Ulnaria: A next generation decentralized wallet

Jerold Koutou

jeroldkoutou@gmail.com

www.ulnaria.flutterflow.app

Legal Disclaimer

Abstract

A completely decentralized wallet offers a combination of security and flexibility for transactions within the digital world in the digital and intelligent age. Indeed, a world governed by essentially digital interactions, through social networks as well as any digital platform, from commerce to financial services. These numerous interactions require both speed and security in a world where the notion of trust is increasingly called into question. Thus, we propose a solution to this problem both to the trust problem encountered in centralized systems through our non-custodial wallet, but also to the problem of crypto asset adoption, a real bulwark against the hegemony of sovereign currencies. We also resolve the problem related to the friction encountered by users when using traditional crypto wallets, by simplifying the user experience by setting up a simplified connection with the most familiar email address and security methods. We thus facilitate the recovery of crypto wallets, thus preventing the permanent loss of assets held by the user.

1. Introduction

The advent of crypto assets and blockchain technology has sparked one of the biggest revolutions in history. Previously, the world was governed by a system based exclusively on mutual trust. Everything functioned through an intermediary, a guarantor. This intermediary thus had the freedom to set the rules and ensure their compliance. However, contemporary history has taught us that these intermediaries, the guarantors of smooth operations, were in fact the first to abuse them with impunity, because who can punish the punisher? The balance of the world was thus based on a principle of structural imbalance adopted and accepted by all as the rule.

Humanity thus waited for the onset of yet another global economic crisis to witness the emergence of a new system, one that would then define the new world economic order by eliminating the intermediary and eradicating the very notion of trust. A system based on consensus, a system where power is no longer held by a minority but rather by all the members constituting the system. This is referred to as the era of decentralization.

It was with this in mind that Satoshi Nakamoto laid the foundation stone of a structure that would then be in perpetual evolution. A system that renounces the rigidity of the conventional world and embraces the flexibility of the new world, thus fostering innovation. A breath of fresh air has swept across the earth, and in its wake has seen the birth of protocols like Ethereum, introducing the decentralization of finance (DeFi), decentralized applications (dApps), and more.

Thanks to this, platform management is no longer left to the whim of managers but rather to the consensus of an active community. Ulnaria is part of the vision of an open and decentralized world. To contribute to this, the platform has implemented a non-custodial wallet, in order to offer each user the freedom to manage these funds and carry out peer-to-peer exchanges in complete security.

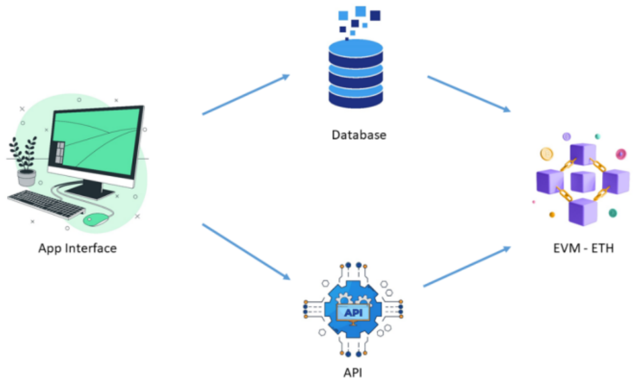


Figure 1: Open wallets must be built on a public and open blockchain. Separating the logic between the user database and API usage makes it easier to integrate a robust and flexible solution at the same time.

2. Existing Works

Decentralized wallet platforms implemented with Ethereum smart contracts have failed to generate significant volumes due to design inefficiencies that impose high friction costs on market makers. Indeed, traditional wallets, while decentralized, are often the center of high transaction fees, which represents significant friction for users, resulting in a slowdown in market transaction volume. If the cost of a transaction is low, the immediate change will be seen in the increase in transaction volume, which can contrast with transaction speed. It is true that user-side benefits do not always converge with those on the network side.

This can be observed in market phases during which the network can be as secure as possible with good transaction speed but, with low volumes, hampered by high fees. In addition to the high costs it imposes on market makers, managing a wallet on the blockchain generates transactions that consume network bandwidth and bloat the blockchain without necessarily resulting in value transfer.

Non-custodial crypto wallet smart contracts are proposed as an alternative to **custodial wallets** backed by centralized exchanges. The proposed smart contract replaces the centralized master wallet in the asset management of centralized wallets. This model allows users the freedom to exchange crypto assets between them without barriers and without an exchange intermediary. The advantages of non-custodial wallets include autonomy (the user is always given the freedom to dispose of these funds and define their different uses) and ease of integration with external smart contracts that can be executed in other wallets, this is called interoperability. Indeed, the wallet created by the user comes from a tree whose root is the Ethereum blockchain. Thus, the user can access their wallet from an external client other than Ulnaria.

