20 Decentralized Identity

In this chapter, we cover decentralized digital identity. Especially with the advent of blockchain, digital identity has taken the limelight as blockchain is seen as an enabler to transform the current digital identity paradigm, which suffers from various issues.

Along the way, we will cover the following topics:

- Identity
- Digital identity
- Identity in Ethereum
- Identity in the world of Web3, DeFi, and Metaverse
- SSI-specific blockchain projects
- Challenges

So, let's dive in.

Identity

Identity refers to the distinguishing features, personality attributes, physical attributes, and expressions that define an individual, entity, or group (someone or something). It is the unique set of characteristics that make an entity recognizable and distinct.

Identity can encompass different aspects, such as personal identity, social identity, cultural identity, and digital identity. It is an important aspect of our day-to-day lives and plays a fundamental role in shaping how individuals or entities interact with others around them.

An important aspect of identity is credentials. Credentials can be defined as an attestation by a relevant authority of an attribute associated with an entity. The attribute can be a personal attribute like age, name, or gender or could also be something that you have earned, such as a qualification. Credentials issued to individuals include a passport, driving license, license to kill, diploma/degree, and many others. An organization can be issued with credentials such as authority for them to operate in a certain jurisdiction, company registrations, building permits, and many more. Traditionally, all these credentials are paper based.

For example, a driving license is a card (paper) issued to you by the concerned authority once you have passed the driving exam. You present it as proof of your ability to drive to a police officer when necessary. Such a model is called a paper-based credential model, which is founded on paper documents issued by a concerned authority. The individual who is issued the paper document is the holder, the entity who issues the paper document is the issuer, and a verifier is an entity that verifies this credential to ensure its legitimacy and allow access to some resources. For example, to open a bank account, I may need to present my driving license as proof of my address and identity. These paper credentials contain some claims, e.g. your qualifications. These credentials are also expected to prove who issued the credential, who the holder of the credential is, and most importantly, that what is claimed is correct and accurate and that the claims have not been altered in any way. In short, the claim must be verifiable. However, in the real world, the forging of documents is quite common. Because this model is based on assumed trust between holders, issuers, and verifiers, it poses some challenges.

When an identity is represented electronically, it can be defined as a digital identity. We will discuss this next.

Digital identity

Digital identity can be defined as an electronic representation of a real-life entity, such as a human, machine, device, or organization. A digital identity can be represented by a citizen identification number, employee ID, host name, and in many other forms.

In other words, a digital identity is information that identifies an individual or entity in the digital world, e.g., on the internet.

Identity management can be defined as the process of creating, updating, deleting, and storing digital identity accounts and managing access to resources through the process of authentication and authorization.

There are several models of digital identity, including centralized, federated, and decentralized identity models. We'll explore these models next.

Centralized identity model

The centralized identity model is what we are most familiar with. It is the most widely used model with identifiers and credentials issued by centralized service providers and governments, for example, Facebook logins, Google logins, passports, citizen cards, and driving licenses. We can think of two variations of this model:

- **Independent centralized identity model**: This model represents paradigms where a single user holds a credential from a single service provider. There is a direct one-to-one relationship, and the issued credentials such as login and password can only work with a single entity from which they're issued. The user has to maintain individual credentials issued by each of the entities.
- **Shared centralized model**: This model is based on a paradigm where a central service provider maintains the credentials and they are shared across different service providers.

This model is the original internet identity model, also called the "account-based identity model." Here, a user establishes their identity by providing registration details to a service provider and creating an account with the service provider, e.g. a website.

There are, however, several problems with this model:

- It's centralized, so all your data resides on the service provider's side.
- You don't have any control over your data, how it's used, where it's used, when it's used, and for what purpose – even if there is regulation around it, it's not really under your control. What if the service provider is trying its best to serve its customers but a hacker manages to steal personal data from the service provider?
- The onus lies with the user to maintain their usernames and passwords for each service provider.
- Being centralized, they are subject to hacking and data breaches. Just do an internet search of "data breaches in 2023" or "recent data breaches" to gauge the magnitude of this issue.
- Every service provider has their own security policies around the management of accounts and can impose those policies on customers, including around the choice of password, its length, its usage, etc. While the intention is good here, in practice it becomes quite cumbersome for customers to manage their accounts.

- Basic account model – a username and password.

Federated identity model

The fundamental of the federated identity model idea is to enable a user to use the same account (username and password) and other credentials to sign on with multiple service providers. Members in a federation (hence the name federated identity) rely on each other or a central provider to authenticate their respective users and vouch for their right to use services. For example, once a user is authenticated using one member, the user doesn't have to log in again to another service provider with a separate username and password. The user can log in using the same credentials that were used to log in to the identity provider on other services in the network without having to enter their username and password again. Once the user is logged in to an identity provider, the user can use all services that are provided by service providers that accept credentials from the identity provider they are using.

The model operates with a central entity called an identity provider, which sits between the organization and the user. Moreover, once a user has logged in to a service provider using their credentials, if accepted by another service provider, the same credentials can be used to authenticate to other service providers. For example, logging in to another website using Google or another third-party identity provider is an example of a federated identity model. In this case, users authenticate themselves to the identity provider (Google) using their login credentials, and the identity provider issues a security token to the user. The user can then use this security token to access other websites or applications without creating a new login or sharing their personal information with the other websites or applications.

This approach allows for a seamless user authentication process while allowing the website or application to maintain its security domain. It also reduces the need for users to remember multiple passwords and login credentials. In addition, it helps ensure that user information is not shared between applications or systems. Federated identity is commonly used on the internet and is supported by protocols such as OAuth and OpenID Connect.

Figure 20.1 below depicts the key architectural difference between these two models:

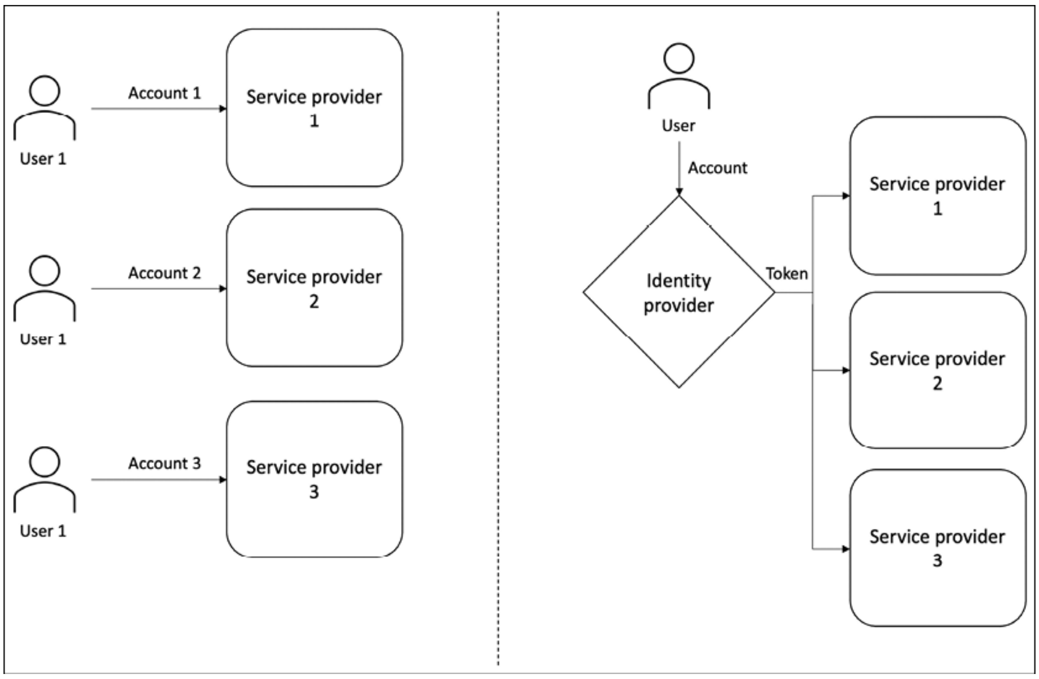

Figure 20.1: Centralized model vs federated model

The preceding diagram shows, on the left-hand side, the relationship between a user and service provider in a traditional centralized identity model where a user has an account with the service provider. On the right-hand side, the identity provider entity below the user provides the identity services in a federated identity model.

