Skynet + OneId

Self Sovereign Identity System

This document provides an overview of Self Sovereign Identity System designed to tackle the issues faced by existing systems and solve them in the best possible manner. In order to gain an understanding of Identity Management, Self Sovereign Identity and why the following system has been proposed, it is recommended to give the Identity 101 a read. That document highlights research on the meaning of Identity, analysis of how current Identity Systems work and why they fail. An overview of decentralised systems implementing the concept of Self Sovereign Identity Systems are also discussed and the logic behind the following proposed system is explained.

Skynet

SkyNet is a highly secure distributed network that acts as a repository for issuers and well known identities. It is run by a single entity of the same name that functions as a cooperative society. It stores cryptographic keys, self declared claims and an index of all the claims issued by the respective issuers. The purpose of SkyNet is to provide the root of trust to the verifiers regarding the authenticity of any digital claim that is shared by an entity.

It provides a fast and scalable method of sharing and distribution of claims by well known identities. The high cost of participation in the network prohibits interference by malicious entities. It promises complete establishment of trust by any entity through the verification of members of the network and their published relationships and claims. SkyNet provides benefits to both the members as well as any other identity network in sharing digital credentials. SkyNet can not only act as a replacement to the current PKI system but also further add to its functionality by allowing for the exchange of Digital Verifiable Credentials.

SkyNet can be used by various organisations and entities to solve their user identification issues. Requests of authorisation can be redirected and handled by the network thus offloading organisations from the tedious and important task of identity verification and maintaining the security of that data. SkyNet is supposed to cater to a select audience of members due to its strict requirements and narrow set of objectives. In hindsight, these properties are what make SkyNet all the more reliable and trustworthy for large scale organisations and entities engaged in issuance and distribution of credentials. It provides a risk free alternative as opposed to current systems. SkyNet is supposed to provide a central nodal point of absolute truth regarding identities and claims. It is first and foremost designed to establish validity in the digital domain and eventually replace all physical credentials.

Objectives

SkyNet is supposed to be a closed network with limited functionality. The purpose is to take a barebones approach for maximum security. The core objectives of SkyNet can be summed up as follows

1. High Cost of Participation
   The network is not intended for every entity, that is the means of registrations and subsequent enrollment on the network is highly limited and done after intense scrutiny. The exact methods of how this can be achieved will be explained in further sections. The reasoning behind this approach is to limit malicious entities from enlisting themselves on the network.

The purpose behind the construction of SkyNet is for those entities to enrol which themselves have incentives to join. For example, entities which either want themselves to be recognised on the internet or are engaged in the issuance and exchange of identity and claims will have a lot to lose by not joining the network.

2. **Root of Trust**
   SkyNet should act as a single root of trust. The exchange of claims, keys and data that is referenced through SkyNet should be held as the universal truth. SkyNet does not delegate the functionality further to any other entity. This means that SkyNet manages the whole repository by itself. This should not be confused with how entities are provided membership on the network. The exact process of member admission on SkyNet will be taken up in a subsequent section.

3. **Secure**
   The network represents a trusted central repository, any compromise can undermine the functionality of the network. The ultimate purpose of SkyNet is to provide trust and assurance regarding the verification and authentication of entities online. This also expands to include the basic cryptographic constraints that the network will establish to prevent any form of compromise.

4. **Scalable and Fast Access**
   SkyNet caters to a select membership of entities which are likely to engage in large scale distribution of claims and certifications. This can result in huge amount of redirection requests landing on the network regarding any form of authorisation or verification. SkyNet intends to replace all forms of online authentication mechanisms which includes PKI as well as username and password based systems. This will help organisations to offload these crucial yet highly demanding tasks on the network and focus more on what matters for their business.

   This would directly translate to huge amounts of traffic. Assuming the network does witness widespread adoption, the resulting traffic hits can theoretically consist of all forms of authentication requests. Any form of downtime can severely cripple any form of online functioning as users will not be able to log in. Hence it is imperative that the system is designed to scale and respond to variations in traffic and prevent any form of downtime

Membership

SkyNet will initially be bootstrapped by well known large scale entities which shall include government institutions and private organisations. Eventually, the aim is to charge the members a one time admission fees along with a subscription service in tune with their usage of the network and its services. In order to make it easier for entities to join and

participate in SkyNet, functionality for existing members to sponsor the admission of a new entity would be provided for. However, the network would monitor this distribution of sponsorship and delegated membership. If it is found that a sponsored entity is engaging in malicious behaviour or the sponsor has been acting against the rules of the network, SkyNet will reserve the right to suspend these entities from operating on the network. This ensures that SkyNet stays true to its objective of providing a trusted central repository by continuously removing bad actors.

Control

The inspiration to develop SkyNet was drawn from the success of Society for Worldwide Interbank Financial Telecommunication(SWIFT). SWIFT is a cooperative society established by various financial institutions. It provides a network enabling the member financial institutions to send and receive information regarding financial transactions in a safe, secure, standardised and reliable environment. The success of SWIFT demonstrates the fact that organisations can cooperate to work on systems and processes that can significantly benefit all the participants. While joining an open decentralised network asks for a lot from these institutions to give up, working together to form a cooperative organisation with exclusive support presents an option that is viable and practical. This was the reasoning behind developing *SkyNet*.

