Octopus

VaultChain® - A self-sovereign network

It begins with privacy by design.

That means from the very first step, always being in control of your data, what you store, what you share, and how & when you prove ownership of that data and the statements you make about it.

True privacy however, begins with anonymity.

Anonymity by design

Anonymity is the starting point for true privacy, otherwise another party always has access to some form of your private data that you can't control, and like it or not, you are effectively forced to trust that party.

If you are compelled to authenticate to a system with your email address or forced to expose your SMS number to verify some form of access, your privacy is no longer controlled by you.

To begin with absolute anonymity, and ensure all relationships are consensual, a different network design is required. With VaultChain®, trust is built, earned and lost through the relationships we choose to create with other parties. At every step, those relationships are voluntary and consensual. You are never compelled or forced to trust another party in the network.

In short, you are sovereign - **You own you**.

Trust and truth

What is "truth"? This is a question that touches on philosophical logic as much as the technical and legal, and poses the awkward question of what we actually mean by "trust" or "truth".

This is especially relevant in this age of so called "fake news". Facebook, YouTube et al, have decided that their "truth" is best served by their own officially sanctioned "*source of truth*".

Distributed ledgers (DLTs) take a less centralised approach, requiring a consensus, and when enough participants decide that a statement is true, then that is the official "*source of truth*".

These approaches are fine until my "*official*" source disagrees with your "*official*" source. At one end is "*big brother*", and the other end "*mob-rule*", neither very consensual.

In trying to agree on an "official source of truth" in one form of another, we will always hit a variant of this problem, and in the world of the decentralised web this type of problem is not going away.

But what if we decide that we don't need to agree on everything all of the time? What if we accept that our network is loosely coherent?

Network of Trust

VaultChain takes a different approach, one that effectively works on two levels - "***proofs***" - statements that can be cryptographically or mathematically proved and "***claims***" that require another form of "***proof***" or "***claim***" to validate them.

For example, a VaultChain audit trail only ever contains "***proofs***", so we never store timestamps in a transaction, because it would require both of us to agree on what the time was, and if my time is different to your time what happens? Who's timestamp is the "truth"? Instead we store "referential timestamps" that are related to the previous one, and we agree on an order of transactions because the order can be mathematically "proved" and cryptographically enforced.

The other side of this are your "***claims***". You are free to "***claim***" anything you like within your own data - but that doesn't make them true - either temporarily or permanently.

For example, you are free to "***claim***" you are the Prime Minister, that doesn't mean that you "***are***" the Prime Minister, one or more other "***proofs***" or “***claims***” are required to substantiate that "***claim***". At this point we need some form of trusted verification process, and this is where the VaultChain consensual "*network of trust*" solves our problem.

Consider this real-world example:

A bank may require you to present yourself in person, so that an official can verify your passport. But this “*verification*” is in reality still just a "***claim***" - not a "***proof***". The official "***claims***" they saw your passport and "***claims***" it was your picture and name on it. Even when they photocopy it, the official "***claims***" it was your passport they photocopied.

The question of “trust” now just shifts to another party, does the bank "trust" the official’s "***claims***"? So the bank chooses to trust the official’s claims, and the official trusts the passport presented. The validity of your claim, rests on the claims of others. The bank has relied upon a *network of trust*.

Ah you say, but you can always “trust” some people - like perhaps governments. Can you? Which governments? What if we trust one passport more or less than others? Who gets to choose? Again, we quickly find ourselves back to either “*big brother*” or “*mob rule*”.

So in reality, there is rarely if ever, a true "*single source of truth*" in life. Ultimately “trust” and “truth” is always a choice, and to consider them absolutes is a fallacy.

For VaultChain, trust is multi-layered and always consensual, its level is entirely dictated by you, your choices and the network of trust you choose to rely on. You must satisfy yourself as to the level of assurance you require, and who you choose to trust - just as you would in the non-digital world.

But how do you establish and maintain this network of trust?

