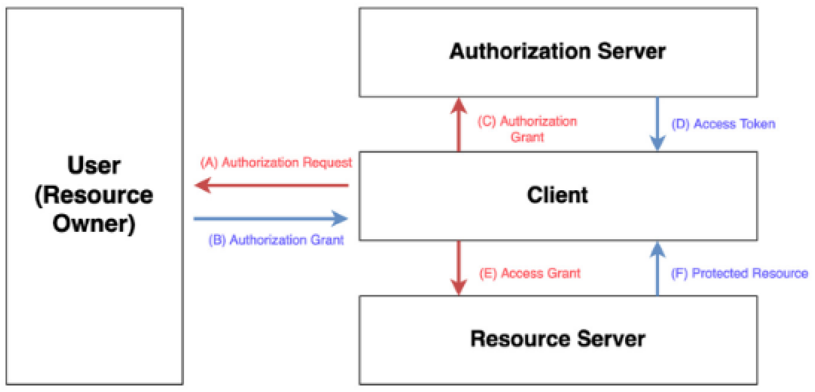


Figure 1.3: The abstract flow of OAuth 2.0

1.4.1 User (resource owner)

An entity capable of authorizing access to a protected resource. When the resource owner is a person, it is called an user. Resource owners are central elements in the OAuth 2.0 concept. Usually resource owner are users who play a certain role in the respective business scenario, for example salespersons. They make resources (see Resource) available for other users by delegating their scopes to OAuth 2.0 enables client applications. These applications use an OAuth 2.0 client to access the resources on behalf of the resource owners.

Depending on the grant type used, the resource owners can further restrict the number of scopes for certain client during the access token request. They can freely decide which applications can access which business resources by assigning them to the business resources

1.4.2 Client

It's third-party application that wants access to the private resources of the resource owner. The OAuth client can make protected resource requests on behalf of the resource owner after the resource owner grants it authorization. OAuth 2.0 introduces two types of clients: confidential and public. Confidential clients are registered with a client secret, while public clients are not.

1.4.3 Authorization Server (AS)

Known as the Authorization server in OAuth 2.0. The server that gives OAuth clients scoped access to a protected resource on behalf of the resource owner. The server issues an access token to the OAuth client after it successfully does the following actions:

- Authenticates the resource owner.
- Validates a request or an authorization grant.
- Obtains resource owner authorization.

An authorization server can also be the resource server.

1.4.4 Resource Server (RS)

1.4.4.1 Definition

The server that hosts the protected resources. It can use access tokens to accept and respond to protected resource requests. The resource server might be the same server as the authorization server.

The resource server is the OAuth 2.0 term for your API server. The resource server handles authenticated requests after the application has obtained an access token.

Large scale deployments may have more than one resource server. Google's services, for example, have dozens of resource servers, such as the Google Cloud platform, Google Maps, Google Drive, Youtube, Google+, and many others. Each of these resource servers are distinctly separate, but they all share the same authorization server.

1.4.4.2 Verifying Access Tokens

The resource server will be getting requests from applications with an HTTP Authorization header containing an access token. The resource server needs to be able to verify the access token to determine whether to process the request, and find the associated user account, etc.

If you're using self-encoded access tokens, then verifying the tokens can be done entirely in the resource server without interacting with a database or external servers.

If your tokens are stored in a database, then verifying the token is simply a database lookup on the token table.

Another option is to use the Token Introspection spec to build an API to verify access tokens. This is a good way to handle verifying access tokens across a large number of resource servers, since it means you can encapsulate all of the logic of access tokens in a single server, exposing the information via an API to other parts of the system. The token introspection endpoint is intended to be used only internally, so you will want to protect it with some internal authorization, or only enable it on a server within the firewall of the system.

1.4.4.3 Verifying Scope

The resource server needs to know the list of scopes that are associated with the access token. The server is responsible for denying the request if the scopes in the access token do not include the required scope to perform the designated action.

The OAuth 2.0 spec does not define any scopes itself, nor is there a central registry of scopes. The list of scopes is up to the service to decide for itself. See Scope for more information.

1.4.4.4 Expired Tokens

If your service uses short-lived access tokens with long-lived refresh tokens, then you'll need to make sure to return the proper error response when an application makes a request with an expired token.

Return an HTTP 401 response with a WWW-Authenticate header as described below. If your API typically returns JSON responses, then you can also return a JSON body with the same error information.

1.4.5 Limitations of Oauth2.0

OAuth 2.0 allows applications to access other people's data without revealing the user credentials. It uses username and password tokens instead outmoded. Sensitive data such as credit card numbers, medical records, statements banking or passwords are stored remotely and given a token ID so that merchants and third parties cannot access credit card numbers, credit, records medical, passwords, etc., but can verify the transaction tokens.

The centralized architecture of Oauth2.0 is based on the authorization server that is responsible for managing access requests of resource requester, at the same time the management of access token what is entailed the problem of the point of single failure, we start with the explanation of this obstacle. A single point of failure is part of the system and will cause the entire system to shut down in the event of a failure. In short, if one thing fails, then everything will fail.

SPOF is caused by poor design and technique of Implementation. They are unpopular in any system, be it software applications, hardware modules, manufacturing systems or business practices. Figure 3.3 shows the consequences of the presence of SPOF in the architecture of Oauth2.0: in our case if the resource requester requests access and the authorization server will be a single point of failure, the server goes down down resource requesters will not be able to access the requested service. This SPOF can put at any activity related to your authorization server .

1.5 OAuth2.0 Claim based

In the new distributed architecture of OAuth2.0 the resource owner is the generator of new token named Claim. This claim is based on the verifaible SSI model credential which will store in the etherum blockchain (immutable ledger) .