Like SWIFT, SkyNet is supposed to be a cooperative society formed by various issuers, primarily ranging from Government backed functionaries to financial and private organisations engaged in generation and distributions of claims on a large scale. This ensures that the major players in the field of identity generation and verification are incentivised for moving to a new platform. The exact mechanics of this cooperative society are left for future exploration. However, networks like SWIFT shall provide insight on how to go about the same.

Benefits

SkyNet promises to bring about a sea change in how identity and claims are shared online. A successful deployment of SkyNet can help

1. Eliminate the need for Public Key Infrastructure based on Digital Certificates.
2. Introduce the added functionality of exchanging digital verifiable claims
3. Make it easier to make and share claims
4. Allow the management of claims to be done in a safe, secure, transparent and fast manner.
5. Make it easier for other identity systems to harness SkyNet for resolving well know identities and their claims
6. Reduce the burden from the organisation regarding the management of identities and authorisation.

Architecture

In order to explain how SkyNet functions we will first explain the attributes that will be associated with every entity that enrols on SkyNet

1. ***Assigned a Unique Identifier***

   The Unique Identifier will be generated centrally by SkyNet. As the name suggests, no two entities on the network will have the same identifier. This identifier will be used for any operations on the network involving the respective entity.

2. ***Assigned a Public/Private Key Pair***

   Every entity will be assigned a pair of asymmetric keys. Any authorisation operation by the entity will be performed by digital signatures. Authentication will be done using the public key.

   The private key will be generated once and then destroyed. SkyNet will not store the private key on its servers. There will be no way to recover the key from SkyNet. The security of the private key is the responsibility of the holding entity. It is recommended that the private key is sharded and distributed. Cryptographic techniques such as BLS allow signing from shards themselves without requiring the whole key.

3. ***Claims generated and controlled by the entity***

   Everything in SkyNet is in the form of Claims. Any information that an entity wants to be made public about itself, for example, its address, phone number, or other information can be made into a claim. There are 2 key advantages of using this

   1. It brings uniformity in the architecture and processing methods
   2. Every claim can have a method. This allows for these entities to automate the reading of any kind of data through the claim methods.

   Claim are explained in depth in subsequent sections.

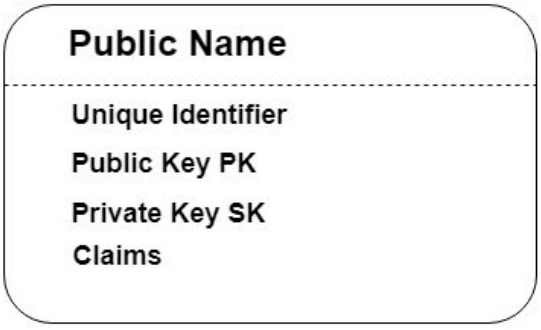

Figure 1. Overview of the contents of an entity in SkyNet

Public Repository

The endpoint of any identity enrolled in SkyNet will be referred to by its identifier. In order to view the information made public by an entity, simply type in the id. The request will be redirected to *SkyNet/view/identifier_of_entity.* The key point of this service is for well known identities to share their credentials as well as issue the same in a trusted manner. In order to fulfil its purpose of a trusted repository, it will maintain a public database of the participating entities. The repository

1. Will be accessible to any user through a web based interface
2. Will implement search based indexing, allowing users to quickly search through the database for information.

For example, suppose a bank operates three websites under different names. If a user wants to verify whether a given website is operated by the bank, it can simply search SkyNet Public Repository to obtain the public claims generated by the bank, where it can find the names of the websites currently operated by the bank. However, one thing that should be made clear at the outset is, that **no entity will be forced to make any of their data points public. It is entirely possible that an entity can choose to keep certain or all attributes private.** This means, that even though the verification and authorisation features will function through API calls, the information itself would not be visible on the public interface itself. Figure 2 here provides an overview of the architecture of SkyNet. We will be explaining the functioning of each segment in the following section.

