Document Management

Collaborate, share, sign, and store your documents with ZK

NeosVault

Contents

1. Introduction

In today's digital age, the need for secure and decentralized document collaboration, sharing, signing, and storage has become increasingly important. With traditional methods of document handling becoming more obsolete, there is a need for an innovative solution that meets the demands of modern-day users. This is where NeosVault comes in.

NeosVault is an innovative project that aims to provide a decentralized platform for document collaboration, sharing, signing, and storage. The platform utilizes zero-knowledge proof technology to ensure the privacy and security of its users' data.

One of the key features of NeosVault is its zkKYC verification system, which enables non-repudiation of documents by real-world verification proofs. This means that users can prove the authenticity and integrity of their documents without revealing any sensitive information or compromising their privacy.

The platform also allows for seamless collaboration between multiple users, making it ideal for teams and organizations working on projects that require document sharing and collaboration. With NeosVault, users can securely share and sign documents, while also ensuring that their data is protected from unauthorized access.

In addition to its collaboration and sharing features, NeosVault also provides a secure storage system for all types of documents, including sensitive and confidential files. The platform utilizes advanced encryption techniques to ensure that user’s data is protected from theft or unauthorized access.

Overall, NeosVault is an exciting project that promises to revolutionize the way we collaborate and share documents online. With its advanced security features and decentralized architecture, the platform provides a secure and private way for users to share, collaborate, sign, and store documents.

This paper will provide an in-depth analysis of NeosVault, its features, benefits, and potential impact on the document collaboration, sharing, signing, and storage industry.

1.1 Early Document Management Systems

Document management systems have been in existence for centuries, with early examples dating back to the ancient Egyptians. These early systems involved the use of hieroglyphics to record important information, which was then stored on papyrus scrolls or clay tablets.

In more modern times, the earliest document management systems were paper-based, with documents being physically filed and stored in cabinets. These systems were time-consuming and labor-intensive, and it was difficult to organize and retrieve documents efficiently. The introduction of typewriters and carbon paper in the early 20th century made document creation and duplication easier, but the storage and retrieval issues persisted.

1.2 The Rise of Digital Document Management Systems

The advent of digital technology in the latter half of the 20th century brought about a significant change in document management. With the development of computers, it became possible to create, store, and retrieve documents electronically. This led to the rise of digital document management systems (DMS), which allowed for the efficient storage and retrieval of documents.

The first digital document management systems were designed to function as electronic filing cabinets, with documents being scanned and saved as digital files. These systems allowed for easy document retrieval and sharing, but they still required significant physical infrastructure, such as servers and storage devices.

1.3 The Limitations of Traditional Document Management Systems

While digital document management systems were a significant improvement over paper-based systems, they still had limitations. The centralized nature of these systems made them vulnerable to security breaches, unauthorized access, and loss of data. In addition, the use of traditional authentication methods, such as passwords, was not always effective in ensuring the security of sensitive documents.

Moreover, the need for third-party intermediaries for document verification and authentication created significant barriers to the efficient and secure sharing of documents. This is where NeosVault comes in.

NeosVault

2. NeosVaullt - The Future of Document Management

2.1 What is NeosVault?

NeosVault is a decentralized platform that allows users to collaborate, share, sign, and store documents securely. The platform utilizes zero-knowledge proof technology, which allows for privacy and security of users' data. With NeosVault, users can collaborate on documents without having to worry about data breaches or unauthorized access.

2.2 How does NeosVault work?

NeosVault works by using a decentralized architecture, which means that it is not reliant on a central authority to manage its operations. Instead, the platform relies on a network of nodes to ensure that data is stored and accessed securely. The platform uses advanced encryption techniques to protect user data, ensuring that it is secure and protected from unauthorized access.

2.3 What are the benefits of using NeosVault?

One of the main benefits of using NeosVault is the platform's advanced security features. With zero-knowledge proof technology and advanced encryption techniques, users can be confident that their data is protected from theft or unauthorized access.

In addition to its security features, also provides a decentralized platform for document collaboration, sharing, signing, and storage. This means that users can collaborate on documents without having to rely on a centralized authority, making it ideal for organizations and teams that require secure and private document sharing and collaboration.

2.4 Elementary Components

The NeosVault solution for decentralized document management and sharing builds upon two components:

- NeosVault CMS ZkApp: The decentralized document management and collaboration platform.
- NeosVault ZkKYC: A KYC and DiD verifier solution to allow the users to be the owners of their digital identities and manage the access to their ID documents.

2.5 zkApp Overview

A zkApp, short for zero-knowledge proof application, is a type of decentralized application that utilizes zero-knowledge proofs to ensure the privacy and security of user data. Zero-knowledge proofs are a type of cryptographic protocol that allows for secure and private transactions without the need for intermediaries. This means that zkApps can enable secure and private data sharing and transactions without the need for a central authority or third-party intermediaries.

In the context of NeosVault, a zkApp is utilized to provide a decentralized platform for document collaboration, sharing, signing, and storage. By utilizing zero-knowledge proofs, NeosVault ensures that only authorized parties can access documents, while also providing a high degree of privacy and security for its users. This makes NeosVault a cutting-edge solution for document management in today's digital age.

A “zkApp” consists of two parts: 1) a smart contract and 2) an UI (user interface) for users to interact with it.

NeosVault

3. Definition of a Decentralized Zero Knowledge Content Management System

A Decentralized Zero Knowledge Content Management System (DZKCMS) is a content management system that is designed to store, manage, and distribute digital content such as documents, images, videos, and other media in a decentralized zero knowledge blockchain. In a DZKCMS, the content is distributed between off-chain storage and on-chain storage.