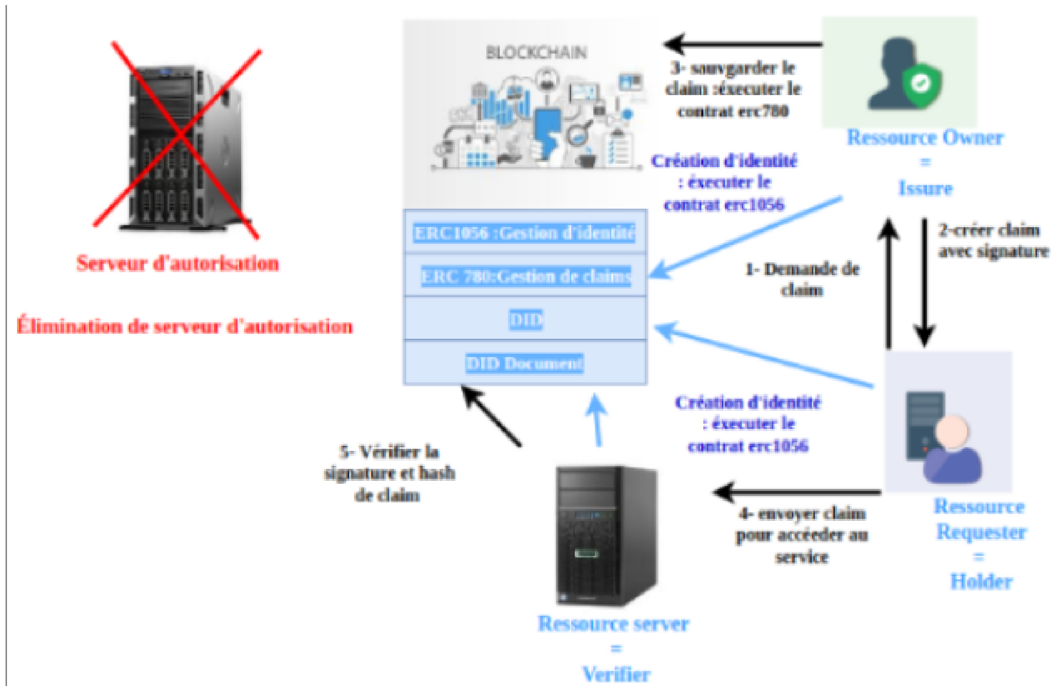

Serveur d'autorisation
Élimination de serveur d'autorisation
BLOCKCHAIN
ERC1056 :Gestion d'identité
ERC 780:Gestion de claims
DID
DID Document
3- sauvgarder le claim :éxecuter le contrat erc780
Ressource Owner = Issure
Création d'identité : éxecuter le contrat erc1056
2-créer claim avec signature
1- Demande de claim
Création d'identité : éxecuter le contrat erc1056
5- Vérifier la signature et hash de claim
4- envoyer claim pour accéder au service
Ressource Requester = Holder
Ressource server = Verifier

Figure 1.4: architecture of OAuth2.0 claim based

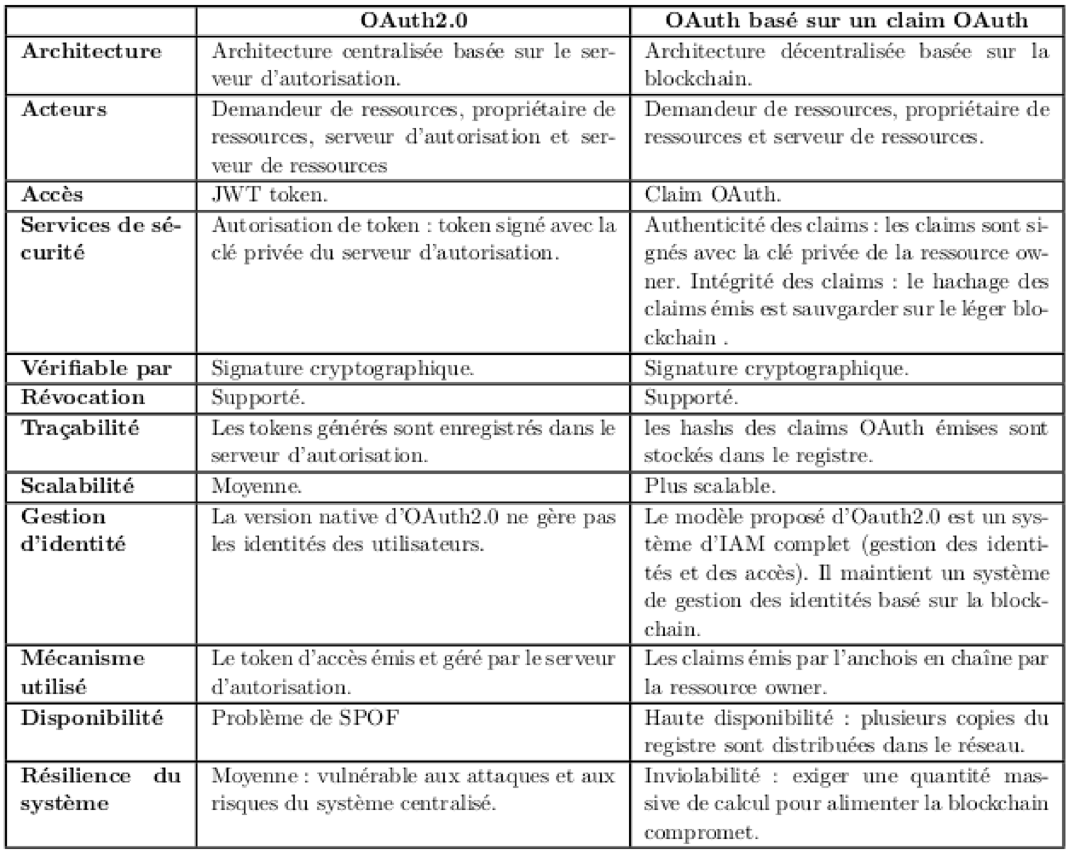
1.5.1 Benefits of Oauth2.0 Improved Architecture

Improve security :

The Blockchain technique combines security with the ability to verify and authorize transactions issued by trusted parties and encryption during data transmission and storage. This technique provides transparency over who has access, who executes transactions and records all interactions.
Additionally, blockchain adds a layer of security in terms of encryption, removal of single points of failure, and the ability to quickly identify weak links on the network.

Lower the cost :

By automating the stages of verification and processing of blockchain transactions, the entire ecosystem can be made proactive at a lower cost.



	OAuth2.0	**OAuth basé sur un claim OAuth**
Architecture	Architecture centralisée basée sur le serveur d'autorisation.	Architecture décentralisée basée sur la blockchain.
Acteurs	Demandeur de ressources, propriétaire de ressources, serveur d'autorisation et serveur de ressources	Demandeur de ressources, propriétaire de ressources et serveur de ressources.
Accès	JWT token.	Claim OAuth.
Services de sécurité	Autorisation de token : token signé avec la clé privée du serveur d'autorisation.	Authenticité des claims : les claims sont signés avec la clé privée de la ressource owner. Intégrité des claims : le hachage des claims émis est sauvgarder sur le léger blockchain .
Vérifiable par	Signature cryptographique.	Signature cryptographique.
Révocation	Supporté.	Supporté.
Traçabilité	Les tokens générés sont enregistrés dans le serveur d'autorisation.	les hashs des claims OAuth émises sont stockés dans le registre.
Scalabilité	Moyenne.	Plus scalable.
Gestion d'identité	La version native d'OAuth2.0 ne gère pas les identités des utilisateurs.	Le modèle proposé d'Oauth2.0 est un système d'IAM complet (gestion des identités et des accès). Il maintient un système de gestion des identités basé sur la blockchain.
Mécanisme utilisé	Le token d'accès émis et géré par le serveur d'autorisation.	Les claims émis par l'anchois en chaîne par la ressource owner.
Disponibilité	Problème de SPOF	Haute disponibilité : plusieurs copies du registre sont distribuées dans le réseau.
Résilience du système	Moyenne : vulnérable aux attaques et aux risques du système centralisé.	Inviolabilité : exiger une quantité massive de calcul pour alimenter la blockchain compromet.