The idea is to use a central identity provider where you just have one identity account. However, now, through the IDP, you can log in to multiple service providers, e.g., websites or apps. This collection of all the service providers that use the same identity provider is called a federation, hence the name of the model. The parties in a federation rely on each other for authentication, hence they are called relying parties. Several federated identity protocols to enable **Single Sign-On** (**SSO**) have been developed and work quite well, including:

- **Security Assertion Markup Language** (**SAML**) enables the exchange of authentication and authorization data between entities, allowing for federated single sign-on and access to resources across different domains. It uses XML-based messages to securely transmit identity and authorization information between an identity provider and a service provider.
- **OAuth: Open Authorization** is an open standard protocol for delegated access authorization that enables users to grant third-party applications or services access to specific resources on their behalf with-

out the need to share their login credentials. It provides a token-based authentication mechanism that allows resource owners to grant and revoke access to their resources anytime, providing a secure and user-friendly way to share information across different platforms and services.
The OAuth protocol enables authorization between different systems in a flexible and standardized way, making it a widely adopted framework for securing user data in the digital environment. It is commonly used by service providers like Amazon, Google, Facebook, and Twitter to enable users to authorize access to services provided by third-party services. More information on the standard is available here: `https://oauth.net/2/`.

- **OpenID**: The OpenID protocol is an open and decentralized authentication mechanism that the non-profit OpenID Foundation promotes. It allows user authentication across multiple cooperating websites, known as **Relying Parties** (**RPs**), using a third-party **Identity Provider** (**IDP**) service, eliminating the need for website owners to create their own login mechanism. This means users can log in to different websites without having to remember separate usernames and passwords for each one. To create an account, users select an OpenID identity provider, which they can use to sign on to any website that accepts OpenID authentication. It is a commonly used standard by many services on the internet. Some providers of OpenID include Ubuntu One and Microsoft.

This model is also used on the internet quite commonly, e.g. where you use your Twitter account to log in to some other website or use your Amazon account to check out other online sellers' websites.

This model can alleviate some of the problems in the centralized model, but it still suffers from some key problems, which are described below:

- No single service; there are many identity providers, leading to users having to maintain credentials from different identity providers.
- Cross-identity provider security, contractual obligation, data sharing, and management become difficult and also raise security and privacy concerns.
- Identity providers are still centralized.
- High reliability on a single identity provider, which is subject to the same hacking and data breach problems that the centralized identity

model suffers from.
- Moreover, if an identity provider is compromised, it means compromising all service providers linked with the same account.

In short, the federated identity model works well but does have limitations mainly due to its centralized architecture. Generally, both of these models suffer from trust, access management, resource allocation, shared identity, privacy, and difficult-to-use problems.

To manage users' identities and their access to the system, **Identity and Access Management Systems** (**IAMS**) are used. These systems mainly perform two functions, authentication and authorization. Authentication provides assurance about the identity of an entity and authorization determines the access rights, i.e., what an authenticated user is allowed to do on the system.

There are several actors that make up an identity and access management system. These entities include users, service providers, identity providers, and attribute authorities, such as directory services, DNS, and certificate authorities. These systems are prevalent on the internet and in enterprise settings.

IAMS suffer from several challenges, including:

- Privacy and security
- Identity theft possibility
- Forged credentials
- Still using paper credentials for onboarding or, at best, scanned copies of the credential documents!
- How to update personally identifiable information after changes in personal circumstances
- Takes a long time for onboarding, requires lengthy attestation and verification of documents, etc.
- Centralization, subject to cyber-attacks, jurisdictional segmentation, anticompetitive behavior, censorship, exclusion, and inclusion

So, what changed with blockchain? Can we use blockchain to improve or build an altogether new identity system that doesn't suffer from the existing problems that we mentioned earlier? A new model inspired by blockchain has emerged that can alleviate most of these problems, called the decentralized identity model.

Note that it is not necessarily the case that blockchain technology is essential to provide decentralized identity, but the solid infrastructure, security, integrity, and decentralization benefits that blockchain provides makes it a first choice as a platform on which robust decentralized identity solutions can be built. Blockchain provides a strong foundational layer to build a decentralized identity system utilizing the properties of decentralization, integrity, and security that would have otherwise been difficult and in some cases simply impossible (e.g., decentralization) to achieve on their own by other means.

Some benefits that can be realized by using blockchain technology for IAM are listed below:

- Information about identity is auditable, traceable, and verifiable
- Openly accessible by anyone
- Censorship resistant due to blockchain security guarantees
- Thwarts fraud due to transparency
- Promotes better trust due to transparency
- Efficient
- Can lower fraud ratio
- Reduces verification costs
- Can facilitate the provision of **Self-Sovereign Identity** (**SSI**) and DIDs. It can serve as the layer on which SSI and DIDs can be provided with relative ease as compared to other approaches

However, some challenges are the privacy and protection of personally identifiable information, and whether we store personal data on a chain or not. These can be addressed in different ways. See *Chapter 18*, *Blockchain Privacy*, for more details on privacy.

Blockchain can enable the decentralized identity model, facilitating the creation of a decentralized identity, blockchain-based decentralized identity and access management systems, and self-sovereign identity.

Now let’s discuss the decentralized identity model.

Decentralized identity model

This model does not rely on either a centralized or a federated identity model but is decentralized at a fundamental level. It is not account based and works with real identities in a peer-to-peer manner where a direct relationship between entities (peers) is established instead of going through a central provider. There is no "account" that exists in this world; instead, a direct relationship between peers based on credentials is established. Moreover, there is no single party controlling this mechanism; everyone is in control. As long as they want to keep a relationship with an entity, the "connection" is there. If any one party (peer) doesn't want it anymore, they just drop off the channel without leaving any personal information behind.

With this property of peer-to-peer connectivity, decentralization can be achieved because any peer can connect to any other peer directly. The original intent of the internet was to be decentralized and P2P, but over many decades it became centralized due to centralized service providers. What's changed is whether I can now create a truly decentralized identity mechanism that relies on a peer-to-peer network but is truly decentralized, and no single authority can control it. The answer to this is – you guessed it – blockchain!

Blockchain provides a peer-to-peer network and a **Decentralized Public Key Infrastructure** (**DPKI**), which provides the necessary ingredients for developing a decentralized identity model.

Current **Public Key Infrastructure** (**PKI**) systems are based on a trusted third party called a **Certificate Authority** (**CA**), which verifies the identity of users and issues digital certificates that bind public keys to specific identities. They are based on a hierarchical trust model that extends from the user's certificate to a trusted root CA. While this system works reasonably well, ensures the integrity of digital identities, and secures online communication, it is fundamentally centralized. As a result, it suffers from issues such as impersonation and *man-in-the-middle* attacks to get an undeserving valid certificate from the CA, which implicitly trusts a third party, making CAs a single centralized point of failure.

As blockchain is a decentralized and distributed key-value store. It allows for a better approach to managing digital identities securely and transparently without relying on a centralized CA. We can build a more efficient certificate issuance and revocation system using blockchain. Traditional PKI systems have a slow and less secure certificate revocation process. However, in DPKI, revocation and issuance can be done quickly with all nodes on the network updated in real time. By eliminating the dependence on a centralized CA, DPKI makes PKI systems more secure, transparent, efficient, and decentralized, giving users more control over their digital identity.

Fundamentally, it provides a secure way of exchanging public keys to establish a secure P2P connection between two parties and also provides a secure storage layer to store these public keys to allow the verification of digital signatures on digital identity credentials (i.e., **verifiable credentials (VCs)**) that peers can exchange to prove their real-world identity.

As interest in the decentralized identity paradigm grew, a new term emerged: SSI.

Self-sovereign identity

Self-sovereign identity (SSI) can be defined as an identity of a person that is fully owned and controlled by the person. It is not dependent on or subject to any other authority or trusted third party.

Note that SSI doesn't mean that anyone can assert anything about their identity and get away with it. Instead, there is a secure and stable issuance model behind these identities and credentials that, with the help of various other actors, components, and technologies (especially cryptography), ensures the integrity of the system. Moreover, SSI is not just for individuals; it applies to any entity, machine, IoT device, and virtually anything that needs an identity in the digital world.

The control in the decentralized SSI model remains with the user, whereas in the traditional centralized and federated models, the control is in the hands of service providers, issuers, and verifiers.