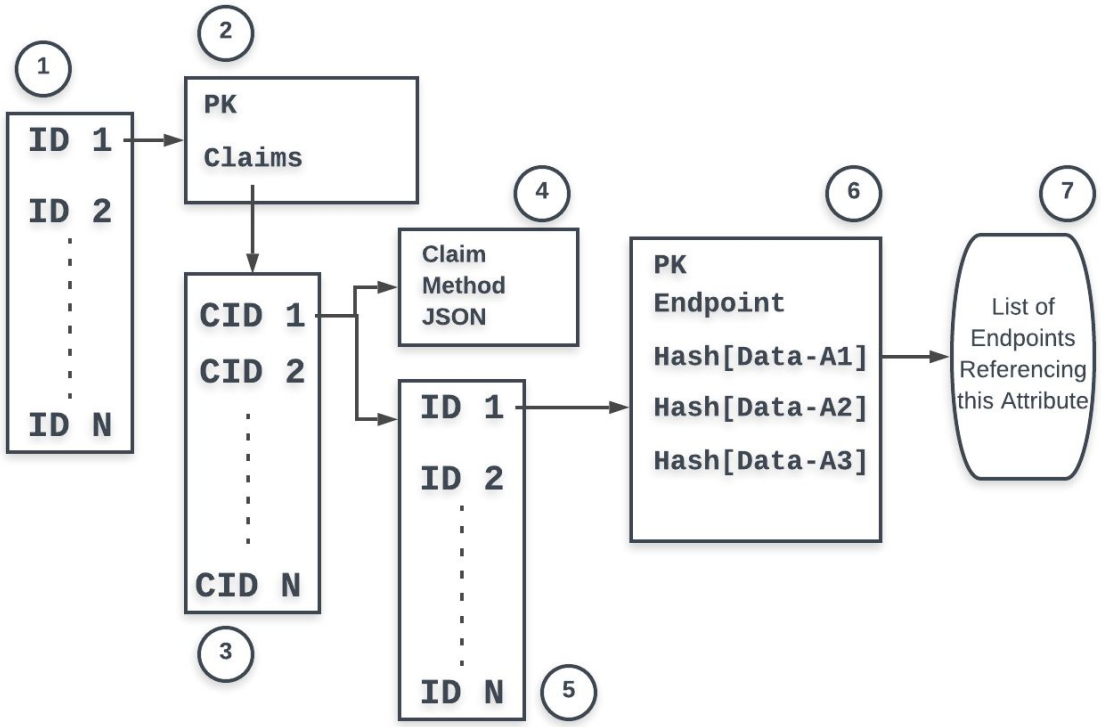

Figure 2. Detailed Overview of Architecture of SkyNet

Label 1

This is the first layer in the SkyNet repository and is used to store the list of all the entities registered on SkyNet. The exact methodology for generating these identifiers is yet to be finalised but is likely a task to be handled by the central server. Support for DID should be a viable option given it is an open standard.

Label 2

The contents of an entity that will be stored in SkyNet have already been discussed in a prior section. The claims that are stored by the entity will be indexed by their respective identifiers. Similar to the case with entity id's, the exact methodology for generating these identifiers is yet to be finalised but is likely to be handled by the central server

Label 3

The index of all the claims belonging to a particular entity in the network. At this point, it is imperative to establish how claims work. A claim as explained earlier is a set of attributes generated by the issuer for some purpose. The issuer then distributes the claim by authenticating the data belonging to the holder against the given attributes. In SkyNet, every claim will have two parts,

A. *Claim Method: Label 4*

   The claimed method is a JSON document that contains the respective attributes as the key value. The properties for every claim attribute will be outlined, along with any other required information that is deemed necessary to interpret the given claim. It is the responsibility of the issuing entity to generate the method, for a lack of it thereof can hinder the use of its claim.

B. *Claim Data*

   The claim data here represents the data belonging to the holder. At the time of issuance, the issuer will generate hashes for the data and store them in the segment Label 6. The data itself will never be stored in the central servers of SkyNet to preserve the privacy of the users. The exact procedure for claims exchange will be taken up in the next section.

Label 5

The identifier index of all the entities to which the issuer has granted this particular claim.

Label 6

Every holder of a claim generated by an issuer will be holding three attributes,

1. The Public key of the holder.
2. The Endpoint of the holder, which also represent the holders' identifier.
3. Hashes of the claim data.

Label 7

For every hashed piece of data, there will be a list of endpoints of all the entities that are referring to this hash. **This is an extended functionality primarily required to provide claim revocation facility. This is not a necessary requirement** and its application can depend on the level of functionality requested by the issuing/verifying entity. For example, use cases demanding that updates regarding claim revocation be immediate can make use of this feature.

Generation, Exchange and Verification of Claims

Here we are going to explain the core functionality SkyNet, that is functioning of claims. In order to facilitate the explanation, we shall take up an example. Given a user Alice, the Transportation Department and a bank. The claim in question here is the Driving Licence.

Claim Generation

The Transportation Department will first create a claim for the Driving License under its well known identity in SkyNet. This will involve specifying the claim method. Once the claim method is generated, the process for an entity to get the claim are shown in Fig 3