3.1 Zero Knowledge Proofs, Provers and Verifiers

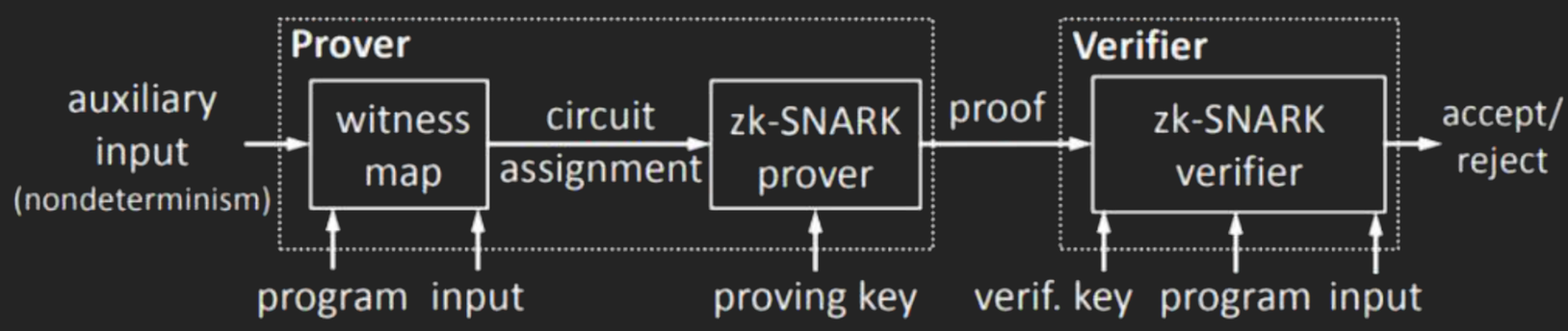
A **zero knowledge proof** is a cryptographic protocol that allows one party, known as the prover, to prove to another party, known as the verifier, that they know a particular piece of information or that a particular statement is true, without revealing any additional information beyond the validity of the statement itself.

In other words, a zero knowledge proof allows the prover to demonstrate knowledge of a solution to a problem, or of the truth of a statement, without actually revealing the solution or the statement itself.

This is achieved through a complex mathematical algorithm that generates a proof, which can be verified by the verifier. The proof allows the verifier to be confident that the prover has the necessary knowledge, without disclosing the knowledge itself.

NeosVault

Zero knowledge proofs are particularly useful in situations where privacy is a concern, as they allow parties to verify each other's claims without sharing any sensitive information.



3.2 Zk-SNARKs

A zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) is a proof system to achieve security and efficiency, is the truly ingenious method of proving that something is true without revealing any other information

In any zero-knowledge proof system, there is a prover who wants to convince a verifier that some statement is true without revealing any other information, e.g., verifier learns that the prover has more than X in his bank account but nothing else (i.e., the actual amount is not disclosed). A protocol should satisfy three properties:

- Completeness — if the statement is true then a prover can convince a verifier
- Soundness — a cheating prover can not convince a verifier of a false statement
- Zero-knowledge — the interaction only reveals if a statement is true and nothing else

A SNARK is a succinct argument of Knowledge which is complete, knowledge sound and succinct. A zk-SNARK is a SNARK and is zero knowledge.

3.3 Security Properties of a Succinct Blockchain

NeosVault is build on top of the Mina blockchain, we will now enlist the security properties of a succinct blockchain. Rather than the blockchain summaries, the properties pertain to the underlying blockchain guaranteed by the chain extractability property.

Consider a blockchain protocol Π and an execution $\mathcal{E}$. Let **C** be the underlying blockchain of the blockchain summary **S** in $\mathcal{E}$. We recall the following properties that were first rigorously formulated in [15]. We will assume that time is divided into predefined slots.

Common Prefix (CP); with parameters $k \in \mathbb{N}$. The blockchains C1, C2 corresponding to two alert parties at the onset of the slots sl1 ≤ sl2 are such that $C1^{\lceil k} \preccurlyeq C2$, where $C1^{\lceil k}$ denotes the blockchain obtained by removing the last k blocks from C1 and $\preccurlyeq$ denotes the prefix relation.

Chain Growth (CG); with parameters $\tau \in (0, 1]$ and $s \in \mathbb{N}$. Consider C, a blockchain possessed by an alert party at the onset of a slot sl. Let sl_1 and sl_2 be two previous slots for which $sl_1 + s \leq sl_2 \leq sl$, so sl_1 is at least s slots prior to sl_2. Then $|C[sl_1,sl_2]| \geq \tau \cdot s$. We call τ the speed coefficient.

Chain Quality (CQ); with parameters $\mu \in (0, 1]$ and $k \in \mathbb{N}$. Consider any portion of length at least k of the blockchain corresponding by an alert party at the onset of a slot; the ratio of blocks originating from alert parties in this portion is at least μ, called the chain quality coefficient.

3.3.1 The Blockchain Protocol Components

NeosVault use Mina Protocol blockchain for notarization and on-chain storage.

Consensus mechanism (UpdateConsensus, VerifyConsensus): The consensus mechanism is the Ouroboros Samasika protocol. In Ouroboros Samasika VerifyConsensus runs in time poly(**λ**), as required.
Blocks: Consider a transition $ti' \in Tau'$ that acts on a state $\sigma i' - 1$ The corresponding block $B_i = (sn_i, st_i, \pi_i^B, d_i, \text{b-pk}_i, \text{b-sig}_i)$ is constructed with $st_i = \sigma_{i-1}'$, π_i^B = consensusProof$_i$ and $d_i = t_i$.
State transition system for SNARK: Consider the state transition system (Σ, T, Update) defined as follows: Σ = {H(σ'), consensusState} $\sigma' \varepsilon \Sigma'$, $_{\text{consensusState}}$. A transition is a block. The function Update(B_i ,σ_{i-1}) verifies if H(st_i) = σ_{i-1}, the signature verifies in the block verifies and that VerifyConsensus(consensusState$_{i-1}$, consensusProof$_i$), where consensusState$_{i-1}$ is part of σ_{i-1} and π_i^B contains consensusProof$_i$.
Blockchain summary: The blockchain summary consists of a state in Σ and a proof $snark_i$.
VerifyChainSummary(S_{i-1} = (σ_i ,$snark_i$)): As mentioned earlier, this algorithm simply verifies the proof $snark_i$ against the statement σ_i.
VerifyBlock(S_{i-1} , B_i): This algorithm simply checks the consistencies, namely, verifying consensus, verifying signature and verifying that the state in S_{i1} is the hash of the state in the block.
UpdateChainSummary(S_{i-1}, B_i): Let S_{i-1} = (σ_{i-1}, $snark_{i-1}$). This algorithm runs the Update function as Update(B_i , σ_{i-1}) to obtain σ_i . Then, it verifies the proof $snark_{i-1}$. Finally, it runs the prover to obtain the new proof $snark_i$.