Sovereign identities - Personas®

At the foundation level, Personas® are essentially a pseudonym for an individual, offering the ability to adopt one of the many "faces" we all "wear" & present to other people throughout our daily lives.

The face you show to your spouse or partner, your daughter or grandson, will be different to the face you show to your bank, your doctor or your electricity company. Each face you show, talks about who you are, how you want them to see you, and how much you want to share with them. These are your Personas.

Personas are a critical part of digital identity and privacy, the ability to present facets of yourself based on the specific relationship you have with another party, is crucial to ensuring that data presented as a result of one relationship cannot be inferred or extended by another.

However, to assume this is the extent of their role in self-sovereign privacy is to fundamentally understate the core concepts that underpins this premise behind identity management.

Authorities

Unlike other "*identity verification providers*", Personas are not about identifying an individual, but about identifying an "***authoritative entity***", and that subtle nuance offers a fundamentally different perspective on identity management. That "*authoritative entity*" may be a human being, but it equally maybe a "*role*" an "*authoritative entity*" is playing or acting on behalf of.

Consider for example, a Persona created to represent the "*Human Resources Director*" of an organisation. That Persona does not represent "Steve" or "Mary" - the actual human HR Director, because they are individuals that perform the "role" of HR Director.

The actual human being who "***is***" the HR Director is granted the right to act on behalf of the HR Director Persona by the owning party.

They are granted the rights to the cryptographic authority that allows them to assume the guardianship to "*sign*" & "*identify*" themselves in that role, but that is not "***who***" they are, but rather the authoritative entity they represent, it is merely a role they play for a prescribed period of time.

When a new employee is hired, they are hired by the authority of the "HR Director" not the individual who is playing that role. When an employee shares their data, they share it with that representative, not the individual human playing that role.

When those individuals leave that role, their cryptographic keys are revoked and they can no longer act on behalf of that role, however, the cryptographic signatures generated by them on behalf of that role, are, and will continue to be valid when a new individual HR Director takes up that position.

Both the Persona of the individual who "***is***" the HR Director, and the Persona that represents the "***role***" of HR Director are "*authorities*" in their own right - one for the individual and one for that role or position - representing the company or organisation.

So our human HR Director, may have individual Personas to segment their relationships with their medical practitioners, and others to segment their personal and work relationships, as well as their "HR Director" **Persona**. Each **Persona** is an authoritative entity for the relationships it consensually engages in.

Skin in the game

However unlike some other digital trust technologies, there is no central authority that validates and approves a **Persona**. The trust & authority in a **Persona**, is based on the trust and authority conveyed to it by those who choose to trust it.

The authority a **Persona** offers is built from the other entities that trust it, and in turn the trust others have in them. To coin the phrase popularised by Nassim Taleb, each **Persona** has *"skin-in-the-game"*, and stands to lose their hard won reputational value should they behave or prove to be unworthy of the conveyed trust.

In this way, the authority of a **Persona** is conveyed not by some G.O.D. (Grand Organising Directorate), but by a chain of reputational conferred trust - who trusts who. In the same way our daily lives trust is gained through interactions with others, so is the authority - and thus value of a **Persona**.

So without any requirement on "*big brother*" or "*mob rule*", a "consensus" can be built of which digital identities can or should not be trusted. A **Persona's** trust will increase with age, and as it "ages" so does its reputation and it consequently stands to lose more by behaving in an "untrustworthy" way.

Once the concept of an abstract entity, such as a "role" is accepted as an "authoritative entity", it becomes easy to see how service endpoints and even IoT devices can be authorities as well, all of which they themselves can participate in the network of trust.

Data Honeypots

Having now established sovereignty of your privacy and so your identity. There is one final key requirement for a truly self-sovereign network, sovereignty over the data itself.

There is little point in going through the effort of ensuring sovereignty of privacy and identity if there is not sovereignty of the underlying data.

Where is my data held? Who controls it? Who has access to it? If the answer is not "***me***" then you are not sovereign.

Using a clever decentralized architecture fails to solve the big picture if the underlying data is ultimately stored in a few large corporate data silos along with everyone else's data.