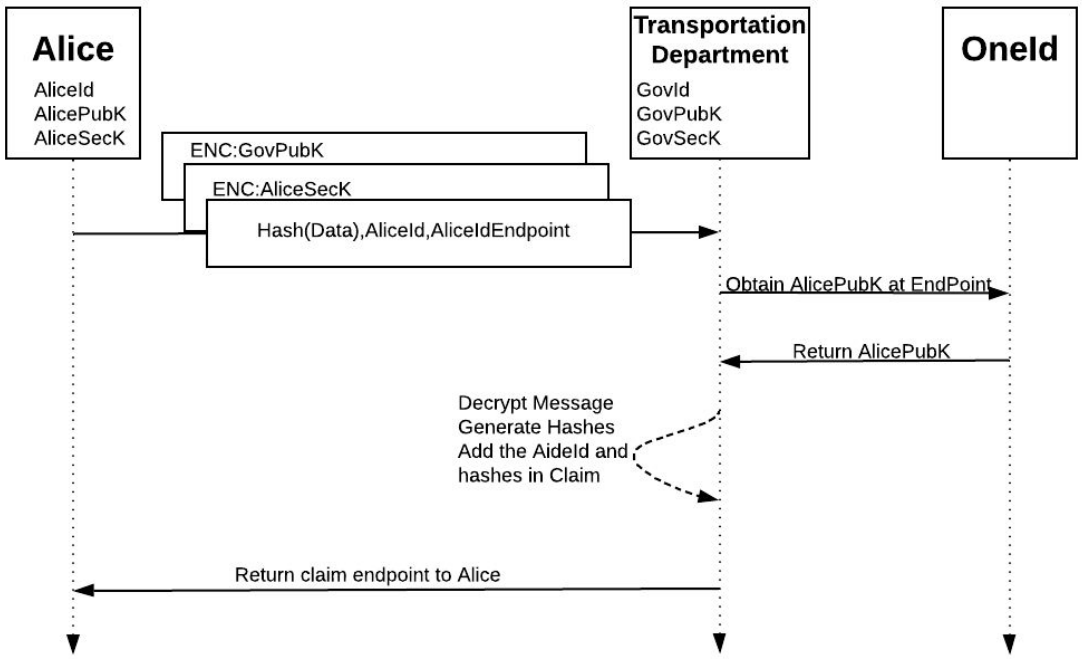

Figure 3. Obtaining a claim through SkyNet

1. Alice will have the option to either create a new set of identifiers for her relationship with Transportation of Department or use an existing one.

2. Alice creates a message M containing the data required for driving licence. The message is encrypted twice, first with the private key of the associated identifier followed by the public key of the Transportation Department. Alice then sends {M,identifier,endpoint} to the Transportation Department endpoint.
3. The Transportation Department queries the endpoint for the given identifier and obtains the public key of the identifier belonging to Alice for this relationship. It then decrypts the message M. Through this step it is ensured that the identifier is indeed in control of Alice.
4. The Transportation Department generates hashes for the data provided. It then stores the identifier of Alice, alongwith its public key, endpoint and the hashed data in SkyNet.
5. The Transportation Department finally returns an endpoint to Alice. Alice can then attach this endpoint to her claim verification messages.

Claim Exchange and Verification

In order to share the claim with a verifier, in this case a bank, Alice needs to complete two steps. First is to set up a new relationship with the verifier, and the second is verification of the claim.

Setting up a relationship with the Bank

1. Alice sends the verifier {M,identifier,endpoint}, encrypted twice. The message M is encrypted twice, first with the private key of the associated identifier followed by the public key of the bank. The message itself can contain either the identifier or the endpoint.
2. The bank can then query the endpoint for the public key and decrypt to check whether the identifier is really in control of the participating entity.

Sharing the claim with Bank

1. The bank then creates a claim request, consisting of {nonce, issuerId, claimId, arrayOfRequestedAttributes} and then encrypts it twice, first with its private key followed by public key of Alice's bank identifier.
2. Alice creates a response message M consisting of {ENC(nonce),Alice TransportDepId,AliceTransportDepEndpoint,arrayOfReqData}, and encrypts it twice, first with the private key of the associated identifier followed by the public key of the bank. The nonce itself is encrypted with the private key associated with the AliceTransportDepId.
3. The bank upon receiving the request, contacts SkyNet at the mentioned endpoint
    a. The endpoint is invalid. This can be the case if the claim is fake or has been revoked.
    b. In the case of a valid endpoint, Bank uses it to obtain the associated public key using it to decrypt the nonce. If it matches with the claim request of the bank, then it is confirmed that the Claim is valid and controlled by Alice.

4. In order to verify whether the claim data is authentic, the Bank hashes the provided data, and simply queries the endpoint on SkyNet for a match.

In case Alice needs to only prove that she is indeed the owner of the claim, the array field will simply will empty and the bank will not need to perform step 4.