3.3.2 Consensus Protocol

NeosVault is desiogned to use the Mina Blockchain. Mina protocol uses Ouroboros Samasika, a provably-secure PoS consensus protocol that is adaptively secure and offers bootstrapping from genesis.

Ouroboros Samasika is a proof-of-stake blockchain consensus protocol that was introduced in 2019 as part of the Cardano blockchain platform. This protocol is an extension of the original Ouroboros protocol, which was introduced in 2017, and it addresses some of the limitations of the original protocol.

In the Ouroboros Samasika protocol, a set of validators, also known as "stakeholders," are responsible for validating transactions and creating new blocks in the blockchain. These stakeholders are selected based on their stake in the network, which is the amount of cryptocurrency they have "staked" or committed to the network.

The protocol is based on a system of epochs, which are fixed periods of time during which a specific set of stakeholders are responsible for validating transactions and creating new blocks. Within each epoch, the stakeholders are further divided into smaller groups, called "committees," which are responsible for validating transactions and creating new blocks in a decentralized manner.

To ensure the security and decentralization of the network, the Ouroboros Samasika protocol incorporates a number of cryptographic techniques, including random number generation, encryption, and digital signatures. These techniques ensure that stakeholders are selected fairly and that transactions are validated in a secure and decentralized manner.

Overall, the Ouroboros Samasika protocol is designed to provide a high level of security, scalability, and decentralization, making it well-suited for use in large-scale blockchain networks.

3.4 Encryption

In a zero-knowledge (zk) app, encryption is used to protect sensitive data from being accessed by unauthorized parties. Encryption is the process of transforming plain text into ciphertext using a cryptographic algorithm and a secret key. The ciphertext can only be decrypted back to the original plaintext by someone who has the secret key.

In a zk.app, encryption is typically used to protect user data, such as login credentials or personal information, from being accessed by unauthorized parties. Here are some key aspects of encryption in a zk.app:

Zero-knowledge encryption: In a zk.app, encryption is often performed in a way that allows the data to be encrypted and decrypted without revealing the secret key or the original plaintext. This is achieved through the use of zero-knowledge proof systems, which allow two parties to prove knowledge of a secret without revealing the secret itself. This ensures that the sensitive data is protected even from those who have access to the encrypted data.

Strong encryption algorithms: In a zkApp, strong encryption algorithms are used to ensure that the encrypted data is protected from attacks by malicious parties. The most common encryption algorithms used in zkApp are symmetric-key encryption algorithms like AES and public-key encryption algorithms like RSA.

Secure key management: In a zkApp, secure key management is essential to ensure that the secret keys used for encryption are kept safe from unauthorized access. This often involves using key management systems that allow for secure generation, storage, and distribution of keys.

End-to-end encryption: In a zkApp, end-to-end encryption is often used to ensure that data is encrypted and decrypted only by the sender and the intended recipient. This means that the data is encrypted on the sender's device and decrypted on the recipient's device, ensuring that the data is protected from interception and decryption by third parties.

In summary, encryption in a zkApp involves using zero-knowledge proof systems, strong encryption algorithms, secure key management, and end-to-end encryption to protect sensitive data from being accessed by unauthorized parties.

3.5 On-chain, Off-chain and Decentralized Storage

On-chain storage refers to the storage of data within a blockchain network itself, rather than in external databases or storage systems.

Smart contracts can also be used to implement on-chain storage by allowing developers to define and store data structures within the blockchain.

NeosVault uses Mina protocol Smart contracts for On-chain state storage, the state is read and updated using Mina's SDK SnarkyJS.

In the current Mina's design, the state can consist of at most 8 Fields of 32 bytes each. These states are stored on the zkApp account. Some structs take up more than one Field: for example, a PublicKey needs 2 of the 8 Fields.

Our implementation of NeosVault stores the following fields in the Smart Contract:

- Document ID
- Document Hash
- User Address
- Access root hash

On-chain storage using smart contracts provides several benefits, including increased security and transparency. Since the data is stored on the blockchain, it is tamper-proof and cannot be easily modified or deleted without the consent of the network participants. This ensures that the data is secure and reliable, and can be easily audited by external parties.

IPFS (InterPlanetary File System) is a distributed file system that aims to make the web faster, safer, and more open. Using the decentralized storage provided by IPFS has the following benefits:

1. **Decentralization**: IPFS is a peer-to-peer network, which means that files are not stored in a central server but distributed across the network. This reduces the risk of a single point of failure, making it more resilient to attacks and more reliable.
2. **Faster content delivery**: Traditional web protocols like HTTP rely on centralized servers to deliver content. IPFS, on the other hand, uses a distributed network of nodes to deliver content. This means that content can be served from the closest node, reducing latency and improving content delivery speed.
3. **Permanent storage**: IPFS uses content-addressed storage, which means that files are identified by their content rather than their location. This ensures that files can be retrieved even if the original uploader is offline or has deleted the file.
4. **Version control**: IPFS allows for version control of files, making it easier to track changes and roll back to previous versions of a file.
5. **Censorship-resistant**: Since files are distributed across the network, it is much harder to censor or remove content from IPFS. This makes it an attractive option for hosting content that may be deemed controversial or sensitive.