Figure 1.5: Comparison between Oauth2.0 of the old version and Oauth2.0 with the new architecture.

1.5.2 Limitation

Using blockchain to improve OAuth2.0 performance was a very good idea to decentralise data and be more secure , But this have also many drawbacks related essentially to blockchain use such as :

1.5.2.1 Wasteful

Every Node runs the blockchain in order to maintain Consensus across the blockchain. This gives extreme levels of fault tolerance, ensures zero downtime, and makes data stored on the blockchain forever unchangeable and censorship-resistant. But all this is wasteful, as each Node repeats a task to reach Consensus burning electricity and time on the way.

This makes computation far slower and more expensive than on a traditional single computer. There are many initiatives that seek to reduce this cost focusing on alternative means of maintaining Consensus, such as Proof-of-Stake.

1.5.2.2 Complexity

Blockchain technology involves an entirely new vocabulary.
It has made cryptography more mainstream, but the highly specialized industry is no place for a beginner...

Thankfully there are blockchain and cryptocurrency courses and indexes being created for newcomers, but overall this is a very complicated industry that will not be soaked in and applied overnight.

1.5.2.3 Network speed/cost

Blockchain networks require Nodes to run. But as many of the networks are new, they lack the number of Nodes to facilitate widespread usage. This lack of resource manifests as:

• Higher costs — as Nodes seek higher rewards for completing Transactions in a Supply and Demand scenario
• Slower transactions — as Nodes prioritise Transactions with higher rewards, backlogs of transactions build up

Over time, successful public blockchain networks will have to incentivise Nodes, whilst creating favourable costs for users, with transactions completed in a relevant timeframe. This balance is key to the economics of each blockchain.

1.5.2.4 Transaction Costs, Network Speed

Now, as the network continues to grow, we can clearly see that at this rate using Bitcoin will NOT be the most cost effective option of transferring money due to rising transaction costs in the network.

1.5.2.5 The size of the block

Each transaction or “block” added to the chain increases the size of the database. As every node has to maintain a the chain to run, the computing requirements increase with each use. For large public implementations of Blockchain this has one of two affects:

• Smaller ledger — Not every Node can carry a full copy of the Blockchain, potentially affecting immutability, consensus etc.
• More centralised — There is a high barrier to entry to become a Node, encouraging a larger amount of centralisation in the Network, with bigger players able to take more control.

Neither of these scenarios is desirable, without considering the full implications, as it will likely affect the use cases for blockchain variants.

1.5.2.6 Immutable Smart contracts

Once the smart contact is added to the blockchain, it becomes immutable, in that it cannot be changed. If there are flaws in the code that may be exploited by hackers, they are there forever. This is not a concern when a smart contract is not being used, but as smart contacts behave like accounts, they can be used to store large amounts of value.

This can create scenarios where hackers can exploit code flaws to send the contents of smart contracts to their own accounts. As the blockchain is immutable, these Transactions are very hard to undo, meaning large amounts of value may be lost forever.

1.5.3 IPFS Use Case

Blockchain is one of the most exciting use cases for IPFS. A blockchain has a natural DAG structure in that past blocks are always linked by their hash from later ones. More advanced blockchains like the Ethereum blockchain also has an associated state database which has a Merkle-Patricia tree structure that also can be emulated using IPFS objects.

We assume a simplistic model of a blockchain where each block contains the following data:

- A list of transaction objects
- A link to the previous block
- The hash of a state tree/database

This blockchain can then be modeled in IPFS as follows:

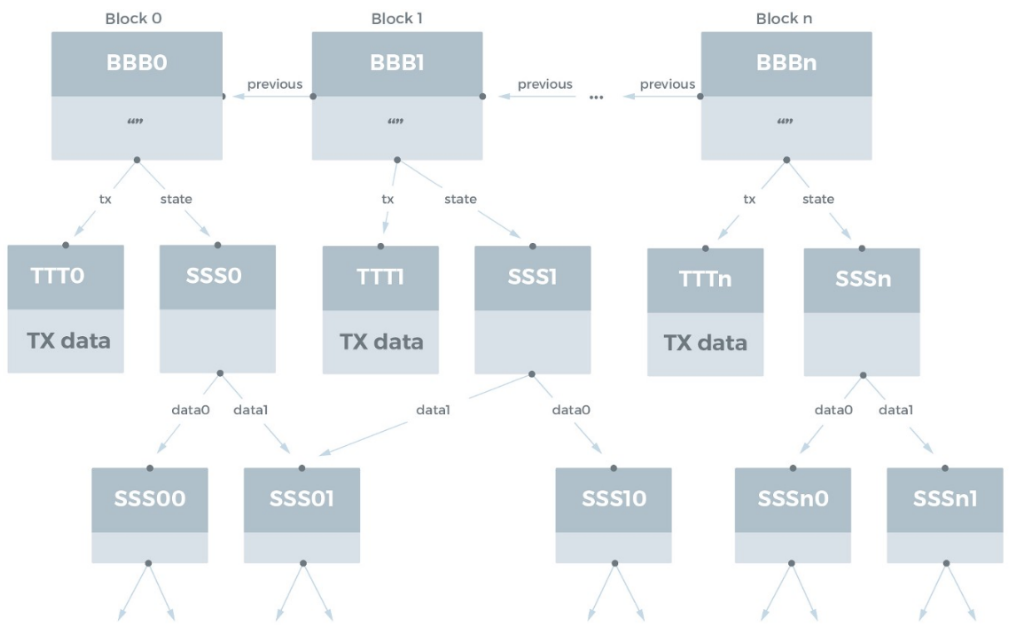


Figure 1.6: Interpolation IPFS-BLOCKCHAIN

We see the deduplication we gain when putting the state database on IPFS — between two blocks only the state entries that have been changed need to be explicitly stored.

An interesting point here is the distinction between storing data on the blockchain and storing hashes of data on the blockchain. On the Ethereum platform you pay a rather large fee for storing data in the associated state database, in order to minimize bloat of the state database (“blockchain bloat”). Thus it’s a common design pattern for larger pieces of data to store not the data itself but an IPFS hash of the data in the state database.

If the blockchain with its associated state database is already represented in IPFS then the distinction between storing a hash on the blockchain and storing the data on the blockchain becomes somewhat blurred, since everything is stored in IPFS anyway, and the hash of the block only needs the hash of the state database. In this case if someone has stored an IPFS link in the blockchain we can seamlessly follow this link to access the data as if the data was stored in the blockchain itself.

We can still make a distinction between on-chain and off-chain data storage however. We do this by looking at what miners need to process when creating a new block. In the current Ethereum network the miners need to process transactions that will update the state database. To do this they need access to the full state database in order to be able to update it wherever it is changed.

Thus in the blockchain state database represented in IPFS we would still need to tag data as being “on-chain” or “off-chain”. The “on-chain” data would be necessary for miners to retain locally in order to mine, and this data would be directly affected by transactions. The “off-chain” data would have to be updated by users and would not need to be touched by miners.