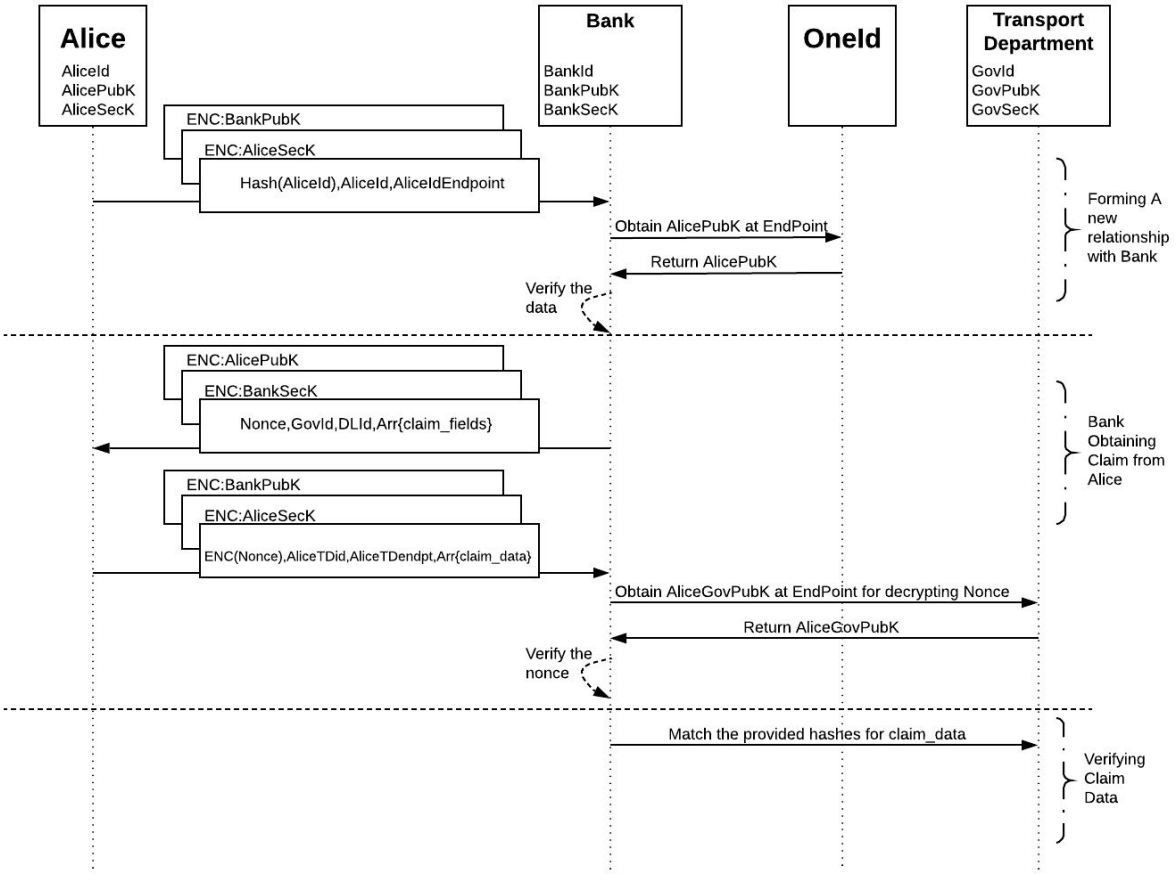

Figure 4. Steps for exchanging claims in SkyNet

API

Some basic API requests supported SkyNet for external Users. This is incomplete, and more interfaces will be added as the project progresses.

1. ***getPublicKey(id identifier)***
   Queries SkyNet for a particular id and return the respective Public Key for the given Identifier

   Return Value

RSA1024{Public Key} of Id if present
Error{Invalid Identifier} if identifier is incorrect/absent

2. ***getClaimMethod(id identifier, claimId claim)***
   Queries SkyNet for a particular claim to obtain its respective method. The function will return a JSON document giving information on the type of attributes being stored in the claim alongwith their properties. Developers can use this information to interpret the claim data that they receive.

   Return Value
   JSON{claim description} describing the claim method if claim and id are valid
   Error{Invalid Identifier} if identifier is incorrect/absent
   Error{Invalid Claim Identifier} if claim identifier is incorrect/absent

3. ***verifyClaim(idOfIssuer identifier, idOfHolder identifier, claimId claim)***
   Queries SkyNet for a particular claim belonging to an id given the id of the claim and its Issuer. SkyNet queries the list of identities stored as values for the given unique pair of {idOfIssuer,claimId}, and returns the public key of idOfHolder is present, error otherwise.

   Return Value
   RSA1024{Public Key} if idOfHolder is present in claim list
   Error{Invalid Claim} if idOfHolder is absent
   Error{Invalid Issuer Identifier} if issuer identifier is incorrect/absent
   Error{Invalid Holder Identifier} if holder identifier is incorrect/absent
   Error{Invalid Claim Identifier} if claim identifier is incorrect/absent

4. ***verifyClaim(idOfIssuer identifier, idOfHolder identifier, claimid claim, dataHashArr HashArray)***
   Queries SkyNet to verify the integrity of particular claim data belonging to a given holder for a particular claim belonging to an id given the id of the claim and its Issuer. The query contains an array of the hash value for the attributes belonging to the claim. The array is map containing the queried attributes as key and the hash of the data as the respective value. The returned Map contains boolean values.