Overall, IPFS storage provides a more resilient, faster, and decentralized alternative to traditional web protocols, making it an important technology for the future of the internet.

NeosVault

3.6 Smart Contracts

NeosVault is designed to use the Mina Blockchain. In Mina protocol a “smart contract” is a program written in SnarkyJS and uses Typescript as development language. In Mina protocol, the smart contract code execution is not performed in the blockchain, it is executed in the transaction sender side, off-chain.

When a contract is deployed, the deployment process sends a transaction containing the verification key to an address on the Mina Blockchain.

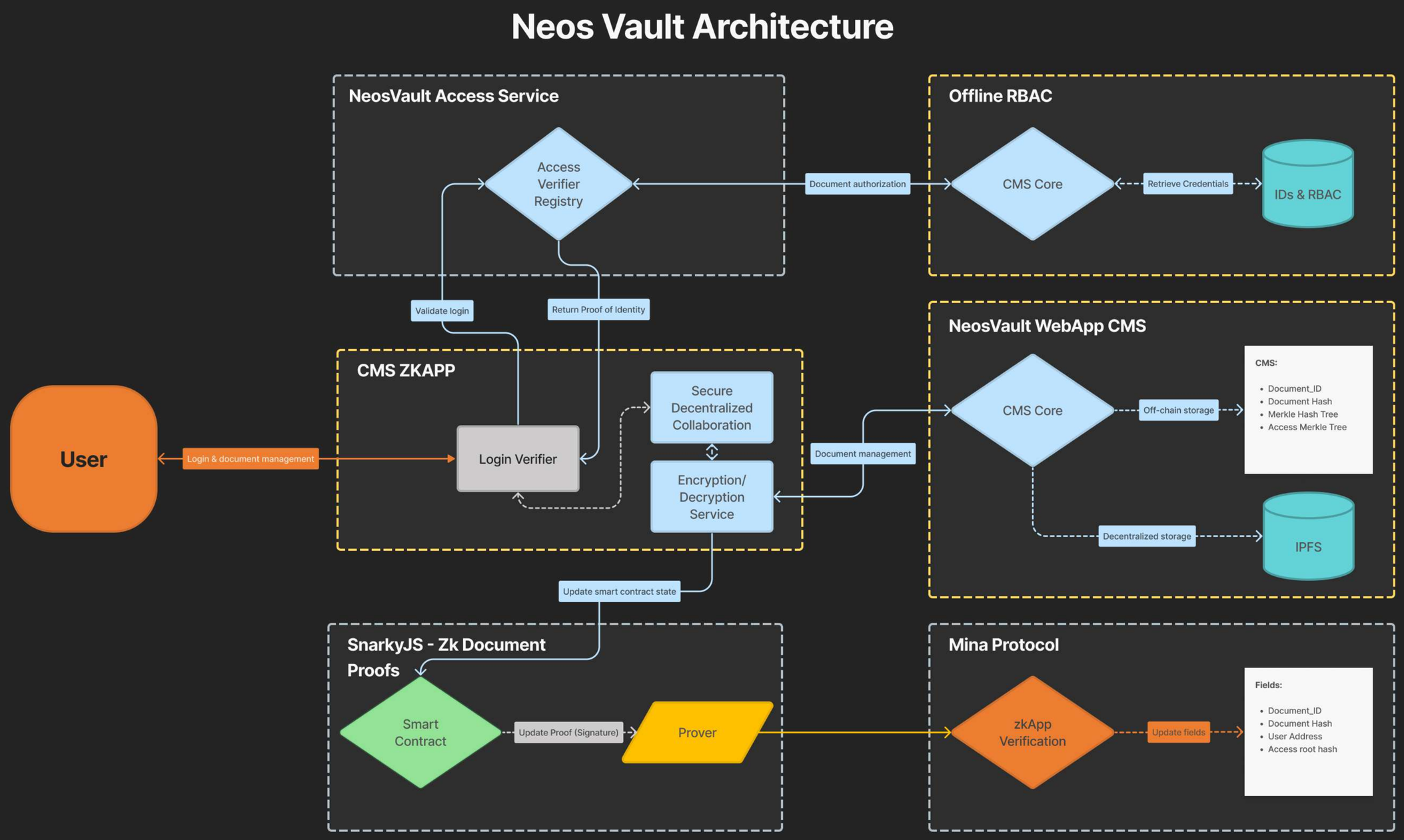
On Mina, there is no strong distinction between normal "user accounts" and "zkApp accounts". A zkApp account is just a normal account that has a smart contract deployed to it – which essentially just means there's a verification key stored on the account, which can verify zero-knowledge proofs generated with the smart contract.

3.7 Transactions

The fundamental data structure that Mina transactions are built from is called an account update. An account update always contains updates to one specific on-chain account. For example, if you transfer MINA from one account to another, that will update the balance on two accounts – the sender and the receiver. Therefore, sending MINA requires two account updates. As you'll see later, account updates are a flexible and powerful data structure, which can express all kinds of updates, events and preconditions that you'll need for developing smart contracts.

4. CMS zkApp

NeosVault is a decentralized platform that allows users to collaborate, share, sign, and store documents securely.



4.1 Secure Decentralized Collaboration

Secure decentralized collaboration refers to a way of working together where multiple parties can collaborate without the need for a centralized authority or intermediary. In the context of blockchain technology, this collaboration is made possible through the use of distributed ledger technology, which enables secure and transparent transactions between parties.

Blockchain technology operates on a decentralized network of nodes that maintain and verify the blockchain's integrity through a consensus mechanism. This network is made up of many nodes, each of which holds a copy of the blockchain's ledger, ensuring that any changes made to the blockchain are validated and agreed upon by the network.

With this technology, secure decentralized collaboration becomes possible as each participant can have access to the same information without needing to trust a central authority or intermediary. The transactions are transparently recorded on the blockchain and verified by the network, ensuring that all parties have a shared understanding of the information.