1.5.4 Conclusion

In this chapter we saw a bibliography of the different terms used in this project we talks first about the Blockchain, Then we talk about OAuth2.0 and it’s workflow and how we improved it to OAuth2.0 claim based using blockchain. In the Second part we saw the architecture used in a Outh 2.0 claim based , the fundamentals terms and finally the use case of the IPFS protocol.

Chapter 2

My Project

2.1 Introduction

Last but not least,after explaining the key terms of this study,The work consists in defining an architecture based on an OAuth2.0 claim based and IPFS and in designing a secure (authentication, access authorization), traced, and consented (by the user) access control service to health data.

For more explanation here is two diagrams that describe in a simple way the general conception :

2.2 Identity creation

The procedure for creating an identity is presented in the following figure :

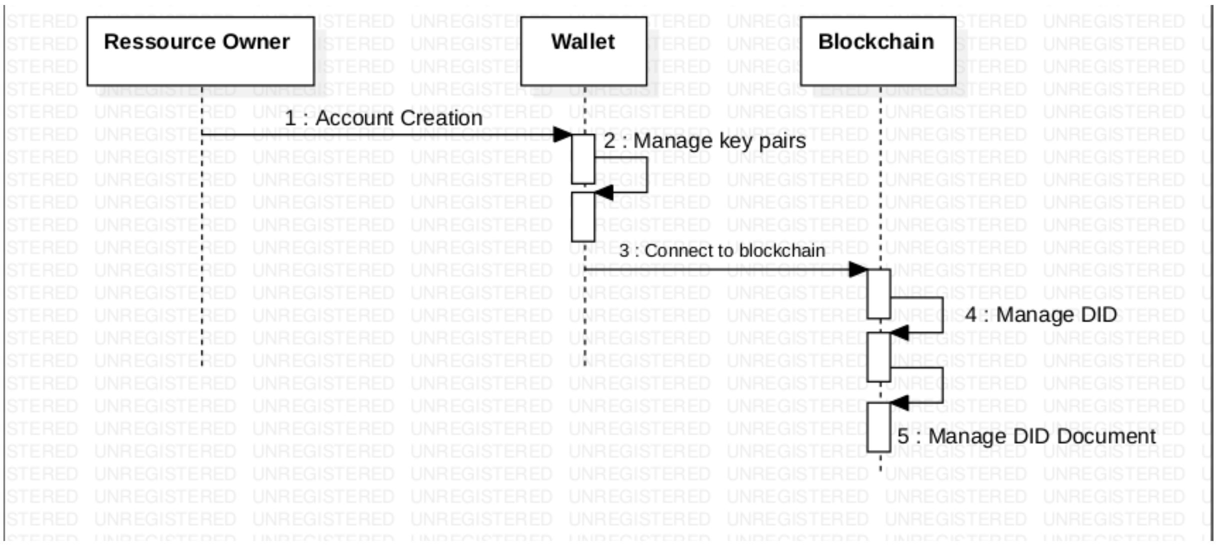


Figure 2.1: Sequence diagram:Identity creation

In this phase, the Resource Requester sends the Resource Owner a request to win the Claim OAuth then the Resource Owner processes the request from the Resource Requester and creates a Claim OAuth with signature. Signing a document uses both asymmetric cryptography and hash functions. It is indeed by the association of these two techniques that we can obtain the 5 characteristics of a signature (authentic, tamper-proof, non-reusable, unalterable, irrevocable). The signature phase proceeds as follows:

- First of all, Resource Owner generates the fingerprint of the OAuth Claim by means of a function of hash.
- Then, Resource Owner encrypts this fingerprint with his private key.
- Resource Owner thus obtains the signature of its OAuth Claim.
- Resource Owner therefore sends these two elements to the blockchain.

Then the Resource Owner transmits the OAuth Claim to the blockchain to register the new Claim OAuth in this time the blockchain runs the Etherum Claims Registry smart contract for backup the Claim OAuth then creates the hash of the latter thereafter the blockchain confirms the addition of Claim OAuth for the Resource Owner finally the Resource Owner transfers the OAuth Claim to the Resource Requester.

2.3 Creation

In Figure 2.2 we explain the steps needed to create an OAuth Claim.

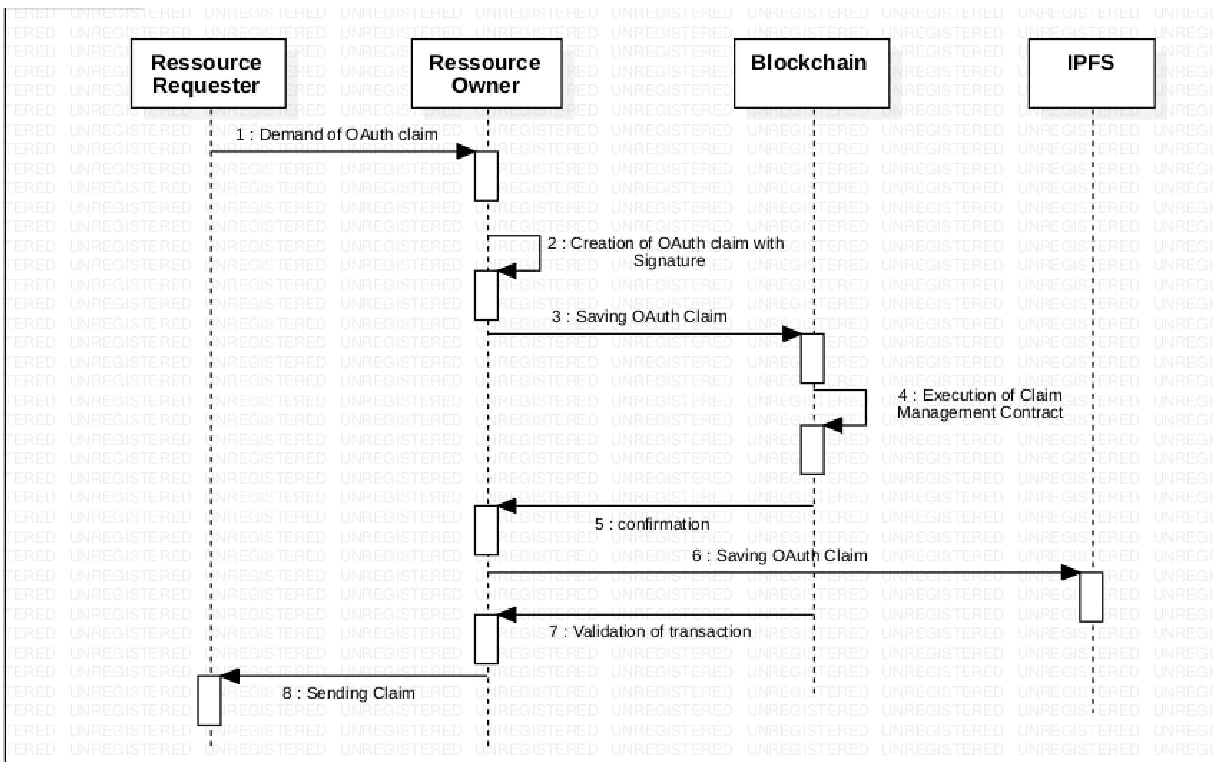

Ressource Requester
Ressource Owner
Blockchain
IPFS
1 : Demand of OAuth claim
2 : Creation of OAuth claim with Signature
3 : Saving OAuth Claim
4 : Execution of Claim Management Contract
5 : confirmation
6 : Saving OAuth Claim
7 : Validation of transaction
8 : Sending Claim

Figure 2.2: Sequence diagram:creation of claim OAuth

In this phase, the Resource Requester sends the Resource Owner a request to win the

Claim OAuth then the Resource Owner processes the request from the Resource Requester and creates a Claim OAuth with signature. Signing a document uses both asymmetric encryption and hash functions. It is indeed by the association of these two techniques that we can obtain the 5 characteristics of a signature (authentic, tamper-proof, non-reusable, unalterable, irrevocable).