We can visualize the difference between the centralized and **decentralized SSI** models in the following image:

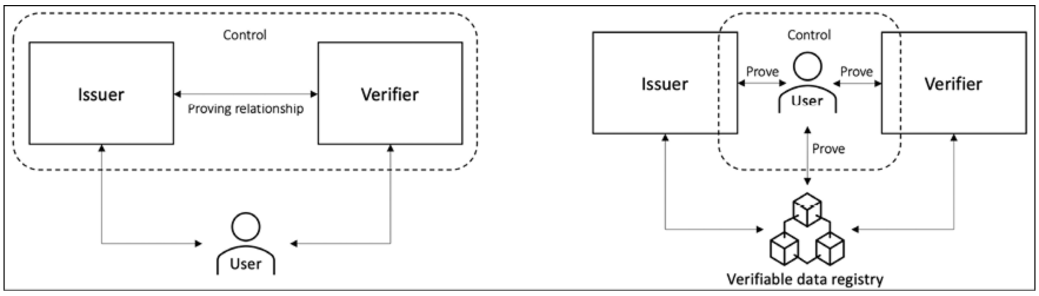

Figure 20.2: Centralized model vs SSI

The preceding diagram shows how the decentralized SSI model, shown on the right-hand side, gives control back to the user, instead of the centralized model shown on the left-hand side, where the user is sitting outside the control mechanism controlled by issuers and verifiers. Note that in the SSI model, the user is in the center of the ecosystem and in control.

This fundamental shift in the control from centralized entities to the user makes the SSI paradigm so impactful and profoundly elegant, which results in enabling tremendously powerful use cases. It can help improve business processes, establish trust, improve the customer experience, address regulatory requirements in innovative ways, resist surveillance and data piracy, and improve government, finance, health, and virtually all other industries.

There are two key constructs that enable this model: VCs and **decentralized identifiers (DIDs)**. Let's now have a look at the composition of SSI.

Components of SSI

In this section, we'll explore the components of SSI, including VCs, which are tamper-evident and cryptographically VCs, and **verifiable presentations (VPs)**, which are data formats used for sharing one or more VCs.

Verifiable credentials

We saw what a paper credential is earlier. VCs are the digital counterparts of paper credentials. Even though we expect paper credentials to be verifiable too, due to the issues covered earlier, the current paper credential paradigm is not secure and foolproof.

Formally, we can define *credentials* as a set of information about an entity (subject) that an authority (issuer) claims to be true, and using these claims, the subject can convince other parties (verifiers) that these claims are true. Here, a trust relationship between the verifier and issuer is implied. Moreover, the subject trusts that the issuer has issued convincing claims that are verifiable. Credentials can belong to human subjects, machines, organizations, and any other entity that needs to convince others about the truth of their claims. The claims can be divided into three broad categories:

- Personal, such as age and ethnicity
- Relationships and associations, such as country of residence, spouse, member of some institute, etc.
- Entitlements, such as legal rights, pension, tax relief, state benefits, etc.

These credentials must be verifiable, i.e., they must be able to convince the verifier that the claims being made are true. To achieve this, the verifier must be able to establish:

- Who the issuer of the credential is, e.g., a university or a government.
- That the credential has not been tampered with.
- That the credential is valid, i.e., has not expired or been revoked. There might be some other validity conditions depending on the use case, e.g., a credential that allows someone to have tax relief is not transferrable. So, one validity condition could be that it has not been transferred or perhaps reused if it is just a one-time offer.

Traditionally, with paper credentials, these guarantees are provided by security features such as guilloche patterns, holograms, or some other feature that cannot be modified or copied. In the digital world, however, we use cryptography to achieve these guarantees. For example, the issuer, validity, and non-tampering (authenticity) can all be verified by using digital signatures.

As we said earlier, VCs are the digital equivalent of physical paper credentials. In essence, VCs are digital versions of the paper credentials that users can carry in their mobile devices just like they were carried in physical wallets before. The difference is that they are digitized (electronically verifiable, cryptographically secure) and have several benefits over their physical counterparts. A comparison between physical paper credentials and digital VCs is shown in the table below, which highlights the benefits that **VCs** have over physical credentials.

Attribute/type	Physical paper credentials	Digital verifiable credentials
Cloning	Possible, even though in some cases it is extremely hard, e.g., national ID cards and passports. This refers to paper passports, not the new digital machine-readable ones; however, there is the possibility of cloning and alteration with them too.	Not possible.
Hacking	Not too difficult, relatively.	Extremely hard, unless the device on which the VCs are stored is fully accessible to the attacker, which is quite difficult with current security schemes such as 2FA, biometrics, etc.
Privacy preservation	No or little privacy. A border control officer sees everything on your passport.	More privacy is possible as selective disclosure is possible using VCs.
Access and permissions	Full access to anyone; a verifier can see all claims and attributes related to an entity.	Supports the principle of least authority/selective permissions (access control), which allows for more security.

Cost	High, especially when producing travel documents and other sensitive documents due to the high cost of special security features that go into every document.	Very low cost, only initial setup/infrastructure cost; once it has been established you can virtually issue and verify as many credentials as required.
Physical limitations	As physical objects, they can be difficult to carry, prone to stealing and loss, and protection from physical wear and tear is difficult.	Virtually impossible to steal; they are usually stored in mobile devices. Even if the mobile device is lost, replacement VCs can be issued relatively quickly. They are not subject to any physical wear and tear. Easy to carry and protect.
Delegation	Not possible.	Possible, if permitted as per the issuer's policy.

Architecture

The VCs ecosystem consists of various actors and components, which are described below:

- **Issuer**: The entity that issues VCs to the subjects.
- **Subject**: An entity whose claims (attributes) are stored in the VC.
- **Holder**: An entity that is in possession of the VC and presents it to the verifier for verification as and when required.
- **Verifier**: An entity that ensures that the claims made in the VC are true and correct. The verifier receives VCs from the holder and verifies them.
- **Digital wallet**: An entity that stores the VCs for the holder.
- **Digital agent**: Software that acts as an interface between the VC ecosystem and the holder, e.g., an app on a mobile phone.

- **Verifiable data registry**: This is an entity that is the foundation of the VC ecosystem and decentralized identity ecosystem. It exists on the internet and is accessible to nearly all the actors in the VC ecosystem. It stores the data necessary to successfully operate a VC ecosystem, such as issuer public keys, the schema of all VC properties (attributes), a revocation list, an expiry list, and metadata describing VCs and other metadata. **Verifiable Data Registries** (**VDRs**) can be blockchains or decentralized and centralized databases. However, blockchains, due to their inherent security features, can be more suitable in the decentralized identity ecosystem.

The following image shows the high-level architecture of the VCs ecosystem:

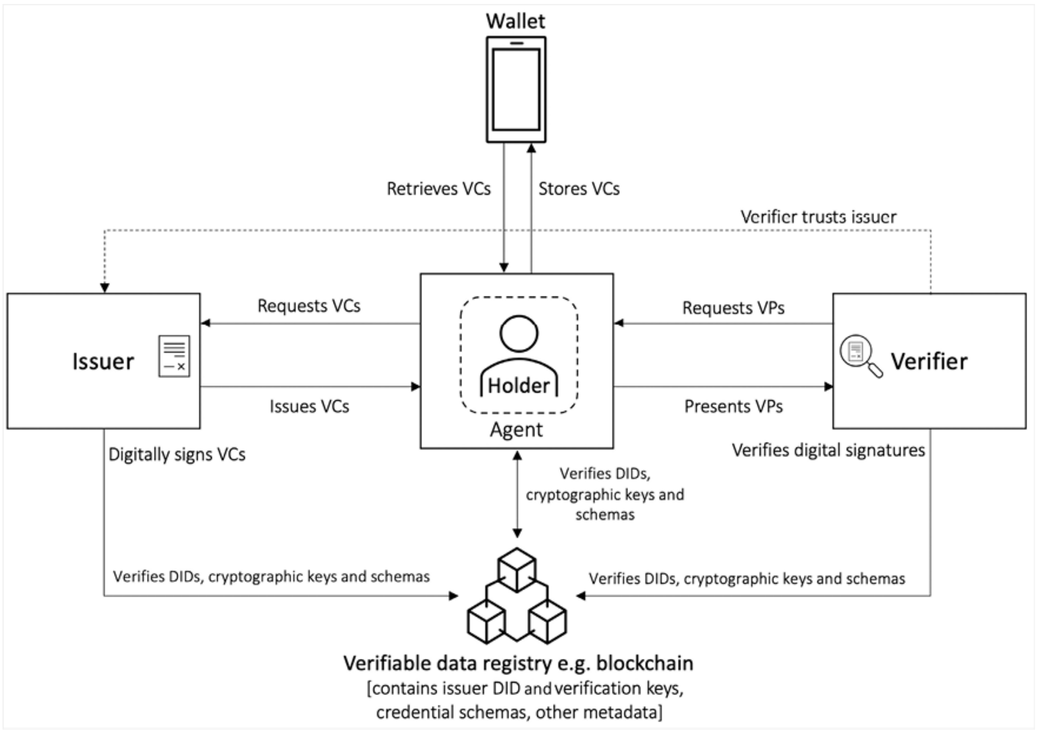
Wallet
Retrieves VCs
Stores VCs
Verifier trusts issuer
Issuer
Requests VCs
Issues VCs
Holder
Agent
Requests VPs
Presents VPs
Verifier
Digitally signs VCs
Verifies digital signatures
Verifies DIDs, cryptographic keys and schemas
Verifies DIDs, cryptographic keys and schemas
Verifies DIDs, cryptographic keys and schemas
Verifiable data registry e.g. blockchain
[contains issuer DID and verification keys, credential schemas, other metadata]