Additionally, blockchain technology uses cryptography to secure transactions and protect against fraud or unauthorized access. This ensures that sensitive information is kept private and secure, further enhancing the security of the decentralized collaboration.

Overall, blockchain technology provides a secure and decentralized way for multiple parties to collaborate without the need for a centralized authority or intermediary, making it an attractive option for a wide range of applications, from financial transactions to supply chain management.

4.2 Encryption and Decryption Service

NeosVault allows multiple decentralized participants to collaborate on a document and access it to read or update, depending on a permissions and role based access list, each participant will access out service using their public key.

To ensure the security and privacy of the information, NeosVault is designed to encrypt the documents before sending them to the off-chain storages like our CMS plantform and IPFS decentralized storage.

Documents encryption flow:

To encrypt a file for multiple public keys using a technique called hybrid encryption. Hybrid encryption combines the advantages of both symmetric and asymmetric encryption. Here are the high-level steps you can follow to encrypt a file for multiple public keys:

1. **Generate a symmetric encryption key**: Choose a strong symmetric encryption algorithm, such as AES, and generate a random symmetric encryption key. This key will be used to encrypt the file.
2. **Encrypt the file**: Use the symmetric encryption key to encrypt the file.
3. **Generate a unique random key for each public ke**y: For each public key, generate a unique random key. These keys will be used to encrypt the symmetric encryption key.
4. **Encrypt the symmetric encryption key for each public key**: Use the public key of each recipient to encrypt the unique random key generated in step 3. This will create an encrypted copy of the symmetric encryption key for each recipient.
5. **Send the encrypted file and encrypted symmetric keys to the recipients**: Send the encrypted file and the encrypted symmetric keys to the recipients. Each recipient can then use their private key to decrypt the encrypted symmetric key that corresponds to their public key. They can then use the symmetric encryption key to decrypt the file.

By encrypting the symmetric encryption key with each recipient's public key, you ensure that only the intended recipient can decrypt the symmetric key and access the file. This way, you can securely share a file with multiple recipients without compromising its confidentiality.

In the decryption process, the users need to be connected with their wallet to decrypt the document’s key, but even if a malicious entity access the encrypted key, it can’t access the encryption key because the private key of each user is in their Auro Wallet.

4.3 NeosVault Acces Service

This NeosVault component implements an internal document authorization service API for collaboration and sharing. Its role is to validate user's login and generate a proof of identity that will be use for the user interaction with the CMS zkApp.

The NeosVault Access Service interacts with the Offline role based access control (RBAC) service to authorize the users and identify their roles.

4.4 Offline Role Based Access Control for Document Authorization

Role-Based Access Control (RBAC) is the method of restricting access to NeosVault resources based on the roles of individual users within the platform. In RBAC, users are assigned roles based on their collaboration level, and access to resources is granted based on those roles.

RBAC is based on the principle of least privilege, which means that users are granted only the permissions necessary to access only the allowed documents. This reduces the risk of unauthorized access and helps to maintain the integrity of NeosVault.

RBAC has three primary components: roles, permissions, and users. Roles are defined based on the kind of collaboration a user will provide. Permissions are defined as the actions that users are allowed to perform on the documents, such as read, write, or archive. Users are assigned to roles, and those roles determine the permissions that the user has. When a document owner assign a role to someone, that person will inherit the permissions of the role.

RBAC provides several benefits over traditional access control methods. For example, it simplifies the management of access control by reducing the number of individual permissions that must be managed. This reduces the risk of errors and makes it easier to track who has access to what document. Additionally, RBAC provides a scalable solution for access control, making it well-suited for large organizations with many users and documents.

Role-Based Access Control is the method implemented by NeosVault for restricting access to documents based on the roles of individual users within an organization or collaboration group. It provides a scalable, manageable, and secure solution for access control, helping organizations to maintain the integrity of their information and reduce the risk of unauthorized access event in a open decentralized environment like NeosVault.

4.5 NeosVault web app CMS

This service is the core of NeosVault off-chain document storage management. It is designed to interact with the document metadata and storage, performing tasks like:

- Saving the document to the two storage locations: our headless CMS and IPFS
- Retrieving the document from the storage locations
- Saving and updating the encrypted document metadata

NeosVault maintains off-chain metadata for each document stored in our CMS, the off-chain metadata fields are:

- Document ID. Is the unique Document Identifier
- Document Hash. Is the hash generated from the encrypted version of the document.
- Merkle Hash Tree. Is the hash tree of the document's version history
- Access Merkle Tree. Is a hash tree of the public keys with access to the document.
- Encrypted document encryption key. Is an encrypted version of the encryption key for each document. This encrypted version of the key is encrypted for each user with access to the document, and we will have a number of encrypted keys equals to the number of user with access.

In the first phase of NeosVault development, we are implementing a Headless CMS to support the off-chain storage management. This component is planned to be removed in future implementations of the platform.

A headless CMS (Content Management System) is a content management system that provides a content authoring and management interface, but does not provide any presentation layer or front-end delivery mechanism.

Traditional CMSs are monolithic systems that combine both content management and content delivery. They are designed to manage the content and presentation layer together, which can lead to limitations in security, flexibility, scalability, and customization.

Headless CMSs, on the other hand, provide content through APIs (Application Programming Interfaces) or web services, allowing developers to build custom front-end applications using any technology stack, programming language, or framework of their choice.

This approach allows for greater flexibility and scalability, as content can be delivered to multiple channels, including websites, mobile apps, digital signage, and IoT devices. It also enables faster development cycles, as developers can work in parallel, without waiting for content authors to complete their work.

NeosVault is also implementing the IPFS storage, this integration is designed to provide a decentralized storage layer that can be used for replication of the encrypted documents.

4.6 Zk Document Proof

This NeosVault component is designed for the internal management of the zk Smart Contract used to save and update the on-chain storage. This component interacts with the zk smart contract and will generate the transactions and validations needed for the on-chain metadata management.