3. Principe of custody

The custodian wallet system uses an implementation that allows it to manage child addresses from a **Master Address**. Child addresses are assigned to each user account upon creation, each user thus receives a deposit address to carry out their transactions. In fact, this address does not interact directly with the blockchain, except during transactions between a centralized exchange and other clients (exchange, wallet, etc.). Everything starts from a **Master Exchange Address** which is the address managed by the owner of the exchange. It is this address that stores the private keys and signs the transactions in the blockchain. This address derives directly from an xpub and a mnemonic created on a network like Ethereum. The master exchange address held by the exchange owner is used to create deposit addresses for each user but also the **Master Gas Pump** address often used to sponsor user transaction fees. This address is important and holds the **slave GP address contract** that is responsible for paying fees for sponsored accounts. It holds a predefined (or rather precalculated) amount of ether by the owner + some dust.

The master exchange address uses the xpub + its private key to create virtual accounts according to the **account abstraction** model. In this model, the platform is divided into two distinct parts: the on-chain part and the off-chain part. We described how the on-chain part works in the preceding lines. The on-chain part is therefore represented by the **Master Exchange wallet**, owned and managed by the owner himself.

4. Offchain Part

The off-chain part is represented by the account services. At the base, we first find the **virtual accounts**. Each virtual account (VA) represents a type of token. We therefore create virtual accounts for ethers, solanas, or even fiat currencies like euros and dollars. Each virtual account contains **virtual currencies** (VC), e.g., VA Ethereum is filled with VC Ethers. This results in a multi-virtual account architecture in which each user account corresponds to an aggregation of several virtual accounts. Virtual accounts are extremely essential to this model because, within a custodial wallet, transactions are not carried out pair-to-pair between blockchain addresses but rather between virtual accounts.

The interest here is in the execution speed because, the tokens are not actually directly exchanged on the main network (Ethereum in our case), these tokens are dormant and stored in the owner's wallet who can dispose of them as he wishes. At this point, the sacred principle of the blockchain is violated. The user is in reality dependent on the same system as that of Fiat, namely a system based on the trust of the latter with regard to the owner of the exchange or the custodial wallet.

5. Specifications

The account abstraction used in custodian wallets features a very precise operation in carrying out transactions. Each virtual currency is created with a **liquidity base rate** set by the owner. If they choose a 1:1 ratio, this implies that for each virtual currency created, they must keep the same amount of native token. However, they are free to choose a ratio of 1:4 or higher. To this end, they could then act like a traditional bank, which often only has a 10% fractional reserve. Users of such a system therefore run the risk of a bank route like those traditional banks that have repeatedly precipitated the world in economic crises, such as the one in 2008 that gave birth to Bitcoin.

As for Fiat-crypto conversions, the wallet simply creates a Fiat-type virtual currency (euro or dollar). To this end, when the user deposits native tokens, they can have their account credited directly in virtual units of account. To execute transactions, the deposit wallet can proceed in two different ways. First, an inter-app execution between two users who have an account themselves equipped with several virtual accounts. To do this, the wallet considers the two users (depending on their action) respectively as Debtor and Creditor. The Debtor is the one who sends the token or Fiat and the Creditor is the one who receives. Therefore, an addition is made at the creditor level via the Mint function, which increases their account by providing them with new virtual currency. The Debtor sees their account decrease due to the Remove function, which simply subtracts their balance. The model used by the deposit wallet is essentially identical to that used by the traditional financial system. This often earns it strict regulation because it is considered a financial institution, just like a bank. Moreover, these platforms offer a large portion of the services found in traditional banks, while adopting the same principles.

6. Non Custodial Wallet

Non-custodial wallets were designed with a decentralized model that can seamlessly interact with all decentralized finance products. Whether **dApps**, **DEXs**, or even **DAOs**, non-custodial wallets are tailored to fit seamlessly into a lock and key. Unlike custodial wallets, which follow a centralized model where the various addresses are managed by a master wallet, non-custodial wallets offer holders the freedom to control and maintain full ownership of their funds.

The first protocol used in the creation of a non-custodial wallet is **BIP-39**. This protocol allows to obtain a secret phrase at the origin of the public and private keys. First of all, we have an **Entropy** (128 bits) which initially gives the **SHA-256**.