Figure 20.3: VCs ecosystem

Note that in the diagram, the holder of the VC is in charge, while all other actors around it facilitate the credential issuance and verification process. In the VC ecosystem, there is a need for the verifier to trust issuers as authorities who are authorized to issue VCs; however, the verifiable data registry is trusted by all actors. Verifiers are allowed to establish their own trust relationship and verification rules for the verification of VCs. For example, the VC issued by a national health authority for a Covid vaccination is trustworthy and nearly all verifiers accept it without hesitation. However, one issued by a private entity (even though it is cryptographically sound and correct) may not bear the same level of authenticity and trustworthiness as the one issued by a national health authority. So, verifiers have the choice as to which VCs they trust and are willing to accept. The subject also trusts the wallet in which the VCs are stored to store them securely. Moreover, the verifier and subject of the VC trust the issuer to issue legitimate and correct credentials and revoke them if they expire, are no longer valid, or are compromised. Trust involves some type of public and private key/certificate utilization and management, and blockchain delivers this mechanism.

Remember that the subject and holder are usually the same entity, but sometimes they can be different, e.g., an access pass to a nursery issued to a child but managed by parents or a vaccine pass for a cat issued to the pet owner. However, it can really be the cat who owns the VC in its chip (tag).

Now we'll describe the structure of a VC.

Structure of a VC

A VC is usually composed of eight elements:

- **Context**: Contains one or more **Uniform Resource Identifiers** (**URIs**), which provide the information needed to properly interpret and verify a credential, including the standards, data models, and vocabularies used to construct the VC.

  > A URI is a string of characters identifying a name or resource on the internet or any other network. It provides a standard way to identify resources such as web pages, images, videos, and other files. A URI locates a resource on a network and specifies the protocol to retrieve it, such as HTTP, FTP, or file.

- **Type**: Contains URIs that assert the type of the VC. The verifier reads the type and uses this information to decide whether or not they can interpret and handle the credential correctly. If the verifier encounters a type that is unfamiliar or unrecognized, they simply reject the credential.
- **ID**: This is the unique ID created by the issuer for this VC and contains a single URI.

- **Issuer**: This describes the entity that issued the VC. It contains a URI that points to a document that describes the issuer.
- **Credential subject**: This property conveys information about the subject of the credential. It includes a pseudonymous identifier of the subject in the form of a URI, along with a set of properties (claims) that the issuer is asserting about the subject. The pseudonym preserves the privacy of the subject.
- **Proof**: Contains proof – usually a digital signature to verify the VC.
- **Issuance date**: The date of issuance of the VC, in RFC3339 format.
- **Status**: Allows discovery via a URI of information about the current status of the VC, such as whether it is suspended, revoked, etc.

The structure of the VC is shown in *Figure 20.4* below:

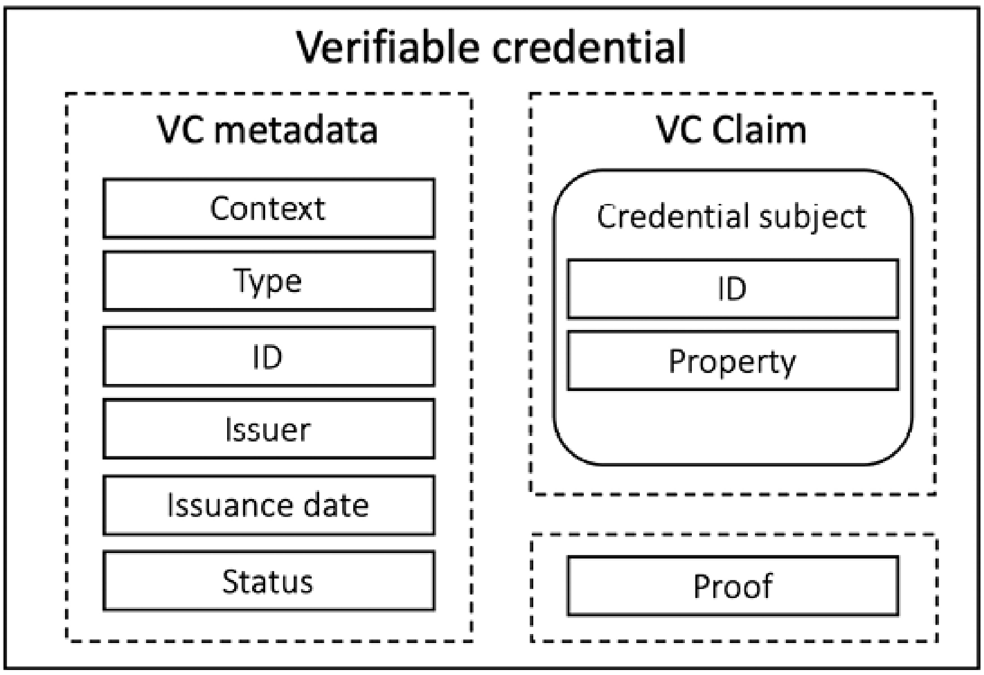

Figure 20.4: Structure of a VC

Figure 20.4 shows the structure of a VC, including metadata, claim, and proof components.

Other properties may include issuance date, expiry date, credential status, and other fields depending on the use case. Moreover, a VC depending on the use case can contain multiple claims and multiple proofs, and can be optionally presented in what's called a VP.