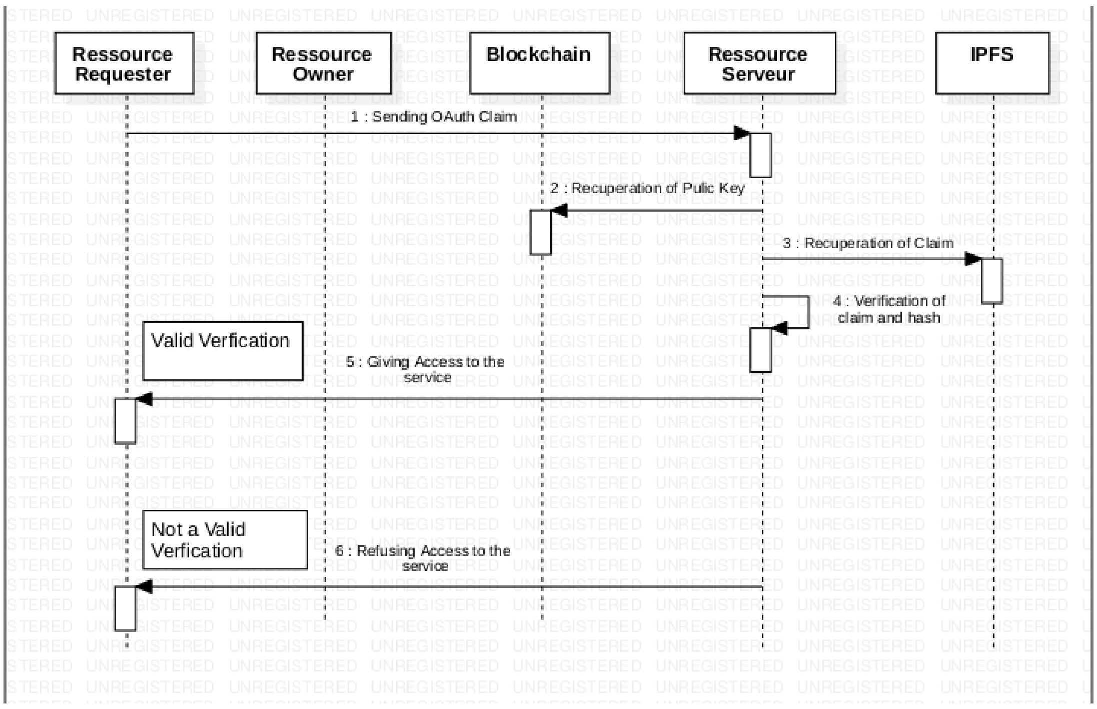
The signature phase proceeds as follows:

• First of all, Resource Owner generates the fingerprint of the OAuth Claim by means of a function of hash.
• Then, Resource Owner encrypts this fingerprint with his private key.
• Resource Owner thus obtains the signature of its OAuth Claim.
• Resource Owner therefore sends these two elements to the blockchain.

Then the Resource Owner transmits the OAuth Claim to the blockchain to register the new Claim OAuth in this time the blockchain runs the Etherum Claims Registry smart contract for backup the Claim OAuth then creates the hash of the latter thereafter the blockchain confirms the addition of OAuth Claim for the Resource Owner in IPFS. finally the Resource Owner transfers the OAuth Claim to the Resource Requester.

2.4 Verification

The verification phase is shown in Figure 2.3:



Ressource Requester
Ressource Owner
Blockchain
Ressource Serveur
IPFS
1 : Sending OAuth Claim
2 : Recuperation of Pulic Key
3 : Recuperation of Claim
4 : Verification of claim and hash
Valid Verfication
5 : Giving Access to the service
Not a Valid Verfication
6 : Refusing Access to the service

Figure 2.3: Sequence diagram:Verification of claim OAuth

In this part, we talk about the verification phase of Claim OAuth which starts after sending OAuth Claim to the Server Resource to access the service requested by the Requester Resource, then the Resource Server requests the recovery of the public key of the Resource Owner from DID document stored in the blockchain.

Then to check the validity of the OAuth claim, the Resource Server must first decrypt the signature using the Resource Owner's public key, Whether it doesn't work, it's because the OAuth Claim was not sent by Resource Owner. Then the Resource Server generates the hash of the OAuth Claim it received, using the same function of hash as the Resource Owner. (It will be assumed that they follow a protocol established beforehand.)

Then,the Resource Server get The OAuth claim from IPFS using the generated hash and then compares the fingerprint generated and that resulting from the signature.

If the two fingerprints are identical, the signature is validated. We are therefore only sure that:

- The Owner resource sent the OAuth claim.

- The OAuth claim has not been modified since which Resource Owner signed it.

- The Resource Requester wins the desired service.

Otherwise, it may mean that:

- The OAuth claim has been modified since it was signed by Resource Owner.

- This is not Verfialble Token which Resource Owner has signed.

- The service requested by the Resource Requester is refused.

2.5 Revocation

Figure 2.7 shows the steps for revoking an OAuth Claim.