Then, using SHA-256, we add a **Checksum** (4 bits) to this Entropy (128 bits), which gives us a total of 132 bits. We then split the 132 bits into **12 segments** of 11 bits each. We then convert each 11 bits into an **English Word** using a base of **2048** available words. We therefore obtain 12 words constituting the **Mnemonic code**.

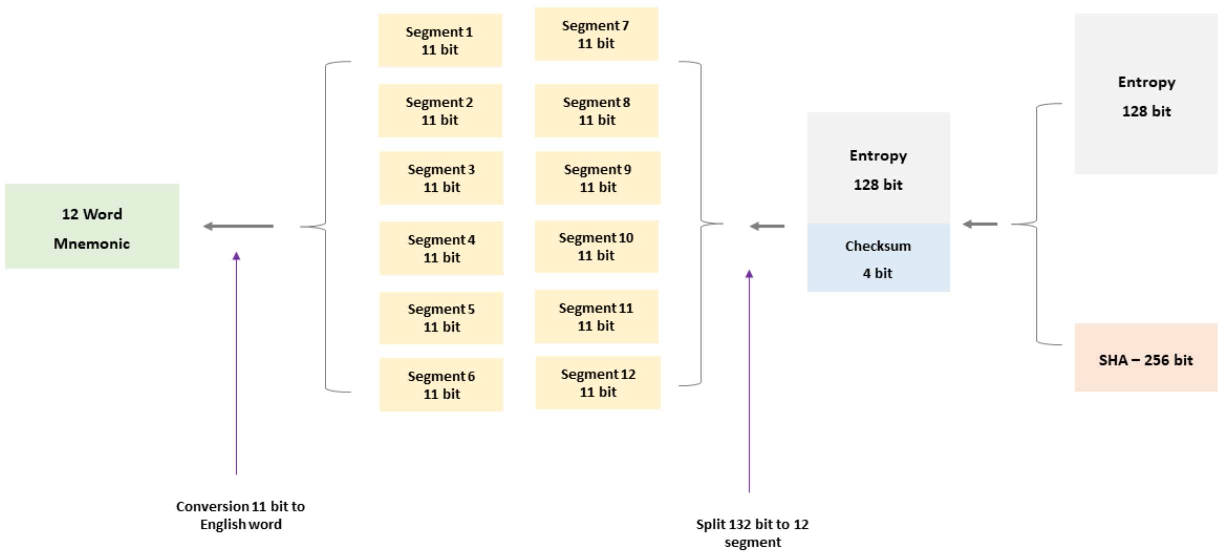


Figure 2: Generation du Mnemonic code à 12 mots a partir d'une Entropy de 128 bit et de SHA-256.

At this stage, we have the Mnemonic code (with its 12 words). We will therefore need a **Salt "Mnemonic"** and a password (optional). Thanks to the combination of the 2 components we create the **PBKF2 [HMAC – SHA 512]** which will use the 2048 words to generate the **Seed** (512 bit) with 2^512 combinaisno. The Seed (512 bit) is itself at the origin of the **HMAC – SHA 512 hash function.**

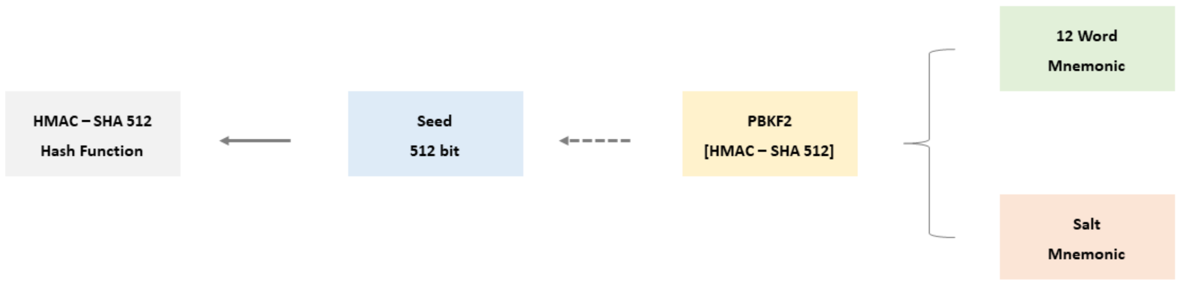


Figure 3: Process of obtaining the cryptographic hash function from the combination of the 12 Mnemonic words and the Salt Mnemonic. The result is therefore a hash function representing the terminal fruit of the BIP-39 proposal.