An example VC is shown below, which shows some of the properties discussed above. It's encoded in JSON-LD:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "https://example.com/credentials/123",
  "type": ["VerifiableCredential", "DegreeCredential"],
  "issuer": {
    "id": "https://example.com/issuers/14",
    "name": "Example University"
  },
  "issuanceDate": "2023-02-16T22:12:03Z",
  "credentialSubject": {
    "id": "did:example:123",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science in Computer Science",
      "college": "College of Engineering and Applied Science",
      "university": "Example University",
      "degreeDate": "2022-05-15",
      "degreeStatus": "awarded"
    }
  },
  "proof": {
    "type": "Ed25519Signature2018",
    "created": "2023-02-16T22:12:03Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://example.com/issuers/14#key-1",
    "jws": "eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..0pw88
KHJmP4x4m0F5m5ZOyP5WAGGbsS0L9CKiKjxI5_ZfE5i5y5q5vqBQ9XivnC5pwcbZi75jmZ1xou4
  }
}
```

The example above shows a VC for a person's degree from `"Example University"` with all the necessary details and cryptographic proof. Let's see what each element means:

- `"@context"` : Specifies the JSON-LD context for the document
- `"id"` : A unique identifier for the credential
- `"type"` : The type of the credential, which in this case is `VerifiableCredential` and `DegreeCredential`

- `"issuer"` : The entity that issued the credential, which includes an identifier and name, i.e., `Example University`
- `"issuanceDate"` : The date and time when the credential was issued by the example university
- `"credentialSubject"` : The subject of the credential, which in this case is a person with a Bachelor of Science in Computer Science degree, including the degree details such as the type, name, college, university, degree date, and degree status
- `"proof"` : Cryptographic proof of the credential, including the type of signature, time of creation, purpose of the proof, verification method, and **JSON Web Signature** (**JWS**) of the credential

---

JWS is an IETF proposed standard (described in RFC 7515) for signing arbitrary data.

It is possible to combine multiple VCs for which VPs are used.

Verifiable presentation

A VP is used to present a single or multiple VCs to a verifier for verification, as shown in *Figure 20.5* below:

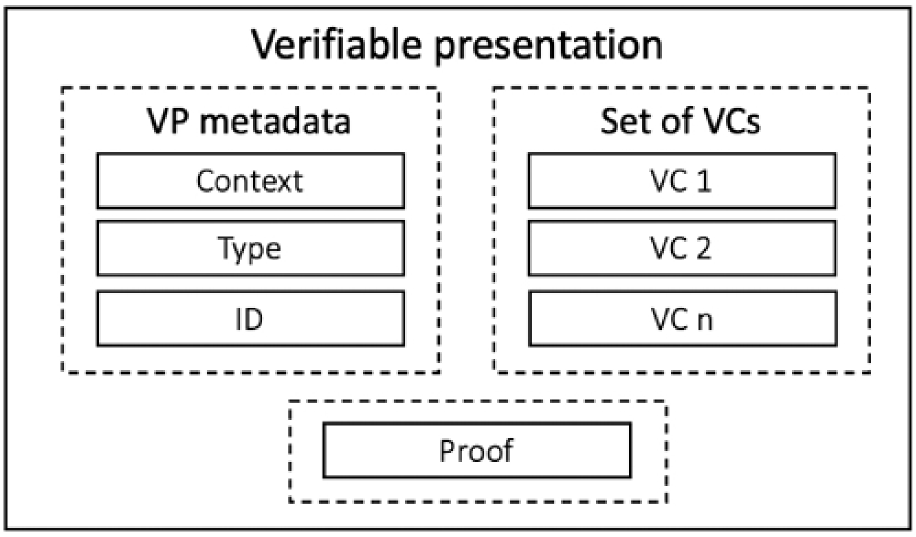

Figure 20.5: Structure of a VP

The preceding diagram shows the structure of a VP, containing metadata about the VP, proof signed by the subject (holder), and a set of VCs.

A VP also allows a holder of a VC to only disclose a subset of information from the VC in order to preserve privacy. A VP can be created that only contains the minimum necessary information required for verification and no more than the holder is willing to share. For example, only reveal that the holder is above 18 years of age and nothing else, so the holder will only send that information as a VP to the verifier for verification. This is called selective disclosure or minimum disclosure.

Decentralized identifiers

A DID is a self-sovereign identity that is permanent, portable, and verifiable and does not depend on any centralized authority. In practical terms, we can think of a DID as a new type of globally unique identifier or address that is cryptographically augmented to support secure verification and decentralization.

Formally, we can define a DID as a globally unique, permanent, usually cryptographically generated and cryptographically registered identifier that doesn't rely on any centralized authority.

A DID is a new type of URI. *Figure 20.6* below shows what a DID looks like:

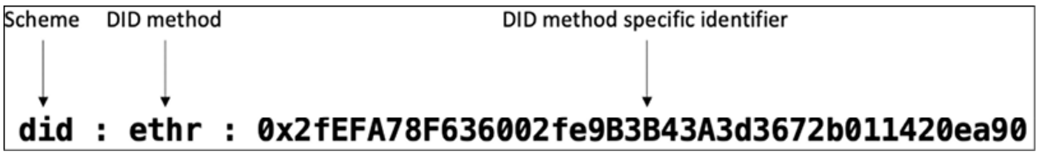
Scheme
DID method
DID method specific identifier
did : ethr : 0x2fEFA78F636002fe9B3B43A3d3672b011420ea90

Figure 20.6: DID

The preceding diagram shows the format of a DID, which is composed of three elements separated by a colon. It's a simple structure, consisting of the scheme name, the DID method, and a DID method-specific string (also called a method-specific identifier). Nevertheless, it carries far-reaching implications in terms of enabling a self-sovereign identity ecosystem:

- **Scheme**: This is a fixed string, `did`, indicating that this is a DID.
- **Method**: This is the method that has been used to generate the DID, e.g., `ethr`, `btcr`, `sov`, `web`, etc. It basically describes where the DID is located and which protocol it is on. This element helps the DID resolve to a corresponding DID document. It can also be defined as a definition of how a specific syntax of an identifier is implemented. The DID method specification specifies the operation using which DIDs

and DID documents are created, resolved, updated, and deactivated. A DID document can be defined as a set of elements that describe the DID subject and cryptographic material that the DID subject can use to authenticate and prove its association with the DID. Each DID method specification defines a scheme that works with a specific DID method.
- **DID method-specific string**: Also called a method-specific identifier. The syntax of the method-specific identifier is defined by the DID method. It is usually a long string generated using cryptographic methods and must be unique. For example, the btcr method-based DID is built on the Bitcoin blockchain. The method-specific identifier (string) is generated from the position of the Bitcoin transaction in the blockchain. Another method, web, is simply a DNS name secured with a TLS/SSL certificate with a path to the DID document. Take the example `did:web:masteringblockchain.com` ; here, the resolution to the DID document is simply through the domain name I already own.
  - `did` is the schema, `web` is the World Wide Web, and `masteringblockchain.com` is the method-specific identifier, i.e., the fully qualified domain name.
  - Another example could be `did:ethr:0x2fEFA78F636002fe9B3B43A3d3672b011420ea90` , where `did` is the schema, `ethr` is the blockchain, i.e., VDR, and the hexadecimal string is the smart contract address.

There are many already-established DID methods and the specifications are available here: https://www.w3.org/TR/did-spec-registries/#did-methods.

Several requirements of DIDs are:

- **Decentralized**: It does not rely on any centralized entity, identity provider, or authority.
- **Cryptographically verifiable**: The ability to cryptographically prove identity and ownership, which is associated with a private/public key pair.
- **Persistent**: It is permanently bound to a subject and doesn't change.
- **Resolvable**: It is possible to discover more information about the subject (for metadata discovery).
- **Ease to create**: Creating DIDs should be quick, easy, and very low cost.

DIDs are the counterparts of VCs. They are cryptographically generated and verifiable. The format of a DID is "scheme: method, did specific string," as shown in *Figure 20.6*.

The DID has several relationships with various entities in the ecosystem. *Figure 20.8* below shows the high-level DID architecture and relationships between the DID, DID document, and DID subject or controller and VDR:

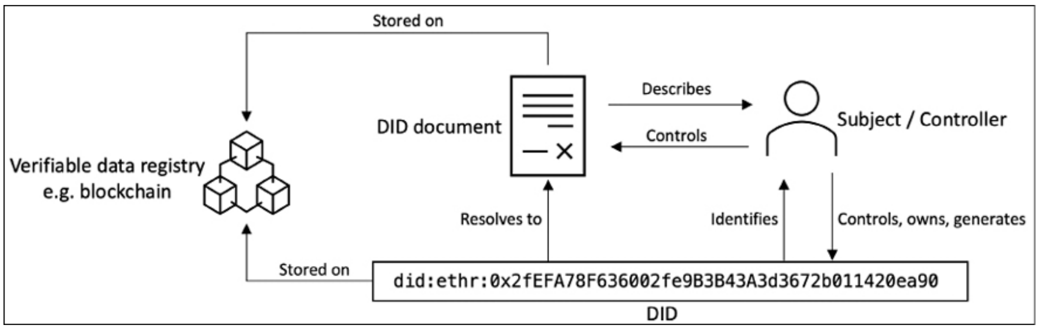

Figure 20.7: DID architecture and relationships

As shown in the diagram, a DID is a unique string following a specific format, described earlier. This DID is resolvable to a DID document. This DID document can exist on a blockchain or some other storage, i.e., a global decentralized key-value store. It can also exist on IPFS, STORJ, or even a web server. The DID document contains controller identification information about the subject, cryptographic material for verification, and some other metadata. The DID and the DID document make cryptographic verifiability possible by a verifier. How it works is that the DID and the DID document are tightly linked together cryptographically. The DID document can be serialized for storage and transmission using JSON, JSON-LD, CBOR, or any other serialization mechanism fit for the use case. DIDs primarily work in conjunction with VCs to form a cryptographically verifiable decentralized identity mechanism.

There are many types of DIDs, and while they all support a basic set of functions, each can be generated using a different mechanism and set of rules, i.e., they differ in how the functionality is implemented. These different implementation mechanisms are called DID methods. There are many DID methods.

Several operations that can be performed on DIDs can be defined as CRUD:

C – how to generate a DID and the associated document – Create

R – the DID document retrieval mechanism – Read

U – the DID document update mechanism – Update

D – the DID deactivation mechanism – Deactivate

For example, on a blockchain, this may translate to executing a smart contract that manages these operations.

A DID method defines how to read or write a DID and a DID document on a specific verifiable data registry, e.g., a blockchain. A DID always identifies a DID subject and resolves to a DID document. A DID document contains information about a specific DID. It contains information such as public keys, authentication methods used for authentication, service end points used for interaction, time stamps, which can be used for audit logs, and digital signatures, which are used to provide data integrity.

So, let's now think about what the relationship between a DID and a VC is. Due to public and private key relationships (i.e., digital signatures), a verifier can verify that the credential did come from an authentic issuer. How does a verifier know that the public key provided is the right one? In a decentralized world, it seems impossible to solve this issue. In traditional PKI, we have high-level CAs that everyone trusts; they sign on behalf of entities who trust them, so in a way a trusted third party is certifying that the public key used by a company is indeed correct, and due to the trust relationship with the root-level CA, everyone is satisfied that the public key is authentic. This is clearly a centralized model.

How does this work in an SSI ecosystem? How is the verifier assured, in a decentralized system, that the public key from the issuer is correct and is indeed from the authentic issuer? This is where DIDs come in. DIDs are identifiers that get tied to the public key, which verifiers can resolve ("lookup") and hence be confident that the public key is indeed authentic.

Now, how can we prove that the DID is bound to the public key and the DID controller (holder, subject) that is in control of the corresponding private key?

> Essentially, the controller is an entity that can make changes to the DID document. Also note that the controller is not nec-

> essarily the subject of the identification.

The solution might be to mathematically generate the DID from the public key instead of arbitrarily creating DIDs. Now, if a mathematical relationship exists between the DID and the public key, the controller can prove that the DID indeed belongs to it, because it has the private key and can respond to any challenges posed by the verifier, e.g., in a challenge/response mechanism. Seems reasonable so far, but what if the controller has to rotate the keys? The DID will have to change, but that's not in line with the goals of DIDs that we saw earlier; DIDs must be persistent. So, this is where DID documents help. The DID document contains the public key, and as a DID is always resolvable to a DID document, there is room to update the DID document and hence the public key and other metadata as required. So, the DID can remain persistent, but the DID document can change, which allows for key rotation and other updates.

The mechanism works as described below:

1. The controller (private key holder) generates the DID based on the genesis public/private key pair.
2. The controller publishes the DID document, which contains the DID and the public key.
3. Now any entity using the DID document can cryptographically verify that the DID and the associated public key are linked (bound) together.
4. If the controller, for some reason, changes (rotates) the key pair, the controller creates an updated DID document and signs it with the previous private key. This can be an update on the blockchain to register the updated DID document. In essence, the controller publishes an updated DID document that contains the original DID and the new public key, but signs with the original private key, which creates a chain of trust between the DID document, which is trackable back through to any number of updates to the original DID document, and the original DID. Each DID document serves as a new digital certificate for the new public key without the need for a certificate authority or any other trusted third party.

So, in summary, we can say that a DID document is a data structure that is associated with a DID and contains information about the entity it represents. It provides a way to publish and retrieve information about a DID, such as its public key, authentication mechanisms, and other relevant metadata. The DID document is typically hosted at a URL, known as the DID endpoint, which can be resolved by a DID resolver. The DID resolver is responsible for looking up the DID document associated with a given DID and returning the associated data to the requester. A generic DID document example is shown below.

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/ethr/credential/v1"
  ],
  "id": "did:ethr:0x123456789abcdefghijklmnopqrstuvwxyz",
  "authentication": [
    {
      "id": "did:ethr:0x123456789abcdefghijklmnopqrstuvwxyz#controller",
      "type": "Secp256k1SignatureAuthentication2018",
      "publicKey": "0x123456789abcdefghijklmnopqrstuvwxyz#controller",
      "controller": "did:ethr:0x123456789abcdefghijklmnopqrstuvwxyz"
    }
  ]
}
```