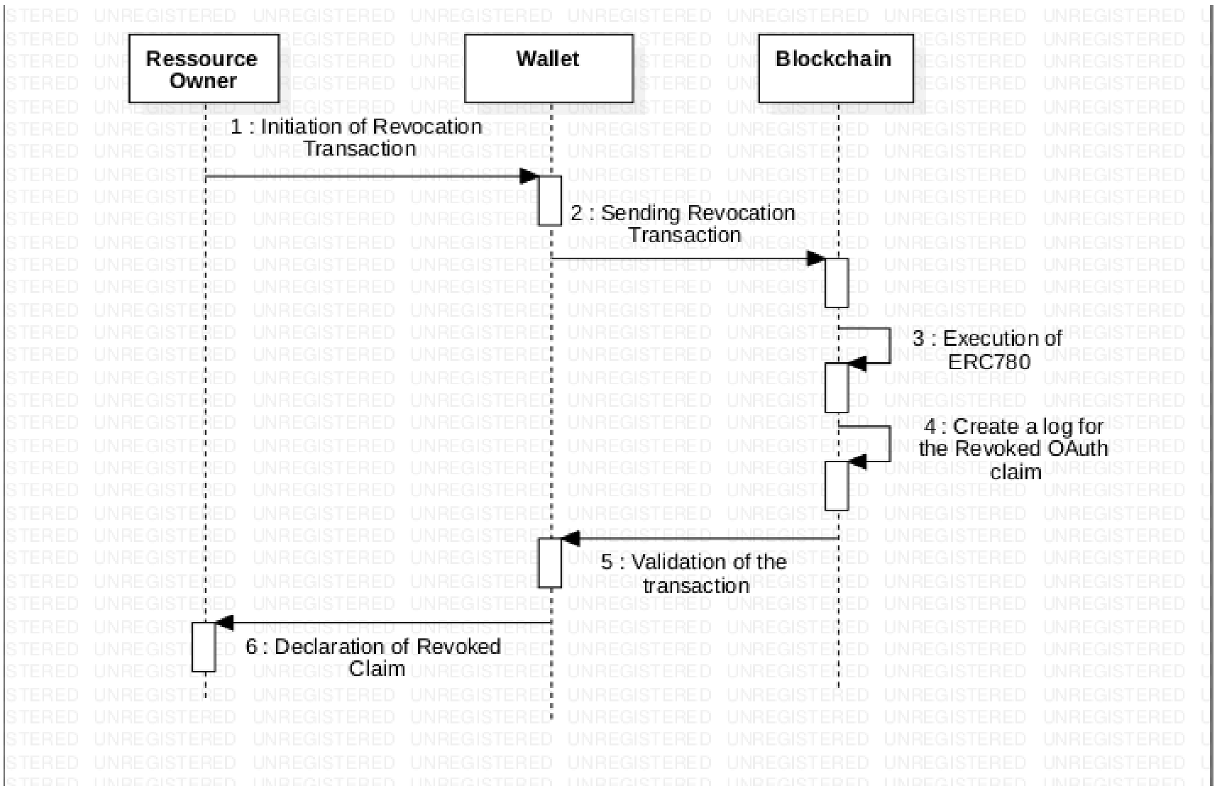

Ressource Owner
Wallet
Blockchain
1 : Initiation of Revocation Transaction
2 : Sending Revocation Transaction
3 : Execution of ERC780
4 : Create a log for the Revoked OAuth claim
5 : Validation of the transaction
6 : Declaration of Revoked Claim

Figure 2.4: Sequence diagram:Revocation of claim OAuth

The resource owner is responsible for this task, the procedure begins with an initiation of transaction created by the onwer resource on the wallet. Second step wallet sends the transaction from blockchain revocation.

Then the blockchain launches the execution of the ERC780 contract that is the manager , The fourth step is to create a log file for the deleted verifiable token.

Finally remains only the transaction validation for the Owner resource.

Chapter 3

Proof of concept

3.1 Introduction

In this chapter, we examine all the experiments carried out to evaluate the performance of the solution and we analyze the results obtained.

3.2 Comparative study

Choosing IPFS

IPFS (InterPlanetary File System) is a protocol and peer-to-peer (p2p) network for storing and accessing files, websites, applications, and data. It is built around a decentralized system. Each participant in the IPFS Network is called a node.

When you place a URL in the address bar of your browser, the computer asks one of the company's WEB servers, which may be located somewhere on the other side of the country (or even the planet), for the requested page. The server is located in a single place, which is called centralisation.

You can access any file thanks to its CID using IPFS protocol with this address:

i p f s :// <CID>

This is equivalent to this address using HTTP protocol :

h t t p : / / i p f s . i o / i p f s /<CID>

IPFS know how to find the information by its content, not its location. The content identifier (CID), which is a string of number similar to a hash, in the middle of the URL allow IPFS to ask lots of computer around the world to share the page with you, instead of asking one of the compagny's servers. When using IPFS, your computer does not just download files from someone else, it also helps distribute them.

Traditional URLs and file paths identify a path by where its located, while IPFS addresses a file by what's in it or by its content thanks to the content identifier. The CID is a unique cryptographic hash of the content of this address, calculating using a Merkle DAGs (Directed Acyclic Graphs). A DAG can be defined as a directed graph with a topological ordering and a Merkle DAG is a DAG where each node has an identifier and this is the result of hashing the node's content. When a file is put on IPFS, it is split into blocks depending on the file size. Each block has its CID. Then a new CID can be generated by combining the blocks in pairs until a root CID is obtained. This allows the integrity of a file to be checked because if a block is modified the root CID will be different.