Once the hash function has been successfully obtained, we proceed to a bifurcation of said function into two (2) hashes. We separate the 512-bit function into two parts: The left hash gives us the **Parent Private Key** "**m**" (256 bit) and the right hash gives us the **Parent Chain Code** "**C**" (256 bit). The Parent Private Key will derive the **Parent Public Key** "**M**" (256 bit). We then proceed to a reaction cascade from the hash function. We use the Parent Private Key "m" which undergoes a hardened to give the one way hash function HMAC-SHA 512 (512 bit output) which then undergoes two (2) hashes in turn:

- ❖ ⍰ Hash left: Combination of HMAC-SHA 512 with the Parent Private Key (256 bit) "m" to give the **Child Private Key** (256 bit).
- ❖ ⍰ Hash Right: Combination of the Parent Public Key "M" (256 bit) with Parent Chain Code "C" (256 bit) and **index Number "i"** (32 bit). The result of this combination is the one way function

HMAC-SHA 512 which will then undergo a Left hash (216 bit) to give the **Child Chain Code** (256 bit).

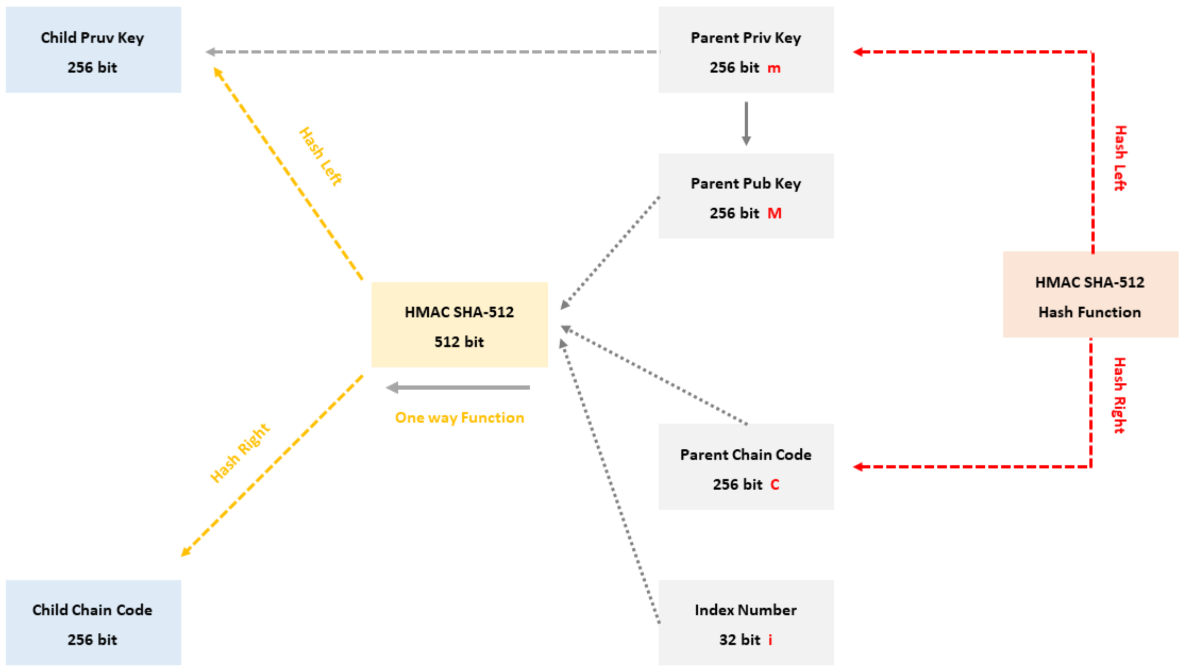


Figure 4: BIP-32 protocol. Creation of the Child Private Key and Child Public Key from the Parent Private Key “m”, Parent Public Key “M”, Parent Chain Code “C” and Index Number “i”.

The Child Private Key (256 bit) has 2^31 possible derivations. At first, Child Private Key generates the Child Public Key (256 bit) representing the public address. This results in the creation of two (2) key and major components :

- ❖ Combination of Child Private Key (256 bit) and Child Chain Code (256 bit) to create **Extended Private Key** (**Xprv** 512 bit). The Xprv in turn derives to give a **GrandChild Private Key** which itself derives into **GrandChild Public Key**. The branches continue to derive up to N.

- ❖ Child Public Key (256 bit) and Child Chain Code (256 bit) to create **Extended Public Key** (**Xpub**) which generates the **GrandChild Public Key**.