These silos become convenient "*honeypots*" for data miscreants, and security breaches because they fail to isolate "***you***" from everyone else.

Islands of Data

VaultChain solves this problem by creating a decentralised and distributed data storage architecture that splits the siloed data up into a network of self-sovereign "*data islands*".

As its name implies, **VaultChain** is a secure chain of interlinked digital repositories called **Vaults** - or the "**VaultChain**".

Each **Vault** is an isolated, ***anonymous*** repository, that contains an optimised cryptographic graph of linked data sets for the data it represents and the relationships it maintains between other **Vaults** in the network.

The actual data each **Vault** maintains can exist in any location of the owners choosing, private servers, public cloud servers, real-time databases, even a distributed file system such as IPFS.

In the same way **Personas** segment identities and relationships, so the **Vaults** segment the data associated with those relationships.

Each **Vault** is an island of data in its own right, a breach or compromise of one **Vault** doesn't grant access either to the rest of the graph in that **Vault** or any other **Vaults** in the network.

As each **Vault** is an isolated data repository, transactions against the audit trail in one Vault are separate from transactions in another, this has the key advantage that the number of transactions the network is capable of scales linearly with the number of **Vaults** in the network.

Peeling the onion

Each Vault is cryptographically secured by one or more encrypted **VaultKeys** that are stored ***inside*** the **Vault** itself. Access to that **VaultKey** is controlled by a cryptographic token held by another **Vault** somewhere else in the network (which itself has its own **VaultKey**).

Once decrypted, the *token* provides just enough cryptographic knowledge to discover the **Vault** it refers to on the network and the actual cryptographic reference in the graph to the **VaultKey** itself. The token is used to retrieve the **VaultKey** from that **Vault**, which when decrypted, contains another set of tokens that grant access in a similar way to a different subset of the graph that **Vault** contains.

Some of that graph will point to other **Vaults** elsewhere in the network, and those **Vaults** themselves have **VaultKeys** that must first be retrieved and decrypted.

Through this mechanism, the onion can be progressively peeled, and like a collection of infinitely stacking *Russian Dolls*, an individual **Vault** never contains enough information to decrypt itself, but is reliant on another **Vault** on the network having been unlocked first.

A sovereign private network

As each **Vault** is unlocked and the graph datasets it contains decrypted, a private contiguous network space is created for each party.

The graph each **Vault** exposes is unique to the **VaultKey** used to unlock it, thus the graph becomes a private and unbounded network, progressively growing as more **Vaults** in the chain are unlocked.

As the **Vaults** are unlocked, their data is federated into a single composite graph, a graph that can be queried, analysed and updated as if it was a single data silo.

Single source of truth

To ensure inferences can never be made between **Vaults** owned by the same party, nor data accidentally exposed, a new **Vault** with its own **VaultKey** is generated for each relationship and then linked into the graph.

Only the data related to that relationship is placed in that **Vault**. The graph exchanged is encoded in such a way that the owner can maintain their own contiguous graph without exposing the rest of that graph to other parties.

This has the unique property of allowing a single piece of data to be updated in one place and yet flow through the owner's network space, updating all the **Vaults** that contain that piece of data.

Artefacts®

To facilitate this data flow, a **Vault** stores individual fields of data by describing them as a unique set of graph statements. These statements are collated together into any number of distributed graph documents called **Artefacts**.

The set of statements an **Artefact** describes is called its *geometry*. It is this *geometry* that is shared with a recipient not the actual data itself. This allows the value of the data to be updated in one place, yet referenced by the same geometry in multiple locations.

This architecture has two major advantages:

Firstly, because an **Artefact** represents the data geometry, not the data itself, it is the geometry that is shared with another party.

For example, I may construct an **Artefact** that represents my home address, this **Artefact** contains the references to my address not the actual values. Therefore whenever I update a field of data that geometry references, any **Artefact** that references that data is able to be automatically updated. Now I can share my address with 10 parties and update it once, in one place. Moving house just got a whole lot simpler.