NeosVault uses Mina Protocol zk Smart Contracts developed using the SnarkyJS SDK, this smart contracts are executed off-chain and work as described in the Smart Contracts section of this document.

4.7 Smart Contracts notarization

Each one of the documents stored in our off-chain storage, will have a generated hash which will be notarized to the Mina blockchain using the zk smart contract.

The document's metadata will be notarized using Mina Blockchain, the stored On-chain metadata fields are:

- Document ID. Is the unique Document Identifier
- Document Hash. Is the hash of the encrypted version of the document.
- User Address. Document's owner public key.
- Access root hash. Is the root hash of the access merkle tree.

NeosVault

5. Zk Document Management & Sharing

NeosVault is a zero-knowledge document management and sharing system, it allows users to store, manage, and share documents without revealing any sensitive information to the system or any other third-party. In other words, the system operates on the principle of "zero-knowledge proof" where only the users who have access to the document can read or modify it.

To achieve this, the system uses encryption and decryption mechanisms, along with advanced cryptographic protocols, to ensure that only authorized users can access the documents. The encryption process involves converting the documents into a code that cannot be deciphered without a unique key. This key is only known to the user who created the document and the users in the NeosVault access list for the document, and is not revealed to NeosVault or any other users.

The decryption process is similarly secure, with NeosVault verifying the user's identity before providing access to the encryption key of the document. The system does not store the user's password or private key, making it impossible for any other third-party to access the document.

To enable sharing, NeosVault uses secret sharing to allow multiple users to access the same document without revealing the encryption key if the user don’t have the private key related to the public key used for the encryption of the document’s encryption key. Secret sharing is a method for distributing a secret among multiple parties, such that only authorized subsets of the parties can reconstruct the secret.

NeosVault as a zero-knowledge document management and sharing system allows users to store, manage, and share documents without revealing any sensitive information to the system or any other third-party. NeosVault uses encryption and decryption mechanisms, along with advanced cryptographic protocols, to ensure that only authorized users can access the documents.

5.1 Wallet based CMS authorization

Wallet-based authorization mechanism is a security mechanism used in decentralized apps (dApps) that leverages a user's digital wallet to authenticate and authorize transactions on the blockchain. NeosVault uses this authorization mechanism to authenticate the users and to approve and sign the transactions that the users sends to the Mina blockchain. NeosVault is a zkApp that allows the users to sign transactions using their Auro wallet (browser extension).

Once the connection is established in NeosVault CMS zkApp, NeosVault can use the wallet to authenticate and authorize the user's transactions.

The wallet-based authorization mechanism provides several benefits, including increased security, privacy, and control. Since the private keys are stored in the user's wallet, the risk of hacking or theft is significantly reduced. Additionally, users have greater control over their assets, as they do not need to rely on a third-party provider to manage their private keys.

Wallet-based authorization mechanism is a security mechanism used in dApps that leverages a user's digital wallet to authenticate and authorize transactions on the blockchain. The mechanism provides increased security, privacy, and control to users.

5.1.1 Connection to Auro Wallet

In a wallet-based authorization mechanism, a user's private keys are stored securely in their digital wallet, Mina Blockchain uses the Auro Wallet to sign transactions. The private keys of the accounts are used to sign transactions on the blockchain.

To use the NeosVault cMS zkApp that utilizes wallet-based authorization, the user must first connect their wallet to the zkApp user interface. This connection is established through the Auro browser extensions.

Available as a browser extension and as a mobile App, Auro Wallet has multiple languages version, perfectly supports Mina Protocol and is completely open-source. With it, the users easily send, receive or stake MINA, and view the transaction records anytime. It also supports sign zkApp transactions.

5.2 Upload

In NeosVault, the document upload process involves securely encrypting the document and uploading it to the off-chain storage including our CMS and IPFS, without revealing any sensitive information to NeosVault.

The upload process involves the following steps:

1. **Document Encryption**: The user encrypts the document as described in the Encryption an Decryption section of this whitepaper.
2. **ZK Proof Generation**: The user generates a zero-knowledge proof (ZKP) that demonstrates the authenticity of the encrypted document and the encrypted key without revealing the contents of the document or the key. The ZKP is generated using mathematical algorithms and protocols, such as zk-SNARKs.
3. **Upload to CMS**: The user uploads the encrypted document, the encrypted key, and the ZKP to the CMS. The CMS stores the encrypted document and the ZKP, but does not have access to the encryption key or the contents of the document.
4. **Access Control**: The user can specify access control rules for the document, such as who can view, modify or archive it. The access control rules are stored on the NeosVault CMS.
5. **v**: When a user requests to view the document, they must provide a valid proof that they are authorized to access the document. The proof is generated using the encryption key and the ZKP, without revealing the contents of the document or the key. NeosVault verifies the proof and decrypts the document, without having access to the encryption key or the contents of the document.

The document upload process in a zk content management system involves securely encrypting the document and generating a ZKP to prove its authenticity without revealing any sensitive information to the system or any other third-party. The encrypted document, encrypted key, and ZKP are uploaded to NeosVault, and access control rules are specified for the document. When a user requests to view the document, they must provide a valid proof, and NeosVault verifies the proof and decrypts the document without having access to the encryption key or the contents of the document.

5.3 Update

NeosVault allows for secure and private storage and management of content, where the user has complete control over their data and information.

The document update process involves the following steps:

1. **User authentication**: The user needs to authenticate to access the zk CMS app and make updates to the document. This involves authentication using the Auro wallet.
2. **Access control**: Once the user is authenticated, NeosVault needs to ensure that the user has the appropriate role and permissions to access the document and make changes.
3. **Retrieval of the document:** NeosVault retrieves the current version of the document from the secure off-chain storage, using the decryption key provided for the user.
4. **Updating the document**: Once the document is retrieved, the user is able to make updates to the content as required. Any changes made to the document is encrypted and stored securely in the system, ensuring that only the user can access the updated content.
5. **Verification and synchronization**: After the document has been updated, NeosVault verifies the integrity of the updated content and synchronize it with the CMS. This ensures that any changes made to the document are consistent across all instances of the content.
6. **Encryption and storage**: The updated document is encrypted and stored securely in the off-chain storage that are the CMS and IPFS, generating a new encrypted version of the document and the corresponding hash and metadata.
7. **Notarization**: The updated hash and metadata is now updated on the on-chain storage using the smart contract and the Mina blockchain.