It is worth noting that for the Number index, if i < 2^31 == **Normal Child** and if i >= 2^31 == **Hardened Child**. Extended Parent can generate 2 billion Normal Child and 2 billion Hardened Child. And, each child can generate 4 billion children and so on.

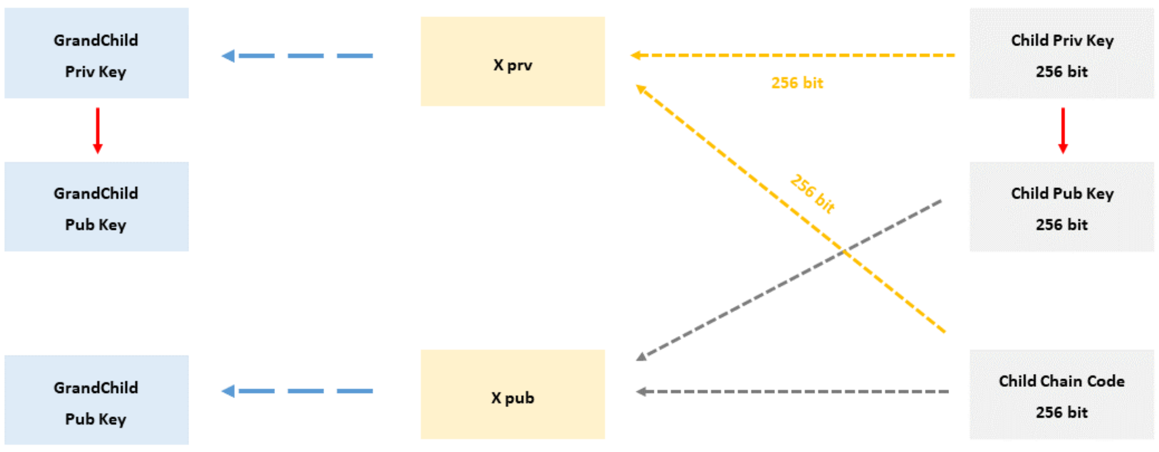


Figure 5: BIP-32 protocol. Creation of the GrandChild Private Key and GrandChild Public Key from the Extended Private Key (Xprv) and Extended Public Key (Xpub).

7. Hierarchical Deterministic Wallet

An HD wallet, or hierarchical deterministic wallet, is a type of cryptocurrency wallet that generates an unlimited number of addresses from a single seed phrase. This seed phrase is a sequence of words that, when entered into an HD wallet, generates all associated private keys and addresses. An HD wallet is a wallet that allows you to easily and securely generate multiple addresses from a single recovery phrase.

There are two general types of wallets in our model: "**Non-Deterministic**" wallets and "**Deterministic**" wallets. Non-deterministic wallets include the JBOK and the principle of random key generation. This model randomly generates keys to assign them to a wallet. We have not integrated it into our system, aware of these limitations. Furthermore, deterministic wallets rely on three major concepts: **Master key, Generate Key**, and **Seed**. This is where HD Wallets stem from, which are deterministic wallets that rely on an additional concept: Hierarchy. This type of wallet incorporates strong security elements such as **Entropy**, **Mnemonic code**, and **Backup**.

This protocol combines several proposals such as **BIP-39, BIP-32, BIP-43, and BIP-44.** BIP-39 is responsible for generating the Mnemonic code word. BIP-32 structures the algorithm for creating HD Wallets. BIP-43 handles the Multi-purpose HD Wallet. BIP-44 creates the Multicurrency Wallet and Multiaccount.

The protocol is designed according to the BIP-44 proposal. This protocol allows for the structured generation of addresses and keys. The system is based on a secure model using BIP-32, which we discussed earlier. It allows the creation of multi-purpose HD Wallets (BIP-43) coupled with multicurrency and multiaccount support. We therefore have a root on which all derivations are performed. The notation is reminiscent of an absolute link and is written as follow :

m / purpose’ / coin_type’ / account’ / change / addresse_index

:m / = corresponds to the root of the Derivation path. This is the Master Private Key “m”.