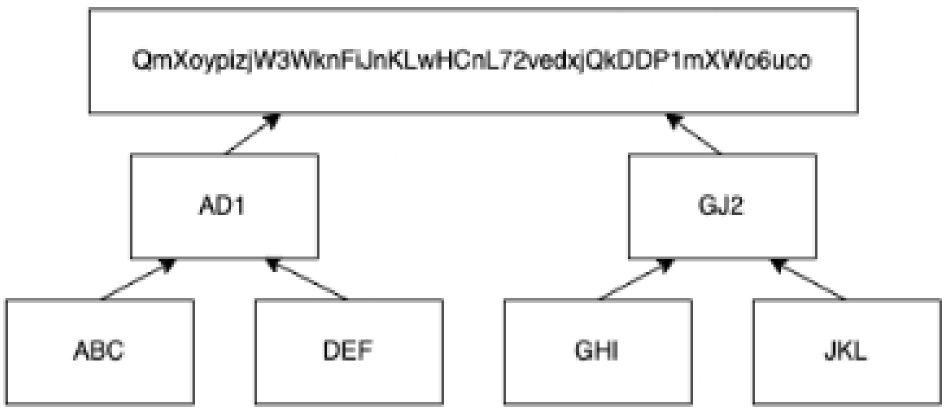


Figure 3.1: Merkle DAG

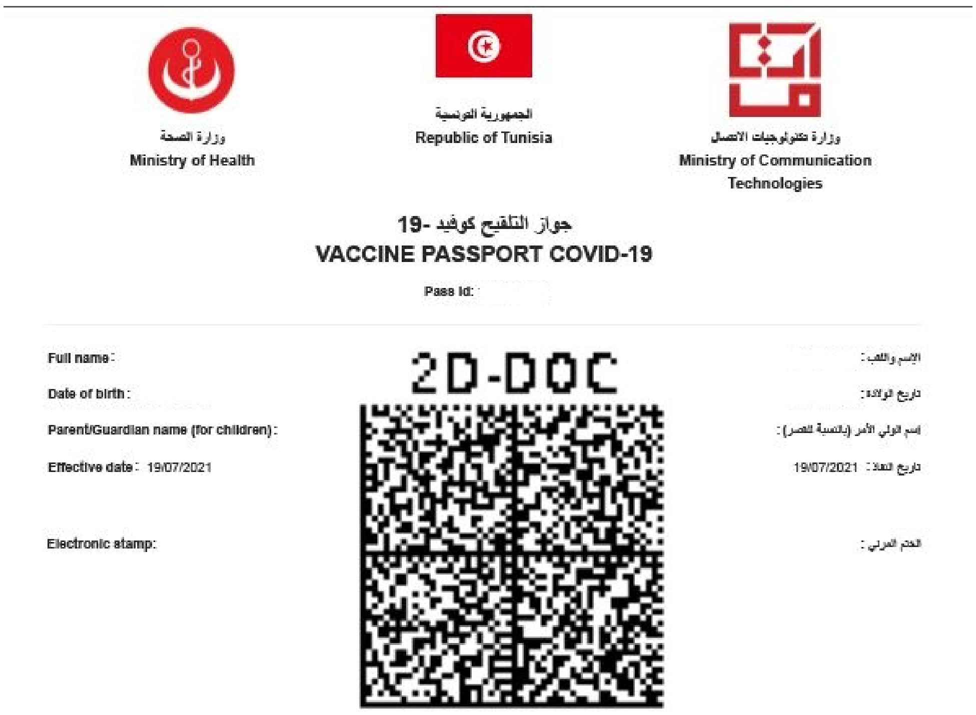
IPFS is considered to be the cornerstone of WEB3.0. It is starting to be deployed, as in the Brave browser. It addresses the problems of the centralised WEB.

Choosing a blockchain

The blockchain is supposed to be used to share information about personal medical records of the patients. Moreover, the validating nodes are supposed to be the computers of a consortium of hospitals working together. An external validator is not supposed to join the validating nodes. Thus, the blockchain must be private.

Moreover, we exclude the possibility to have observers nodes, since the people can connect to the blockchain and through a validating node, see the written pieces of information. The blockchain do not need to be hybrid, it stays private.

For private blockchain, we have restricted our choices to the ones that do not required to learn new languages, thus we excluded Tezos and Cardano. Then, our main solutions were Hyperledger, which required coding in python, or Polygon, a sidechain of Ethereum, which required coding in solidity.



وزارة الصحة
Ministry of Health

الجمهورية التونسية
Republic of Tunisia

وزارة تكنولوجيات الاتصال
Ministry of Communication Technologies

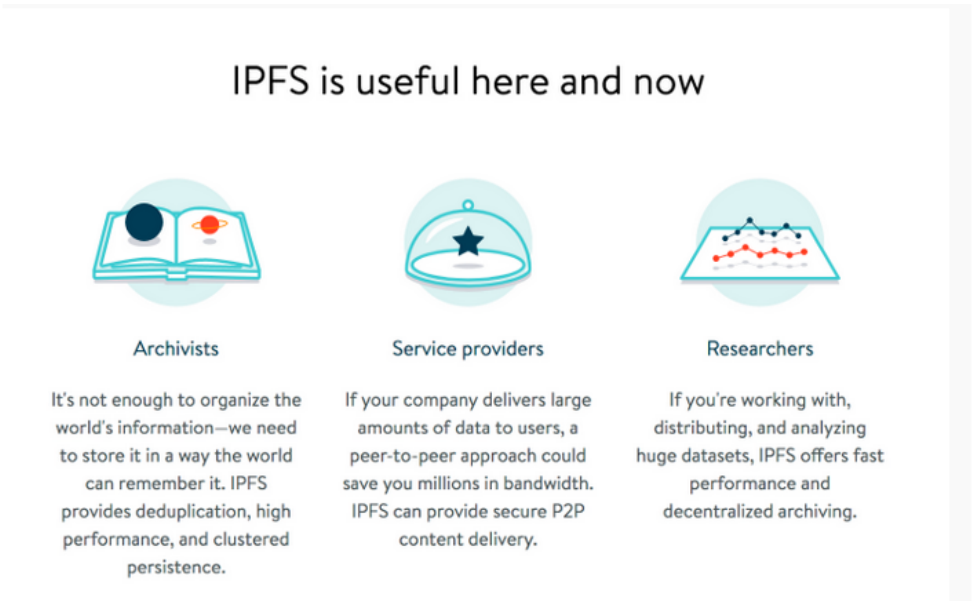
جواز التلقيح كوفيد -19
VACCINE PASSPORT COVID-19

Pass Id:

2D-DOC

Full name:	الإسم واللقب :
Date of birth:	تاريخ الولادة :
Parent/Guardian name (for children):	إسم الولي الأمر (بالنسبة للقصر) :
Effective date: 19/07/2021	19/07/2021 : تاريخ النفاذ
Electronic stamp:	الختم الرقمي :

Figure 3.2: Vaccine Certificate Provided By EVAX



IPFS is useful here and now

Archivists

It's not enough to organize the world's information—we need to store it in a way the world can remember it. IPFS provides deduplication, high performance, and clustered persistence.

Service providers

If your company delivers large amounts of data to users, a peer-to-peer approach could save you millions in bandwidth. IPFS can provide secure P2P content delivery.

Researchers

If you're working with, distributing, and analyzing huge datasets, IPFS offers fast performance and decentralized archiving.

Figure 3.3: IPFS

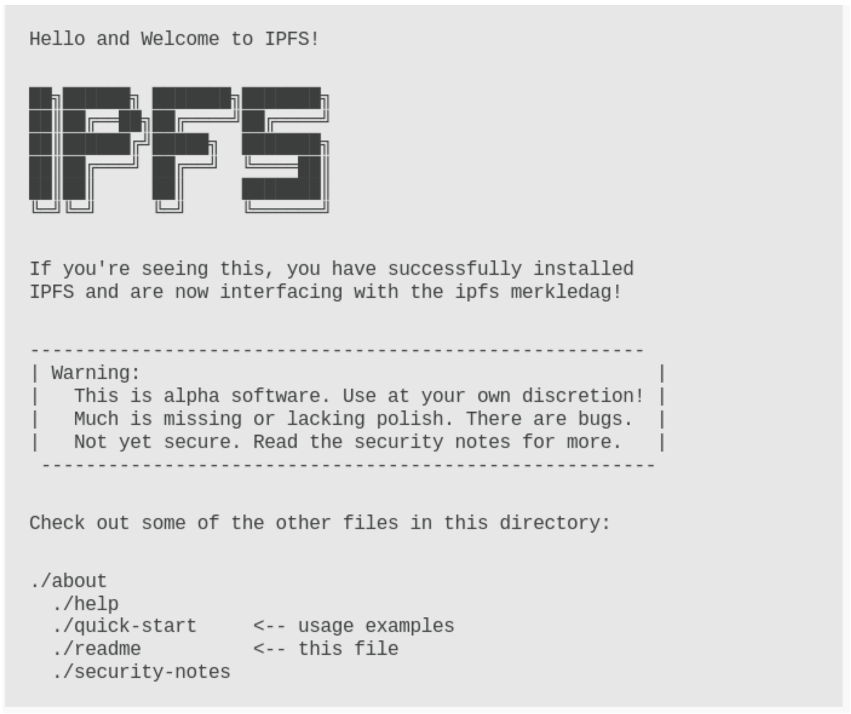
3.3 Implementation

3.4 How IPFS works

IPFS works by connecting all devices on the network to the same file structure. This file structure is a Merkle DAG, which combines Merkle trees (used in blockchains to ensure immutability), and Directed Acyclic Graphs (used in Git version control, which also allows users to see the versions of content on IPFS). Think of it as a large BitTorrent swarm.