5.4 Share

In NeosVault, document sharing involves securely and privately sharing content with other users while maintaining control over who has access to the content.

Here are the steps involved in the document sharing process:

1. **User authentication**: The user needs to authenticate to access the zk CMS app and make updates to the document. This involves authentication using the Auro wallet.
2. **Access control**: Once the user is authenticated, they can define the access control policies for the document they wish to share. The user can decide who can access the document, and what level of access they have (read-only or read-write).
3. **Encryption and decryption**: The content management system will encrypt the document using an encryption key. The system will then generate a unique encryption key for each user that is granted access to the document.
4. **Sharing the document**: The user can then share the document with other users, NeosVault will provide them with the unique encryption key generated by the system.
5. **Accessing the document:** NeosVault notifies users the invitation to collaborate on a document sharing a link, the users can use it to access the document in the content management system. The system will decrypt the document using the user's private key and the unique encryption key generated by the system. This process ensures that only the authorized users can access the content and that the content remains private.
6. **Revoking access**: The user who initially shared the document can revoke access at any time. The content management system will delete the encryption key associated with the user whose access has been revoked, ensuring that they can no longer access the document.

The document sharing process in a NeosVault involves encrypting the document with an encryption key, generating unique encryption keys for each authorized user.

5.5 Transfer Ownership

In NeosVault, ownership transfer of a document involves securely and privately transferring control of the content to another user.

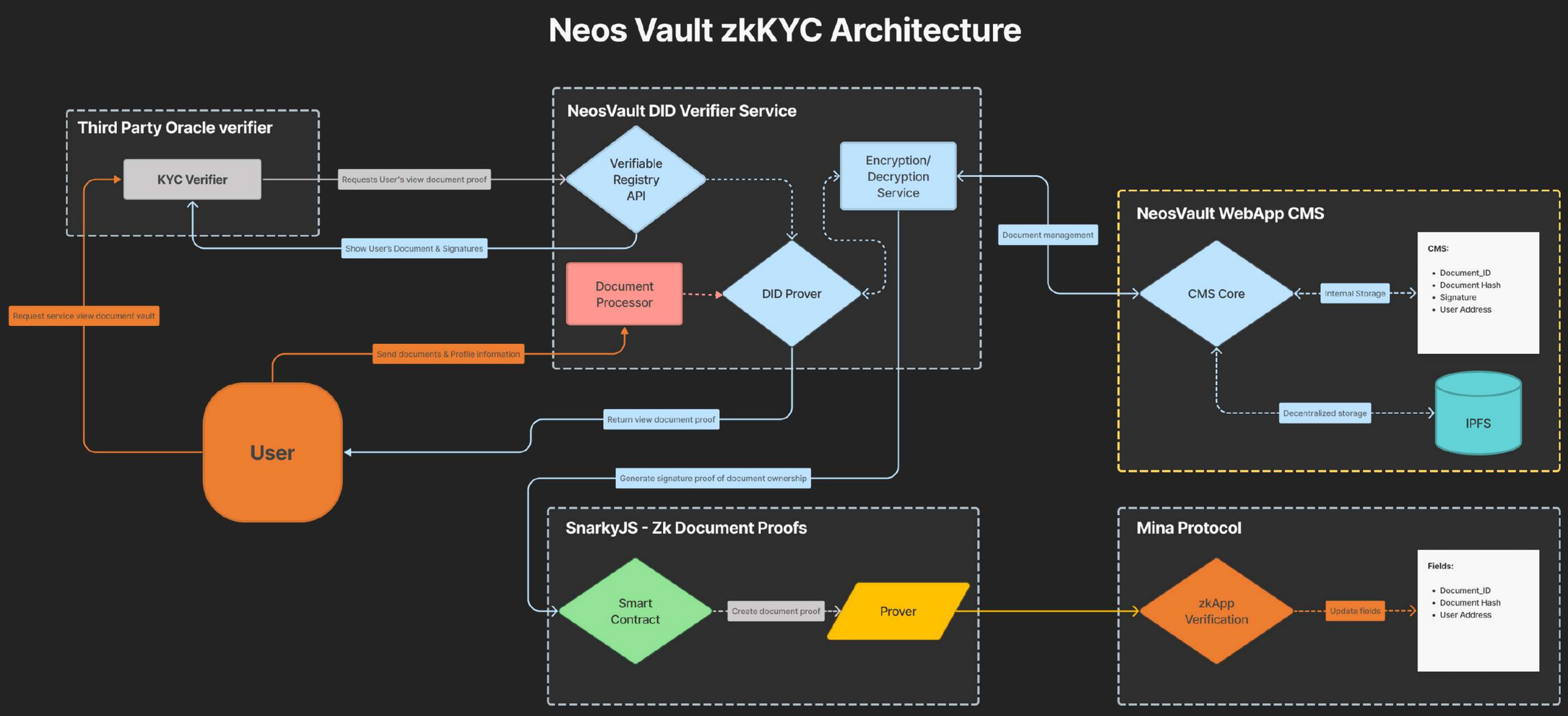
Here are the steps involved in the ownership transfer process:

1. **User authentication:** The user needs to authenticate to access the zk CMS app and make updates to the document. This involves authentication using the Auro wallet.
2. **Access control:** Once the current owner is authenticated, they can modify the access control policies for the document to transfer ownership to another user. The current owner must remove their access to the document and grant access to the new owner.
3. **Encryption and decryption:** The content management system will encrypt the document using an encryption key. The system will then generate a unique encryption key for each user that is granted access to the document.