   Return Value
   Map{[String]Bool} A[i] = True if hashes matched, otherwise A[i] = False
   Error{Invalid Identifier} if identifier is incorrect
   Error{Invalid Claim Identifier} if claim identifier is incorrect
   Error{Missing Claim Method} if claim method is absent

OneId

The ultimate goal of this project is to provide Self Sovereign Identity for every user. Constructing a system that truly achieves this faces considerable issues as we have seen in our critical analysis of the Sovrin network. In order for the system to be practical while staying true to its principle required distribution of services between a centralised and a decentralised system. As explained in the previous section, SkyNet is supposed to act as a trusted repository of identities for well known entities. The natural question that arises is, how do other users, such as individuals or small scale entities participate and make use of these features. Unless the users are provided with the ability to onboard the network, SkyNet would not be able to gain much functionality. The factors to be kept in mind for such a network include

1. The use of centralised service is suited for well known entities, however, for individual users, it can present complications that come attached to any other centralised service. There will have to be a lot of trust riding on the service and the question of manipulation and self sovereignty will be out of the equation.
2. This central service can be forced by governments for surveillance or be susceptible to offers by interested third parties to reveal information of the users. The trust of storing invaluable data in the hands of a single entity that transcends boundaries of countries can be hard to achieve.
3. The service needs to be free of cost, at least for some given basic set of functionality. Identity is universal, it is the same for every rich or poor person. Placing a barrier to participation can severely hamper the adoption of the network.

In order to fulfil these requirements, we propose a decentralised network, OneId. The ideology of OneId is to allow any user to join the network and be able to create and store their identity related in a trusted, secure and privacy preserving manner. This is achieved through the use of a permissioned ledger. The OneId ledger by itself is capable of functioning, however the addition of a network like SkyNet along with an independant data storage service can vastly improve functionality.

Objectives

The objectives of OneId can be summarised as follows,

1. Allow any entity to create and manage their identities in a decentralised manner.
2. Ensure the network is not subject to any form of censorship or centralisation.
3. Stay true to the principles of Self Sovereign Identity
4. Allow any user to create and share claims with any other user.
5. Ensure security and privacy of the maximum order.

Benefits

The OneId system is supposed to be a permissioned ledger in order to ensure that it scales accordingly as the number of users and the subsequent number of requests on the network increase.

In order for nodes to voluntarily participate in the permissioned ledger for others to use, the nodes will be rewarded with native currency. The mechanics of how the currency is generated, regulated and distributed is yet to be dictated. However, there are various. A proposed method is to place limit on the number of dids, claims and overall storage for wallets a user will be exempted on the network. The storage services can charge users for any further added functionality or usage. The earning from the storage services will have to be made through the native currency, and a cut will be taken from the earnings to fund the nodes running the permissioned ledger. The incentive mechanisms can work well here as the storage services cannot simply ballon their profits if the permissioned nodes are not paid their dues threatening the functioning of the network. In fact, it is highly likely that most storage services might themselves operate a few nodes in the ledger. The overall benefits of using a Self Sovereign Identity system have already been covered in previous sections, which are expected to be received from the application of OneId.

Architecture

The OneId ecosystem is represented by Figure 5. Every user registered on the OneId network shall make use of the Storage Services to store their wallets and be able to manage them remotely from any device. This is not a necessary requirement and users will be given the option to operate wallets on their own. Most of the claims will be issued through SkyNet given the presence of well known identities in its trusted network. Users of OneId will be able to transact these claims as both SkyNet and OneId are constructed to be highly interoperable. Ensuring standardisation will be important for the success of both networks.

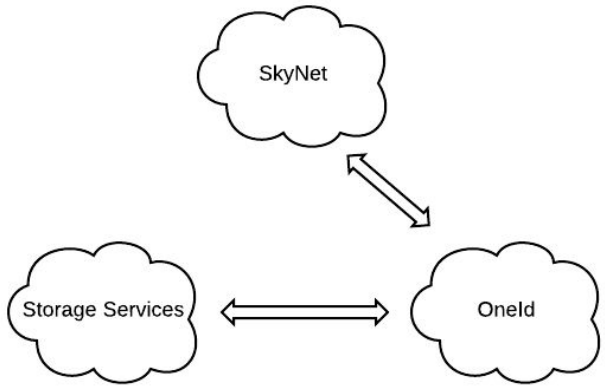

Figure 5. The Identity Ecosystem of SkyNet, OneId and Storage Services

OneId will be making use of multiple independent blockchains to run the self sovereign identity system. The reason for keeping separate blockchains is to ensure scalability and a clear demarcation between rules. The following section will explain how Identities in OneId will function and how their information will be stored.

Creation of Wallets

The wallet is the core functionality of OneId. All the information belonging to an entity will be stored in the wallet. The Wallet Chain is shown in figure 6. The functioning of the wallet has been divided into two types which are described below