Secondly, as the geometry is a set of tokenised cryptographic statements, by just examining an Artefact, there is no way to determine what data values it contains nor what data fields its geometry will resolve to.

Thus, an Artefact to an unauthorised party is just a JSON-LD document with a set of anonymous tokenized statements inside it. Only an *authorised entity* has the ability to resolve those statements to specific **Vaults** in the network and the actual assertions within them.

One interesting consequence of this, is that the graph remains coherent despite being encrypted and so has the highly useful property of being able to be queried without first being decrypted.

Chain of Custody

An **Artefact** can be transferred to 3rd parties to verify and attest to the "*claims*" the **Artefact** makes. The **Artefact's** assertions are signed by that party and returned to the claimant, which can then be re-presented to other interested parties. Those parties may themselves verify and attest to the "*claims*" in the **Artefact**. This process of *claim, verify, attest* creates the **Chain of Custody** (CoC) for an **Artefact**. This CoC is itself part of the network graph and so interested parties can confirm the CoC, and ascertain their own level of trustworthiness of those parties that verified and attested to the statements it contains.

Tragedy of the commons

In peer-to-peer distribution systems such as BitTorrent, as the network grows in size the entire network benefits. In a distributed ledger architecture (DLT), the wider network gains no benefit from increased usage.

In a DLT, increased usage is a pure cost on other network participants. As more participants join, more data must be synchronised and more parties must be involved in the consensus to maintain the "*official source of truth*". DLTs suffer from the "*Tragedy of the commons*".

As we touched on in page 1, VaultChain is a loosely coherent network. This means the network does not attempt to create a consensus across all of its participants. It accepts there may be different states, in different parts of the network. Those different states can be resolved, if they ever need to be, by point to point negotiation.

A loosely coherent network is acceptable, for one core reason. VaultChain does not attempt to solve the "*double-spending problem*".

Double spending

Blockchain is a truly impressive technology, with some exceptional, game changing use cases. But it's raison d'etre was to solve the "*double-spending problem*" for digital currencies. This requires at least a majority of network participants to agree on an "*official source of truth*" to prevent someone re-attempting a state that has already occurred.

But in most use cases of data self-sovereignty, there is no benefit to every other participant, who has zero interest in our transaction, agreeing with us. In fact, inflicting such a required state on the network is utterly redundant. There is no need to ensure I don't "*spend*" my first name twice. There is no requirement - especially given Personas, that I am always known as "*Michael*" not "*Mike*" - given I am sovereign and it should be a choice between me and my consensual relationships.

Network Partitions

As such VaultChain again takes a different approach. It partitions the network into Hubs of interested participants.

By creating a Hub, a secure partition in the network is created. This partition involves only parties interested in the data exchange of its members. A Hub allows these participants to consensually exchange data with each other whilst choosing their own constraints.

Any consensus that needs to be reached, need only be agreed between the members of the Hub for the benefit of their transactions, not the wider network. The Hub can remain coherent whilst the wider network may diverge from what a specific Hub accepts as their "*official source of truth*".

This then accepts that there may be multiple "*sources of truth*" on the network and places the burden of storage and computation squarely on those that benefit from the transaction and need the data. In a VaultChain Hub only those participants that want to hold or verify the data store the data, the wider network is unaffected by their choices and actions.

The members of a Hub are sovereign within their partition of the network and other network participants aren't compelled to spend their computing resources on transactions they have no interest in, or gain no benefit from.

The network is self-sovereign.

The right to be forgotten
Immutability of data is critical in some architectures, for very good reasons. However, in a self-sovereign network, your data, identity and any statements made about them must be mutable.

A network that enforces data immutability, even when all network participants agree to a changed state is not self-sovereign, it is yet another example of "*big-brother*" or "*mob rule*". The right to retract or revoke your previous assertions, the *right to be forgotten* by the network is critical if all participants are to exercise their own agency, act independently and be able to make their own free choices.