1. **Transfer of ownership**: The current owner can transfer ownership of the document, NeosVault will require signing a transaction and notarizing the change in the mina blockchain.
2. **Accessing the document**: When the new owner receives the notification email with the access link, he can use it to access the document in the content management system. The system will decrypt the document using the new owner's public key and the unique encrypted key of the encryption key.
3. **Verification and synchronization**: The content management system will verify the integrity of the updated content and synchronize it with any other copies of the document that exist in the system. This ensures that any changes made to the document are consistent across all instances of the content.

In NeosVault, the ownership transfer process in a zk content management system involves re-encrypting the document generating a unique encryption key and generation a new encrypted version of the encryption key for the new owner and all the users in the access list.

6. Advanced Features



The NeosVault development has some advanced features planned to be developed in a future phase of the development.

6.1 DiD Verifier Service

This component of the advanced architecture helps to implement a verifier service for decentralized identities.
It includes: a verifiable registry api gateway to process external requests to validate users credentials or functions.

6.1.1 Verifiable Registry API

This is a service that will enable an API gateway to interact with the User's identity external validators.

Public API services that verify user off-chain legal identity, generates a real world signature for document signing & share document visualization proofs for the KYC oracle verifier.

6.1.2 DiD Prover

This service is an orchestrator to perform activities like:

- Validates user's document
- Validates user's identity
- Generate document view proofs
- Generate Identity proofs

6.1.3 E-Signatures

Each user of the NeosVault will have the option to generate E-Signatures, offering an option to the classical restaurant.

Electronic signatures, or e-signatures, can be a helpful tool for signing documents in a more efficient and convenient way. Instead of requiring a physical signature on a paper document, e-signatures allow individuals to sign documents electronically using a variety of methods, such as a typed name, a scanned image of a handwritten signature, or a digital signature.

Here are some ways that e-signatures can help with signing documents:

Convenience: With e-signatures, you can sign documents from anywhere, at any time, as long as you have an internet connection. This eliminates the need to physically sign documents in person, which can be time-consuming and inconvenient.

Security: Many e-signature tools use encryption and other security measures to protect your signature and the document from unauthorized access and tampering. This can help ensure that your signature is secure and that the document is legally binding.

Efficiency: E-signatures can save time by allowing you to sign documents quickly and easily, without the need for printing, scanning, or mailing. This can be especially helpful for businesses and organizations that need to process large volumes of documents.

Legality: In many countries, including the United States and the European Union, e-signatures are legally recognized as valid signatures for most types of documents, as long as certain requirements are met. This means that e-signatures can be used for a wide range of legal agreements and contracts.

Overall, e-signatures can be a useful tool for signing documents, offering convenience, security, efficiency, and legality. However, it is important to choose a reputable e-signature tool that complies with relevant legal requirements and industry standards to ensure that your e-signatures are legally binding and secure.

6.2 ZkKYC with self sovereign Identity documents management

6.2.1 Upload ID Document

For his phase, the users will have the functionality to update personal documents that will help them to support their identities using the documents.
NeosVault will validate the documents and ask the users to follow the KYC process to validate this uploaded documents.

6.2.2 Authorize document access

For this stage, our solution for zk KYC will enable the users to share and authorize anyone to their documents.

6.2.3 View and Read User’s document

The final users can permit some external providers to read their information in their documents.

In a decentralized identity context, document sharing refers to the ability to securely and selectively share personal identity documents with other parties without the need for a central authority to verify and manage access.

Decentralized identity systems use blockchain technology to create a distributed ledger that stores identity information, which is owned and controlled by the individual. With this system, individuals can control their personal identity data and decide who has access to it.

When it comes to document sharing, individuals can choose to share specific identity documents with a particular party or parties. For example, if an individual wants to prove their age to a liquor store, they can share a digital copy of their driver's license, without revealing other personal information.

NeosVault

This process of document sharing is done through a decentralized identity wallet, which acts as a secure digital container for all of an individual's identity information. The wallet is accessible only by the individual, who controls the private keys that allow access to the information.

Document sharing in a decentralized identity context provides several benefits. First, it gives individuals greater control over their personal identity data, reducing the risk of identity theft and fraud. Second, it allows individuals to share only the necessary information required for a particular transaction, reducing the risk of data breaches and protecting privacy. Finally, it eliminates the need for intermediaries such as government agencies, banks, and other identity verification services, reducing costs and improving efficiency.

However, as with any technology, decentralized identity systems have their own set of challenges and limitations, such as interoperability, scalability, and standardization. It is important to carefully evaluate the technology and its implementation to ensure that it meets the specific needs of an organization or individual.

6.2.4 Update ID Document

In a decentralized identity context, document identity update refers to the process of updating personal identity documents in a secure, verifiable, and decentralized manner.

Traditionally, updating personal identity documents, such as a driver's license or passport, requires an individual to visit a government agency, provide updated information, and go through a verification process. This process can be time-consuming and often requires the individual to physically visit the agency.

Using NeosVault, the users can update their personal identity documents in a more efficient and secure manner using a decentralized identity wallet. The wallet acts as a secure digital container for all of an individual's identity information, including updated documents.

When an individual updates their personal identity document, such as a new address or name change, they can update it in their decentralized identity wallet, which will then automatically update the distributed ledger. The distributed ledger, which is maintained by multiple nodes in a decentralized network, ensures that the updated information is accurate and verified, without the need for a central authority.

Updating personal identity documents in a decentralized identity context provides several benefits. First, it gives individuals greater control over their personal identity data, allowing them to update their information more easily and quickly. Second, it eliminates the need for intermediaries, reducing costs and improving efficiency. Finally, it provides a more secure and tamper-proof method of updating personal identity information, reducing the risk of identity theft and fraud.

However, it is important to note that decentralized identity systems are still in their early stages of development, and there are challenges and limitations that need to be addressed, such as interoperability, standardization, and adoption. It is important to carefully evaluate the technology and its implementation to ensure that it meets the specific needs of an organization or individual.