The Technical Foundations of Sovrin

A White Paper from the Sovrin Foundation

Drummond Reed, Jason Law & Daniel Hardman

29th September 2016

sovrin.org

Table of Contents

Status Note

This is a living document. The first version is being published on September 29, 2016 in conjunction with the public announcement of the Sovrin Foundation at the Ctrl-Shift Personal Information Economy Conference in London. This white paper will be maintained by the Sovrin Foundation Technical Governance Board (TGB). New versions will be published whenever significant technical decisions about Sovrin architecture or policies have been made by the TGB. The next two versions are planned in conjunction with key digital identity industry events:

- **Version 1.1** will be published prior to the next Rebooting the Web of Trust workshop to be held in San Francisco October 19-21.
- **Version 1.2** will be published after the Internet Identity Workshop (Oct 25-27) and the first meeting of the W3C Verifiable Claims Working Group (Oct 27-28), both at the Computer History Museum in Mountain View, CA.

The latest version will always be available on the sovrin.org website. Please see the final section for more information.

Terminology Note

All terms used in this and other Sovrin Foundation white papers and documentation are defined in the Sovrin Glossary.

Introduction

The first white paper from the Sovrin Foundation, The Inevitable Rise of Self Sovereign Identity, explains why the Internet does not currently have an identity layer and how the emergence of distributed ledger technology (DLT) finally makes one possible. It also shows how a globally shared ledger can enable true **self-sovereign identity**, where every person, organization, or thing can have its own truly independent digital identity that no other person, company, or government can take away.

Most importantly, the paper explains what most distinguishes Sovrin as a distributed identity system: it is the first **public permissioned ledger**.

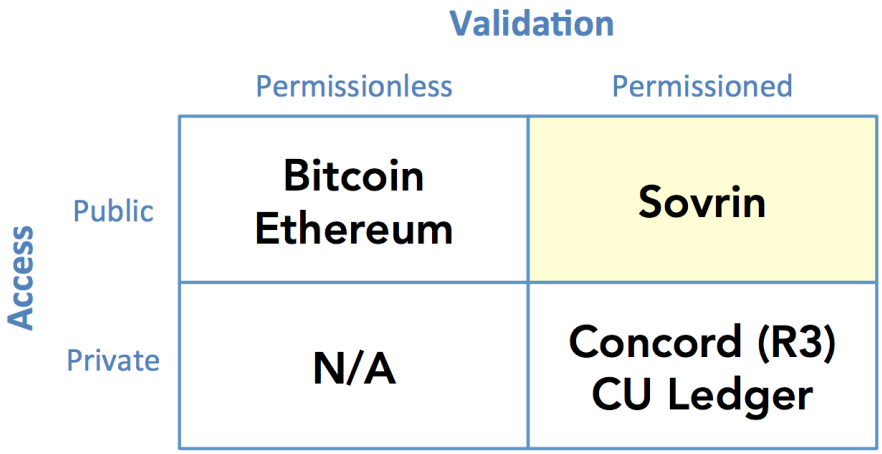

Figure 1: Sovrin is the first public permissioned ledger

As Figure 1 illustrates, **permissionless** ledgers like Bitcoin and Ethereum have opened the world's eyes to the tremendous potential of DLT. They were the first widely deployed systems that enable trust to be established between non-trusting parties without the need for centralized authorities. They did this by adding a layer of complex cryptographic machinery ("mining") that, while slowing throughput, serves to protect the ledger against attacks.

However this cryptographic overhead is only needed for a ledger designed for open **public** usage—where anyone can access the ledger and run a validator node. It is not needed for a **private** ledger used by a restricted set of members, such as the Concord DLT from the R3 consortium of the world's major banks, or the CU Ledger from the credit union industry. These **permissioned** ledgers can run at much higher throughput while still achieving a high level of trust because validator nodes are only operated by known, trusted parties.

By early 2016 it became clear that a hybrid between these two models—a **public permissioned** ledger—could finally solve the Internet identity problem. Such a ledger could provide a fully decentralized layer of root identity records on which people, organizations, and governments around the world could rely. And it could achieve both high throughput and high trust because the validator nodes could be operated by a global network of trusted institutions, all of whom are monitoring each other for good behavior.

This was the genesis of the **Sovrin Foundation**—a global non-profit organization whose sole purpose is governance of a public permissioned ledger for self-sovereign identity. More information about the organization and operation of the Sovrin Foundation can be found in The Inevitable Rise of Self Sovereign Identity white paper and sovrin.org website.

The purpose of this white paper is to explore the technical foundations of the Sovrin identity network for Internet architects, analysts and developers. In particular it will show how Sovrin architecture implements the unique combination of a **public** DLT for self-sovereign identity operated by **permissioned** nodes governed by a global non-profit foundation.