: **purpose ‘/** = It is assigned a number corresponding to an index that identifies itself to the chosen blockchain network; derivation path = m/44’/

: coin_type ’/ = corresponds to the type of token concerned; derivation path = m/44’/0’. These three (3) parts constitute the skeleton of the wallet

: **account ’/** = the account assigned to a user. Note that each account includes several “Change”. derivation path = m/44’/0’/0’

:**change /** = each change is able to generate several addresses according to the chosen index starting from 0. derivation path = m/44’/0’/0’/0, or m/44’/0’/0’/1

:**address_index** = the index of each address to create. This is any integer value starting from zero (0). derivation path = m/44’/0’/0’/1/0

We therefore use a Derivation path for the creation of the main wallet. The main (mother) wallet is built on a blockchain network by assigning it the index corresponding to that of the network on which it will be built (e.g.: BTC = 0). The main wallet thus created is called "**Master Node**". It is the cornerstone of the structure. This is where we find the "**Master Private Key**" m.

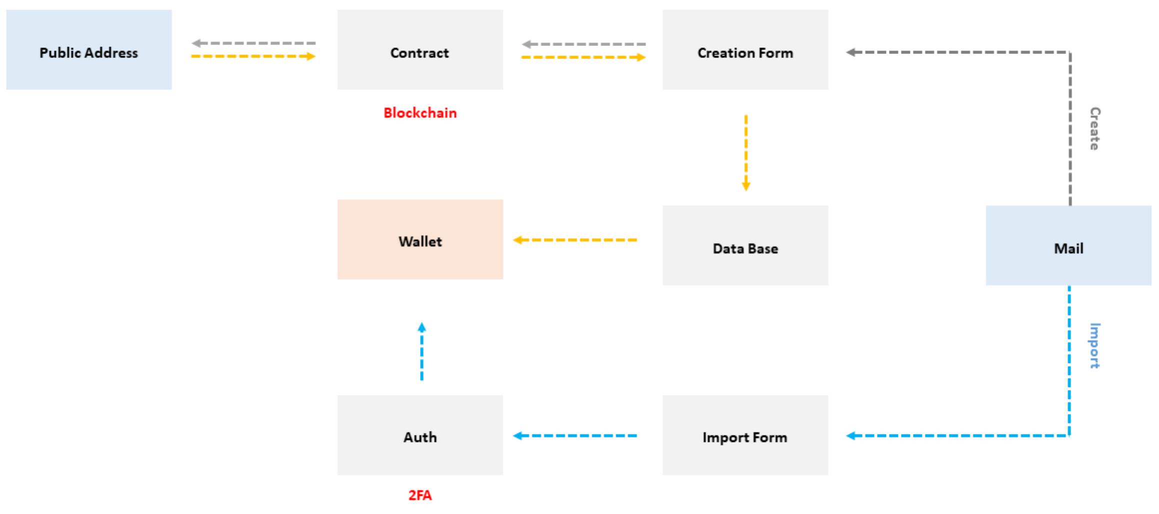
The wallet thus created has the capacity to generate accounts whose values start from 0 -> N. We call them **Wallet Accounts** which are in reality **ChildKey Derivation** (CKD = m/0). Each account created is independent of the other accounts.

Within the same account, we have the freedom to create several "**Changes**" starting from 0 -> N. We also call them **Wallet Chains**. They correspond to the **GrandChild Key** (m/0/0). They come from two (2) origins: on the one hand, **external CKD** and on the other hand, **internal CKD**. Finally, in each **Wallet Chain**, we will be able to generate addresses corresponding to the **Child Public Key**. This is the final stage of our creation. We therefore obtain a cryptographic tree.

8. Create / Import wallet

To manage wallet creation or import, we've chosen to implement a social login system. Unlike most wallets that use a complex wallet creation/import process, we prioritize ease of use and a seamless user experience. To allow users to access this, we've implemented a wallet creation method based on an **email address**.

The user is invited to enter his email address in order to create the wallet. This results in an available and active public address. Subsequently, the user is invited to import this wallet that he has just created, always from his email address. It is appropriate to specify the import email address must be the same as the creation email address. If the two (2) email addresses are identical then the user can access his wallet and consult his activity with complete peace of mind..



Conclusion

We have implemented a non-custodial wallet allowing crypto asset holders to securely store their tokens. To do this, we implemented a system using the BIP-32, BIP-39, BIP-43, and BIP-44 proposals. However, we have identified a barrier that interferes with the user experience. This concerns the management of public and private keys by the user. Indeed, when creating their non-custodial wallet, the user is required to keep the so-called "secret" phrase or Mnemonic code containing between 12 and 24 words. To remove this constraint, we offered users the possibility of creating their wallet without having to remember the Mnemonic code. Thus, the user can do everything from a login with their email address. Therefore, we have implemented various means to secure access to the user wallet via methods such as multi-factor authentication. Aware of the many challenges ahead, we are actively working to improve the user experience and strengthen the security means and methods for crypto asset holders.