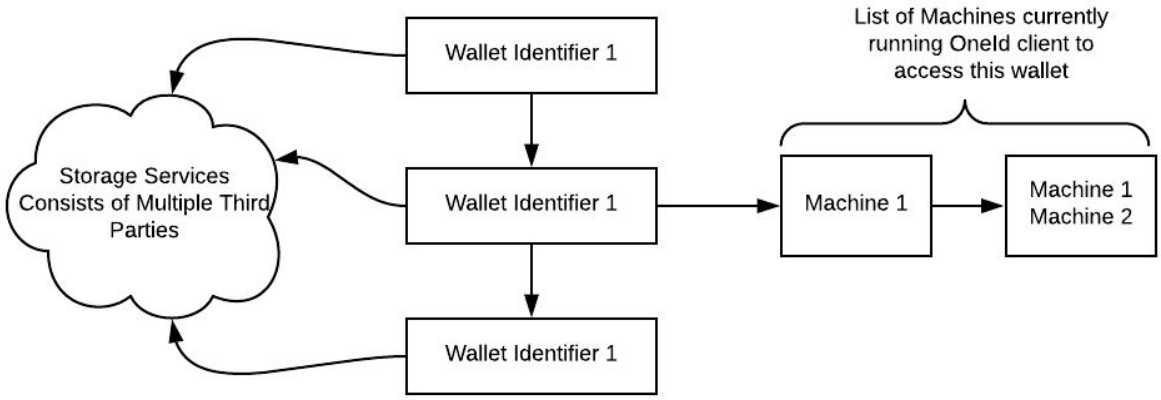

Figure 6. Overview of the Wallet Chain

Master Wallet

The master wallet is the main storage area for all related identifiers and data belonging to the controlling entity. The first time a Master Wallet is created, three attributes will be generated,

1. *Master Wallet Identifier*: This is stored in the Wallet Chain and is used to identify the wallet. The creation of a block on the Wallet Chain involves the subsequent creation of secured storage area in the Storage Service. The functioning of the storage service will be covered in a subsequent section.
2. *Controlling Keys*: A public private key pair will be created to control the wallet. The private key will be destroyed after its creation. The exact method of recovery is still left out but can be something along the lines of Sovrin which includes a paper based QR code or sharding and distribution amongst friends and relatives. The public key will be used to encrypt the backups on the Storage Service. In case of loss of data stored in the Device Wallets, the only way to recover the encrypted data will be to reconstruct the private key to decrypt the data of the Master Wallet from the Storage Service. The public key is stored in each of the Device Wallet

which is using this Master Wallet as the backup. The public key is not stored on the chain.

Device Wallet

The device wallets are used to store the identity related data locally on the machine. Any form of data created by an entity is stored within the wallet. The only other place where the data is transferred is to the Master Wallet linked with the given Device Wallet to provide backup, and that too is encrypted with the decrypting private key destroyed to ensure maximum security and privacy.

Every Device Wallet on initiation will complete the following steps,

1. *Set up an Authentication Method*: The method of authentication for logging into the device wallet can vary, however, OneId will define some basic guidelines to ensure that device wallets do not prove to be the weak link their security. The most widely used and preferred medium will be a mobile phone. Using authentication methods such as fingerprint scanning along with password protection can provide extra security.
2. *Device Wallet Identifier*: Create a unique identifier. The method of identification creation is not yet outlined, however using characteristics unique to the properties of the machine can prove to be useful.
3. *Controlling Keys*: A public private key pair will be created to control the wallet.
4. The last step will be determine either of the following options
    1. Create a Master Wallet
       In this scenario, the local client will initiate the creation of a Master Wallet according to the steps highlighted in the Master Wallet section. Once this is done, the public key along with the identifier will be registered on the Wallet Chain under the Master Wallet.
    2. Link to an existing Master Wallet
       This is the case when a user wants to access their wallet from multiple machines. This requires two things, a)authorising the new device wallet b)copying the local data from an existing device wallet. These steps will require the use of an existing device wallet. A possible chain of steps can be as follows,
        1. The new device wallet creates a QR code containing its public key and identifier.
        2. This is scanned using the existing device wallet. It then forms an encrypted communication channel with the new device wallet and sends the data encrypted with its public key.
        3. The existing device then adds the identifier and public key of the new device on the Wallet Chain under their common Master Wallet thus authorising the new device wallet.

Maintaining consistency between multiple device wallets without decrypting the data stored in the Cloud can present some problems. Though the consideration of storing private key for the Master Wallet in every device wallet is very viable from deployment, ease of construction and operational purposes, the security is far less than the proposed model. Methods of consistency maintenance similar to versioning software can provide some insight on how to approach this problem.

A few other important things to note here is that a single user can create more than one wallet for greater security. Although managing identifiers across various wallets can prove to be cumbersome. Also, the user can also discard the option to store their backups in a Cloud Storage and take the responsibility of backup and recovery.

Creation of Identifiers

Any entity that joins OneId can create native Identifiers. The exact nature and functioning of these identifiers is yet to be finalised, but will likely take inspiration from Decentralised Identifiers Specification proposed by W3C as it shall allow users to generate identifiers in a decentralised manner. Every identifier will be added to this chain and hence will able to referred to through a public endpoint as *OneId/identity/identifier.* Figure 7 depicts the working of the Identifier Chain.