Imagine you want to read the IPFS whitepaper. What you would normally do is type in a URL, which can be resolved to an IP address, which provides information about the location of the file (would be the IPFS servers, if they had those), which then allows your client to make a connection with the host and get the file. There are numerous ways in which this can go wrong, a lot of which have been discussed above.

Now let's get started. First, you'll need to initialize the ipfs repository. Do this with [ipfs init] . This will create a key-pair for your node and a repository with some ipfs objects:

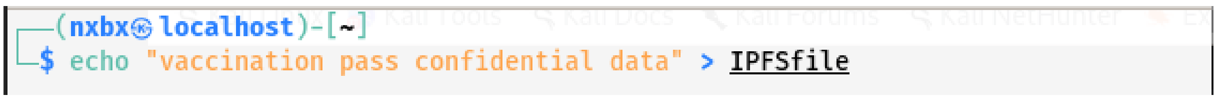


```
Hello and Welcome to IPFS!

If you're seeing this, you have successfully installed
IPFS and are now interfacing with the ipfs merkledag!

 -------------------------------------------------------
| Warning:                                              |
|   This is alpha software. Use at your own discretion! |
|   Much is missing or lacking polish. There are bugs.  |
|   Not yet secure. Read the security notes for more.   |
 -------------------------------------------------------

Check out some of the other files in this directory:

./about
  ./help
  ./quick-start     <-- usage examples
  ./readme          <-- this file
  ./security-notes
```

Figure 3.4: IPFS CLI

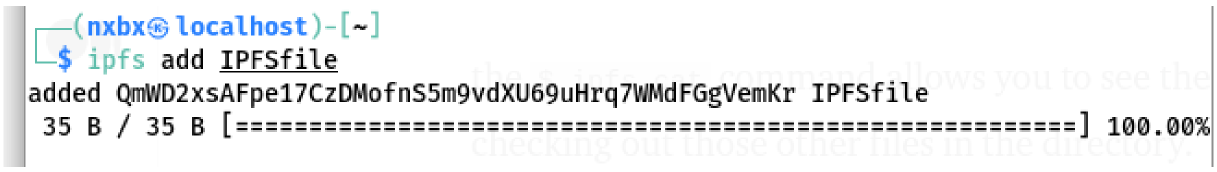
3.5 Adding Files to IPFS

Adding files to IPFS is really simple. Move to an IPFS test-repository and make a simple file:



```
┌──(nxbx㉿localhost)-[~]
└─$ echo "vaccination pass confidential data" > IPFSfile
```

Figure 3.5: IPFS CLI: creating a file

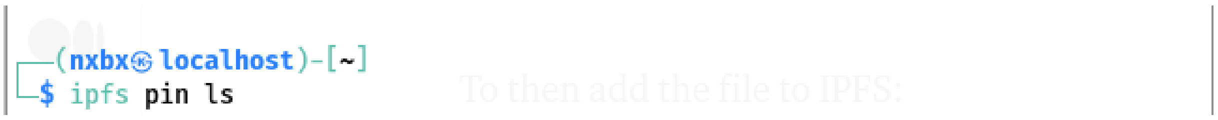


```
┌──(nxbx㉿localhost)-[~]
└─$ ipfs add IPFSfile
added QmWD2xsAFpe17CzDMofnS5m9vdXU69uHrq7WMdFGgVemKr IPFSfile
 35 B / 35 B [=========================================================] 100.00%
```

Figure 3.6: IPFS CLI: adding files

This will return a hash that starts with Qm that serves as the unique identifier for the content of that file. We have now recursively pinned this file to your local storage, which means that once you're connected to the swarm, people that have that hash identifier will be able to request that file to you.

To see which IPFS files are on your system, use:



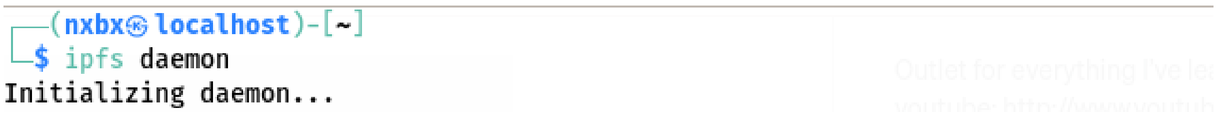
```
┌──(nxbx㉿localhost)-[~]
└─$ ipfs pin ls
```

Figure 3.7: IPFS CLI : list of existing files

This will list all the pinned files, we should (recursively) see the file you just made (as its hash identifier), as well as the files that were made when initializing ipfs, like readme and quick-start. To look at those files, we can ipfs cat them, unless they are a directory, which is also possible. In that case, use [ipfs ls] to look into that directory. If we wish to add a whole directory, simply add [-r] to the ipfs add command.

3.6 Interacting with the Swarm

Everything we've done so far, was done locally. We haven't connected to the swarm yet, to do that, initialize the IPFS daemon:



```
┌──(nxbx㉿localhost)-[~]
└─$ ipfs daemon
Initializing daemon...
```

Figure 3.8: IPFS CLI : initialize

we can now see the details of your connection using [ipfs id]. Let's check out our peers:

```
┌──(nxbx㉿localhost)-[~]
└─$ ipfs swarm peers
```

Figure 3.9: IPFS CLI : Connection details

Note that they all have a unique hash as their ID. we can inspect them with [ipfs id <insert hash>]

The IPFS daemon set up a localhost which allows you to interact with the IPFS network through your browser. The default is 8080.

Let's look at our files through the web-browser: [localhost:8080/ipfs/<hash of file>]

We should now see your file. As before, We can read the IPFS whitepaper on the network by inserting the hash of its content (QmV9tSDx9UiPeWExXEeH6aoDvmihvx6jD5eLb4jbTaKGps).

PS : It was very diffuclt to install smart contracts and complete the full integration of IPFS with the claim based OAuth, I'm limited to this stage due some issues related to my laptop perfermonce.

General Conclusion

Managing medical data is a crucial outcome for incoming years. The design of new architecture which can cope with new digital attacks techniques is essential to maintain a good level of safety is our organisations. In this work, we have investigate the fundamental concepts behind the IPFS technology, the Oauth2.0 architecture, cryptographic algorithms of encryption as well as centralized and decentralized system of storage. These tools were the conception bricks of a new healthcare architecture, much more based on the individual than on the institution. A special effort has been done in order to hide the most possible sensitive data and avoid correlations. Having a theoretical model, many choices had to been made to chose the most relevant features.