Some key elements in this document are:

- `context` : Denotes the type of the document, e.g., a JSON-LD document
- `id` : The DID
- `authentication` : Describes cryptographic schemes for verification

There are many others; for a detailed reference, refer to the https://www.w3.org/TR/did-core/ document by W3C.

A DID is a unique identifier that is used to represent an individual or entity on a decentralized network. It is a persistent identifier that is independent of any centralized authority or registry and can be used to authenticate and authorize the holder of the identifier to access resources or participate in transactions. A VC is a digital document that contains verifiable claims about an individual or entity, such as their identity, credentials, or attributes. VCs are designed to be portable so that they can be shared with others and are used to establish trust between parties. They are typically issued by trusted entities, such as universities, governments, or financial institutions, and can be verified by reliable parties using cryptographic proofs. The combination of an identifier (DID) and VC (age, citizenship, or qualification) is what makes a digital identity. These VCs live in your digital wallet on your mobile phone and you can present them at will to any party who needs verification. Again, it's in your full control whether you decide to present it or not. The relationship between DIDs and VCs is that DIDs are used as the subject identifier in VCs.

In other words, a VC can contain a verifiable claim about an individual or entity, and that claim can be associated with a DID that uniquely identifies the subject of the claim. By using DIDs in this way, VCs can be tied to specific entities in a decentralized manner, without relying on centralized authorities or registries.

Digital wallets

A digital wallet stores VCs securely, protecting them from theft and corruption and making them available when required. It is like a cryptocurrency wallet, which we discussed in *Chapter 6*, *Bitcoin Architecture*, but there are some differences. By definition, however, it is the same as a cryptocurrency wallet. There is another type of wallet that is common, the one available in iOS and Android to store payment cards and other credentials.

A digital wallet is a piece of software or hardware that allows its owner to create, store, organize, and secure cryptographic keys, secrets, and other types of sensitive, private information. Sensitive information can include VCs, decentralized identifiers, personal data, accounts, and any digital object that comes under the definition of sensitive or private information.

An SSI wallet is a digital wallet that is primarily used in the decentralized identity ecosystem (self-sovereign identity ecosystem) to store and manage VCs. An SSI wallet may store DIDs, VCs, cards, personal data, access credentials, digital identity documents, travel documents, and many others. It is based on the design principles of portability, privacy first, and security first.

A high-level architecture of an SSI wallet is shown in *Figure 20.8* below:

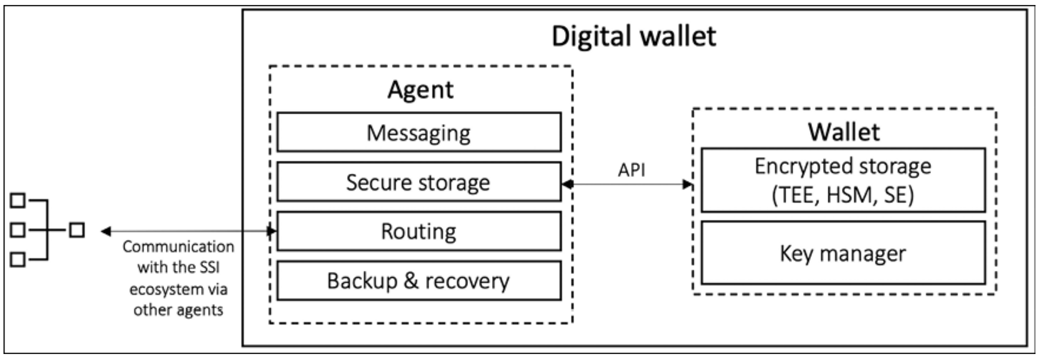

Figure 20.8: SSI wallet high-level architecture

Figure 20.8 shows a component called **Agent**, i.e., a digital agent. This agent mediates communications between wallets, users, and other agents in the SSI ecosystem. The wallet itself is the core protected element where, for example, keys and other sensitive data are stored. One key advantage of this design is the separation of concerns. Even though sometimes there is no distinction made between these two elements, and a wallet is just called a wallet, without any distinction made between the agent and the wallet, internally, the interface component (agent) and wallet are usually two different components.

The agent consists of secure storage, a backup and recovery mechanism, a messaging interface to handle communications, and a routing functionality. Secure storage communicates with the wallet services via a secure API. The wallet consists of two subcomponents, including encrypted storage (trusted execution environments, secure elements, and secure enclaves) and a key management system.