VaultChain enables this mutability of data both through the Vaults as *data islands* and the graph statements that the Vaults contain. Any participant is free to request - but not compel, other participants in their Hub to change or remove previous specific statements. If all members of the Hub agree, the data is updated. The Vaults within the Hub contain an audit trail that the change was made, but not what the actual values changed from or to.

Hub members cannot be compelled to revoke existing data for obvious practical reasons, in that by exchanging data, you no longer have direct control over it anymore, but also for legal reasons. There may well be regulatory requirements that specific sets of data be held for specific periods of time, and so cannot just "vanish" on demand.

A participant is able to send auditable revocation requests, and the receiving party able to audit their responses. When the data is finally revoked, the consenting party is able to prove the data no longer exists in their graph and so has complied with the sender's request.

This party could of course have copied the data somewhere else in the meantime, but the signatory of that proof offers a legal recourse, if it later turns out to be untrue.

Finally, the underlying Vaults themselves can be removed from the network, removing the entire graph it exposed, effectively terminating the relationship between both parties.

Dead Drops
The anonymous Vaults exchanged between parties to store their data ensures that no 3rd party is able to determine which Vaults are related to one another, what any one Vault may or may not contain, or which Vaults are members of a Hub.

However, there is a problem. How do we initiate our relationship? How do we communicate without revealing to an eavesdropper that we even have a relationship?

The answer is a Dead Drop Location (DDL). A DDL is an anonymous cryptographic location created in the network, by both parties and exchanged with each other. When one party wishes to contact another, they need only leave an encrypted notification in the recipient's DDL for them to pick up at a future point in time.

A DDL may be exchanged on a one-to-one basis, shared within a Hub, or published publicly for the network as a whole. DDLs can be long lived or transient, and a new DDL can be generated and exchanged between parties merely by notifying the recipient in their DDL of a new location.

Summary

We start with the basis of a private anonymous digital identity - the **Persona®**, literally the face a party wishes to present as their secure boundary to any relationship. The **Persona** can be kept private or published publicly on the network.

The **Persona** is used to create consensual relationships with other identities on the network. These relationships create secure partitions in the network called **Hubs**. It is through these **Hubs** that data can be securely exchanged between interested parties.

All parties are able to request and exchange data with each other using **Artefacts®**, a digital representation of the data they wish to exchange. It could be as simple as a message, something more complex like a passport or contract, or a physical entity like a car or house.

The data asserted by an **Artefact** is stored in a **Vault**, an anonymous decentralised digital repository. Each graph statement in the original **Artefact** is stored and encrypted separately as a cryptographic graph reference in a **Vault**.

These references can be stored anywhere in the network but still referenced by multiple **Artefacts**. This ensures that when a statement is changed, every Artefact that uses that reference can be updated too.

An **Artefact** contains a subset of the owner's data graph, cryptographically tokenized to ensure anonymity whilst still remaining navigable to authorised parties.

One party may also request an **Artefact** from another. This is again a digital representation of the fields of data that the **Artefact** should contain. The recipient is able to approve or reject the requested data and send back a digitally signed representation of the graph requested.

An **Artefact** is known as a "*claim*". It may or may not be truthful or accurate, so this **Artefact** can be transferred to a trusted 3rd party to attest to the validity of these claims.

The attestation could be to prove the passport is issued to the owner, the date of birth of the owner is correct, or the issuer owns a piece of land.

The trusted 3rd party need not be G.O.D like, it could be a doctor, a lawyer, a bank or an insurance company.

Once the **Artefact** has been attested to by a 3rd party, any recipient of this **Artefact** can verify the authenticity and validity of this document. By examining the **Personas** that have signed the **Artefact** to verify the *claims* they can confirm what is known as an **Artefact's** "chain of custody".

Any party can also examine the trustworthiness of a **Persona** by examining who else on the network trusts that entity and in turn, who trusts them. This process creates a network of conferred trust, that incentivises consually "*acting in good faith*".

This is the self-sovereign network called **VaultChain®**