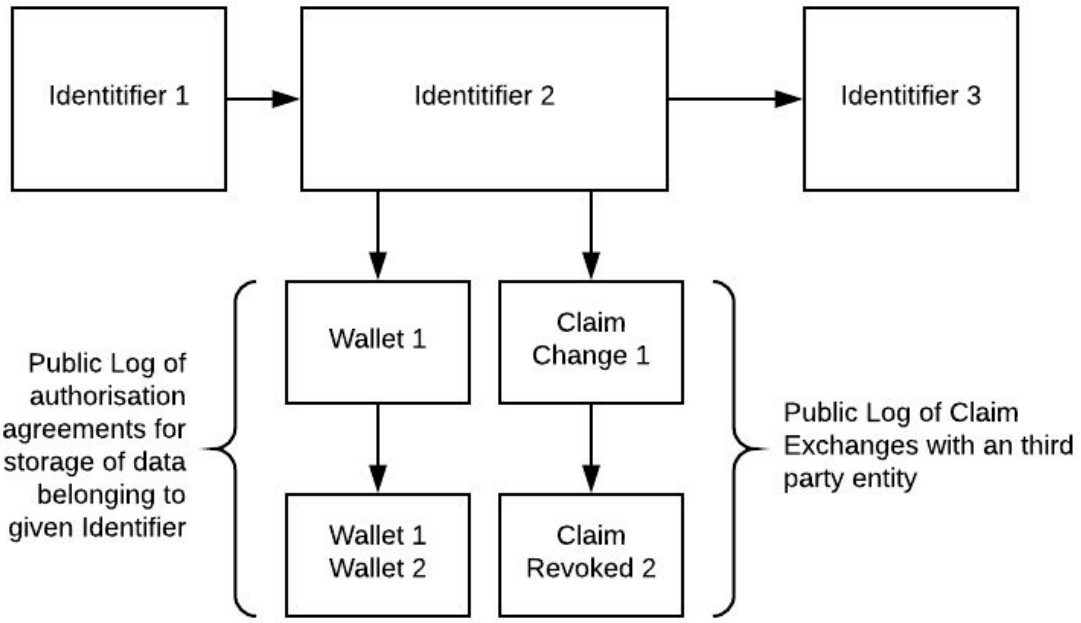

Figure 7. Overview of the Identifier Chain

For every identifier, we have the following set of properties,

1. A public private key pair. The private key is stored in the wallet and the public key is visible on the ledger.
2. A chain of blocks for authorisation records. The first block contains the wallet Id responsible for the creation of the given identifier. The same identifier can be controlled by multiple wallets or completely made to change ownership. The log makes sure the records are transparent.
3. Any messages relating to an entity engaged with this identifier through claims will add blocks to the claim message chain. For example, when a claim is created, claim data is modified, or claim is revoked, the changes would not be accepted until the block expressing the changes is added to this chain. This ensures non repudiation and a log of claim related activities concerning a particular identifier. This is not a permanent requirement, whether or not an entity wishes to engage in this depends on the use case. For example, financial transaction related claims would be better off keeping a public log.

The Wallet and Identifier Chain are sufficient for the general purpose of functioning and fulfils most of the requirements. The only thing to be covered yet is how will a user make and exchange claims if any. This is dealt with in the next section.

Claim Chain

The claims used in SkyNet are native to the system, and logs of changes to the claims is kept within the system. However it might be the case that transparency regarding these changes in the claims be kept public. To solve this problem, if the issuer knows that the public records will be required, it simply uploads it claim id on the claim chain. Any changes thereof in the claim in SkyNet will have to be first uploaded as log on the claim chain in OneId before being implemented in SkyNet.

API

Some basic API requests supported SkyNet for external Users. This is incomplete, and more interfaces will be added as the project progresses.

1. ***makeMasterWallet(Mid master_identifier, DWid_identifier, DW_Public_Key)***
   The local OneId client upon invoking makeMasterWallet will construct three attributes, {Master_Identifier, Master_Private_Key, Master_Public_Key}. The private key will be destroyed after the users chooses the concerned recovery method. The local client will then call a node on OneId and submit the transaction to the OneId network. A successful addition to the Wallet Chain will return the given endpoint to the local client.

   Return Value
   Wallet_Chain/Mid_identifier{endpoint} endpoint to Mid on Wallet Chain

Error{Unable to add} if block was not added

2. ***makeDeviceWallet(Mid master_identifier, Device_Wallet_identifier, Device_Wallet_Public_Key)***
   The local OneId client upon invoking makeDeviceWallet will construct three attributes, {Device_Wallet_Identifier, Device_Wallet_Private_Key, Device_Wallet_Public_Key}. If Mid does not exist, then makeMasterWallet is invoked first. Once the master wallet is initiated, the device wallet is added to the chain

Conclusion

The aim of this project is to implement identity and credential management through decentralised systems. To this end, various other implementations were studied and compared. Based on their analysis, we proposed a solution consisting of a combination of centralised and decentralised services. We highlight in detail on why this approach presents a favourable outcome and present the mechanics of its functioning. Workflows and data management are also presented to show real life use cases.

However, this current version of SkyNet and OneId has left a few of the functionalities incomplete. The most important being the implementation of wallet consistency and claims exchange on OneId. The incentive mechanisms for the permissioned ledger and storage nodes are also left out for further exploration to ensure that the system is practical enough to be used in real life. Overall, the combination of SkyNet and OneId presents an exciting opportunity in the field of self sovereign identity.