Identity wallets can create DIDs. Remember, in the SSI model, you are creating your own identities and are not dependent on some third party to create one for you. These wallets can facilitate storing these DIDs on a blockchain through blockchain transactions. They can store, sign, and present a VC to a verifier.

Verifiable data registries

Fundamentally, VDRs are databases that serve as a canonical source of truth and trust for DIDs and public keys. VDRs are global, distributed key-value databases. We can use centralized databases for achieving SSI but that might only be acceptable within a small group or consortium. For a global-scale SSI, blockchains are expected to become the VDR of choice. The VDRs can be a public blockchain, a consortium chain, or a private permissioned chain. They can also be application-specific blockchains, purposefully built for SSI, for example, Evernym, Sovrin public ledger, Hyperledger Indy, Hyperledger Ursa, Hyperledger Aries, and Veres One. Such ASBCs for SSI support transactions and record types that make managing DIDs easy.

Governance frameworks

Fundamentally, the trust is established in SSI through cryptography, i.e., cryptographic trust. However, this on its own is not necessarily enough because the element of human trust is also required to establish another layer of trusted relationships between issuers, verifiers, and subjects. This is important because VCs issued by some unknown agency may not be considered valuable unless there is a human level of trust and acknowledgment of their authority is established in the real world. This trust can allow verifiers to determine the legitimacy of issuers under a governance framework that the verifier trusts. Governance frameworks also specify the policies, standards, and procedures from a business, legal, and technical perspective that issuers must follow to issue, or subjects must adhere to to obtain, the VC.

With all the components discussed so far in the SSI ecosystem, we can think of the SSI stack as a four-layer model, which is shown in *Figure 20.9*:

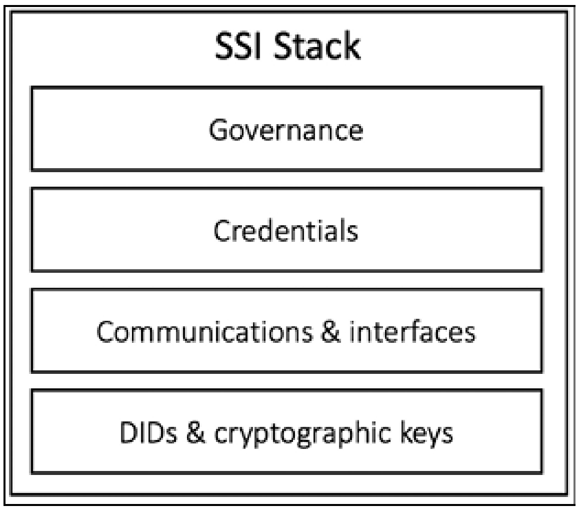

Figure 20.9: Four-layer SSI stack

In the preceding diagram, the governance layer is where a governance authority or a professional body publishes a governance framework that is expected to be adhered to by all actors in the ecosystem. The credentials layer is where the VCs ecosystem exists with issuers, verifiers, and holders. The communication and interfaces layer includes components such as digital agents, wallets, and other communication interfaces. Finally, the identities and keys layer includes DIDs, DID registries, and VDRs. The bottom two layers are concerned with achieving trust at a technical (cryptographic) level, whereas the top two layers achieve trust at a human level.

Identity in Ethereum

Accounts on the Ethereum blockchain can be seen as DIDs. This is because any number of accounts can be generated by anyone without requiring any permission and storing them on a centralized server. While they are not a DID and don't have any VCs associated with it, due to their permissionless nature they can be considered DIDs. As an analogy to DIDs and the SSI ecosystem we discussed earlier, an Ethereum account has a private key and a public key. The public key can be seen as the identity of the controller, whereas the private key is used to sign messages. The Ethereum blockchain can serve as a VDR that can store DIDs. Any VCs issued to the DID can be verified on a chain by validating the issuer's DID that is stored on the Ethereum blockchain. It can be a smart contract where all DIDs are stored.

DIDs are issued, held, and controlled by individuals. An Ethereum account is an example of a DIDs. You can create as many accounts as you want without permission from anyone and without the need to store them in a central registry.

DIDs are stored on distributed ledgers (blockchains) or peer-to-peer networks. This makes DIDs globally unique, resolvable with high availability, and cryptographically verifiable. A DID can be associated with different entities, including people, organizations, or government institutions.

Identity in the world of Web3, DeFi, and Metaverse

Recall that in the times of web 1.0, the simple account-based model of a username and password pair was prevalent. This is still the case with a huge number of websites. However, with web 2.0, a new paradigm started to appear where one identity could be used to authenticate to other service providers too, instead of only the one on which it was originally created. We see this in use on many online services, e.g., "Sign in with Apple" or "Sign in with Google" buttons. Web 2.0 operates in a top-down trust model where centralized platforms are in charge. In the Web3 world, the user is empowered, and we can simply connect to a service provider based on our decentralized identity and the credentials that we possess. No more centralized account models and relying on trusted identity providers; all you need is a wallet and appropriate credentials to connect to a service provider. We saw a glimpse of this when we used MetaMask to connect to Ethereum; however, there are a lot richer applications with this ability.

We can visualize this evolution in a simple diagram; see *Figure 20.10* below.

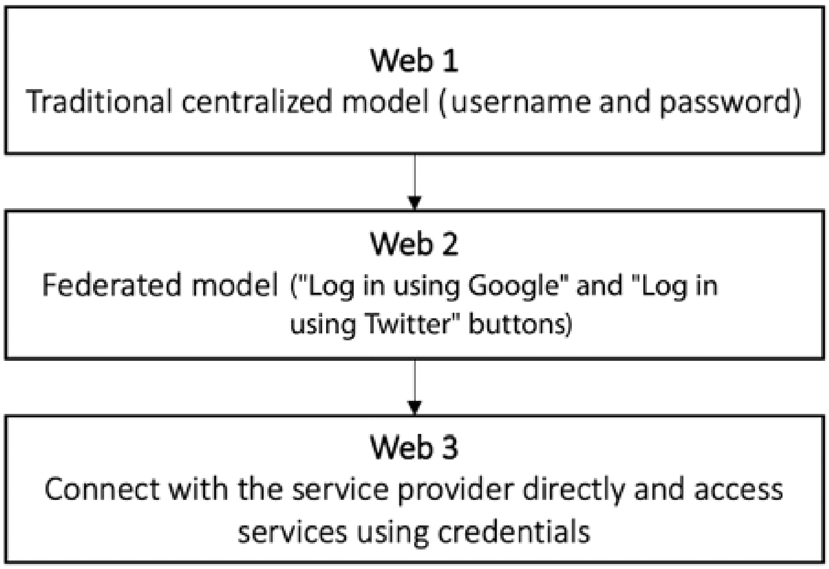
Web 1
Traditional centralized model (username and password)
Web 2
Federated model ("Log in using Google" and "Log in using Twitter" buttons)
Web 3
Connect with the service provider directly and access services using credentials

Figure 20.10: Identity paradigms evolution

Note that we covered this evolution earlier in the chapter, from a different angle; we called them centralized, federated, and decentralized models. However, in the Web3 world, it's helpful to explain these concepts from a web-oriented perspective for better understanding.

A Web3 identity can have applications in many various sectors, including but not limited to finance, gaming, health, and government. Let's discuss some relevant use cases now:

- Imagine a use case on a public blockchain where a financial institution offers a trading facility. It is critical that the identity of the traders is appropriately scrutinized. For this purpose, each trader can be issued a VC from the financial institution that is stored in their digital wallets. At the time of executing the trade, the trading mechanism first checks whether the identity and VC presented by the trader allows them to trade or not. If yes, then the trade can be allowed and executed. This enables use cases that were not possible before. Imagine a fully public blockchain that is running a trading system but is given permission in the sense that only holders of appropriate VCs can use the system. SSI enables such use cases.

- Imagine another use case, where a customer has properly gone through the **Know Your Customer (KYC)** process and has been onboarded by a financial institution after all the checks have passed. Now, this person can be issued with a VC, which the customer receives and is stored in their digital wallet. Now, this VC empowers this customer to go to any other institution, present the VC, and use the financial services provided by that institution. The customer is now accepted by other institutions because of the VC they hold, which tells others that they are already correctly verified, so they don't have to go through the cumbersome and traditionally repetitive process of KYC again. This can immediately and significantly reduce costs and onboarding times, improve user experience, and increase revenue for all because now the customer can easily and conveniently access more services quicker than ever before.
- Another use case could be where a credit scoring company (issuer) checks the credit score of an individual. If the credit score is satisfactory, it issues a VC to the individual (holder), who presents it to a bank (verifier) to get the loan. This VC can then be used with any financial institution without going through credit score checks every time the holder applies for a loan. There will be an expiry associated with the VC, though, to ensure that a new VC is issued based on the new credit score accordingly.

The concepts of full decentralization and data privacy are the foundational stones of Web3 and SSI plays a critical role in making Web3 a reality. The utilization of SSI in DeFi, Metaverse, gaming, and NFTs is going to shape the future of Web3.

Imagine a governance token stored in your digital cryptocurrency wallet. It could allow you to do something on the chain, e.g., vote on a protocol improvement decision. This very ability to use a token to give you the right to do something on a blockchain is the foundation on which many novel Web3 use cases can be built.

Remember the key point here is that if you log in with a username and password, you are in the Web 1.0/Web 2.0 world, whereas if you use your wallet to access a service, you are in the world of Web3. Note that I haven't used the words "log in" and instead used "access a service." This means that you don't really have to log in to the system; your VC enables you to do something by just presenting it to a verifier. Do note that the words log in can still be used synonymously, but remember Web3 works differently.

Imagine that a non-transferrable token is actually a VC. Think of how this could empower anyone, on a global scale, to own a token that works as an access right to do some operation on a chain or even do something in real life. For example, an **Non-Fungible Token** (**NFT**), which is actually a VC, that I own might be my credential to buy groceries at a discount at a certain department store.

SSI is also being considered in many government initiatives for issuing citizen IDs. Such initiatives include Canadian, Australian, and European commissions.

In a metaverse or an online game, a VC could be a token of accomplishments made in an online gaming competition.

Currently, in the DeFi space, there is no way to verify the true identity of payees receiving payments except their wallet addresses. While this allows anonymity, in some cases, it is legally required to identify the payees. Under SSI, utilizing VCs can solve this problem by issuing users VCs that allow them to make and receive payments. Also, there is no need to reveal all attributes; the VC can contain only limited information that is necessary to ensure that the payments are not being made for illegitimate purposes like financing terrorism and money laundering. This can just be an email address or a citizen ID number that only makes sense to a regulatory authority or issuer but not everyone else on the network. We'll discuss liquidity pools in *Chapter 21*, *Decentralized Finance*. Imagine how **Decentralized Finance** (**DeFi**) pools can benefit from VC operating under an SSI model. A DeFi user can be issued with a VCs that exists in a wallet owned by the user. This VC is used when the wallet connects to the service provider, i.e., blockchain in this case where the DeFi protocol is running.

This VC allows the user to do several operations on the DeFi protocol. For example, they can trade, provide liquidity, or act as a liquidator. All of this can be achieved by using VCs issued to them by the DeFi protocol. The user will not have to log in using account names and passwords; all they need is a VC with defined rights as to what they can do on the DeFi protocol, and they simply connect their wallet to the service provider. The service provider (the DeFi protocol) in this case reads the VC and allows users to perform activities in the protocol according to the role defined in the VC.

A key advantage of decentralized identity is that it can replace account-based login mechanisms. There is no need to remember and maintain usernames and passwords; the users keep VCs issued by service providers in their cryptocurrency/identity wallet and can access the services provided by service providers by simply presenting the VC to them. The service providers can fetch the associated VC associated with the blockchain address (identifier) from the blockchain.

Remember that Web3 is certain, and the journey toward achieving it has already begun, with many important milestones already achieved. The infrastructure, standards, ideas and use cases in the form of blockchains, SSI, and various standardization specifications, e.g., W3C specifications, already exist (albeit improvements are required); but it's time to build Web3 DAPPs so that the transition from web 2.0 to Web3 becomes a reality. It's going to be a lengthy process, but as the foundation is already laid, some applications are already built and in production; it's time to accelerate and do more.

SSI-specific blockchain projects

In this section, we describe ASBCs that are developed specifically for facilitating an SSI ecosystem.

Hyperledger Indy, Aries, Ursa, and AnonCreds

The Hyperledger identity stack consists of Indy, Aries, Ursa, and AnonCreds, which are flexible and can interoperate with other layers in the SSI stack:

- **Hyperledger Indy** was the Hyperledger foundation's first blockchain framework that targeted identity use cases. It is a public permissioned blockchain specifically built for decentralized identity use cases. It can build and publish DIDs, VCs, and other similar elements of the blockchain.

- **Hyperledger Ursa** grew out of Hyperledger Indy as a separate project, which consists of cryptography components of Hyperledger Indy. Now it's a separate project, used by not only Indy but also other projects that need cryptography components. Ursa contains the cryptographic components to be used by Indy, Aries, and any other project that needs a vetted cryptographic layer.
- **AnonCreds** is another project that stems out of Indy. It is the ZKP-based VCs mechanism that has been extracted from Indy as a stand-alone project. It works with not only Indy but also other VDRs, and can be seen as a VDR-agnostic layer.
- **Aries** is another project that spun out of Indy. It can be seen as the digital agent mechanism in the Hyperledger identity stack. Aries is an attempt to bridge DIDs and VCs from multiple SSI ecosystems. Aries enables DID-oriented messaging between agents and uses some protocols to enable the issuance, presentation, and verification of VCs.

Other projects

- **KILT**: KILT is a decentralized blockchain protocol for issuing VCs and DIDs for Web3.
- **Proof of Humanity**: This project aims to thwart sybil attacks by utilizing social verification with video submission to allow users to prove that they are real humans.
- **Sovrin**: This is a public service utility that enables SSI on the internet. More details here: `https://sovrin.org`.
- **uPort**: This platform allows users to create a portable, reusable, and decentralized digital identity that can be used across different services. The identity is a smart contract on the Ethereum blockchain as a digital representation of the user. More information here: `https://www.uport.me`.

Some other initiatives

- **Decentralized Identity Foundation**: `https://identity.foundation`
- **Standardization efforts by W3C**:
  - The VC data model specification can be found here: `https://www.w3.org/TR/vc-data-model/`

- DID specifications can be found here: https://www.w3.org/TR/did-core/

There are many other projects; as the ecosystem is rapidly evolving and thriving, it is expected that more and more projects will emerge.

While using VCs and DIDs allows building a robust and secure decentralized identity system or SSI, it does have its own challenges, which need to be addressed to make it even more efficient and secure. We discuss these challenges next.

Challenges

With all this innovation and evolution, it is clear that the future is Web3 and identity will play a vital role in making the ecosystem a reality and will enable novel use cases that were simply not possible in the web 2.0 world. However, some challenges need to be addressed to further improve the SSI ecosystem:

- Lost device – What if a device (mobile phone) on which all of a person's VCs are stored is stolen or left behind on a train? It is actually the same as a physical wallet with paper credentials such as a passport, ID card, etc. being lost. Some effort has been made to alleviate this issue, but a lot needs to be done. How can lost credentials be recovered instead of requesting them again from the issuer? It's probably quicker, due to their digital nature, to get them issued to you again, but still, if there is a possibility of recovery, it would be a much more user-friendly solution.
- The transition from web 2.0 to Web3 poses some challenges, in terms of migration of data, accounts, etc. from web 2.0 to Web3.
- Digital wallet vulnerabilities – As the device on which the wallet is hosted and the wallet itself are subject to malware and hacking attacks, it's important to ensure that the wallets are tested thoroughly before public release. Nevertheless, many information security risks apply to digital wallets, as we have in the usual IT world, including viruses, hacking, storage corruption, substandard software, and many others.
- Other challenges include limitations on the VDR/blockchain layer, such as privacy, scalability, and interoperability.

Summary

In this chapter, we covered the important subject of decentralized digital identity and how blockchain can enable a decentralized identity and self-sovereign identity ecosystem. We covered various concepts including the models of identity, VCs, DIDs, and SSI wallets, and learned how the VCs ecosystem works. We also explored some examples of projects in this space, what role identity plays in Web3, and how Web3 enables decentralized self-sovereign digital identities. We then covered some projects like Hyperledger Indy and finally looked at some challenges.

In the next chapter, we'll cover **Decentralized Finance**, which is another exciting topic.