The Sovrin Stack

As an identity layer for the Internet, Sovrin can be thought of as a three-layer software stack as shown in Figure 2:

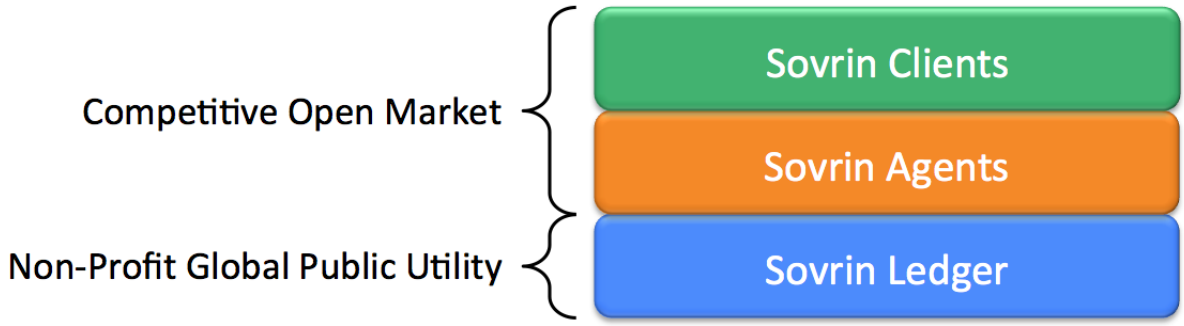

Figure 2: The Sovrin identity stack

1. **The Sovrin ledger** is the foundational component—a globally distributed ledger of root identity records maintained by trusted institutions around the world. The structure and operation of this layer, operated as a non-profit global public utility governed by the Sovrin Foundation, is covered in the first section below.
2. **Sovrin agents** are a new type of network service that give Sovrin identity owners (people and organizations) a permanent, privacy-protecting way to perform identity and data management transactions. Sovrin agents are not strictly required by Sovrin architecture; they simply make a Sovrin identity much easier and more productive to use. Sovrin agents are discussed in detail in the second section below.
3. **Sovrin clients** are apps used by Sovrin identity owners (typically on local devices like smartphones and laptops) to communicate with Sovrin agents and the Sovrin ledger to conduct identity transactions of all types. From a security and encryption standpoint, Sovrin clients are the "key" to Sovrin key management. They are covered in the third section below.

The balance of the paper provides details about other foundational components of Sovrin architecture, including Sovrin transactions, identifiers, keys and key management, claims (attributes) and data management, and the Sovrin Trust Framework.

The Sovrin Ledger

The Plenum Consensus Protocol

Because Sovrin is a public permissioned ledger, it is able run a DLT consensus protocol optimized for both security and scale. The Plenum protocol is an enhancement of the RBFT (Redundant Byzantine Fault Tolerant) protocol first introduced by Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma in 2013. RBFT improved on the earlier PBFT and Aardvark protocols by executing several protocol instances with different primary validator nodes in parallel to detect any performance problems in real-time, without assuming anything about the previous or future performance/condition of the system.

The Plenum protocol, developed primarily by Jason Law and Lovesh Harchandani, further improves on RBFT by adding:

1. Digital signatures for inter-node communication (RBFT uses MAC authenticators, which are faster but do not support non-repudiation).
2. Distribution of requests to only to f+1 nodes (f being the number of faulty nodes).
3. Two election mechanisms for primary validator nodes (one deterministic and one non-deterministic).
4. A gossip protocol (allows Plenum consensus to progress faster in partially partitioned networks).
5. Multiple explicit blacklisting strategies (Plenum considers the severity of a fault and applies the appropriate blacklisting strategy).
6. A catch-up mechanism (for new or crashed/recovered nodes to efficiently and securely regain full state).

Full details about the Plenum protocol, see Scaling a BFT Consensus Protocol for Identity, a paper submitted to the May 2016 ID2020 Rebooting the Web of Trust Design Shop.

Trustees and Stewards

As a public permissioned ledger, Sovrin is ultimately governed by the **Board of Trustees** of the Sovrin Foundation These trustees approve the policies governing selection of **stewards**—the trusted institutions who actually operate the nodes of the distributed ledger. The trustees also ratify decisions of the Technical Governance Board over the evolution of the Sovrin open source code run by each node. Over time, the Sovrin Foundation anticipates that stewards will include financial institutions, universities, healthcare providers, telcos, NGOs, governments, and other institutions in a position of public trust who wish to support a self-sovereign identity layer for the Internet. For more information, please see the Sovrin Foundation white paper Becoming a Sovrin Steward.

Validator and Observer Nodes

The ledger nodes operated by stewards fall into two categories as shown in Figure 3:

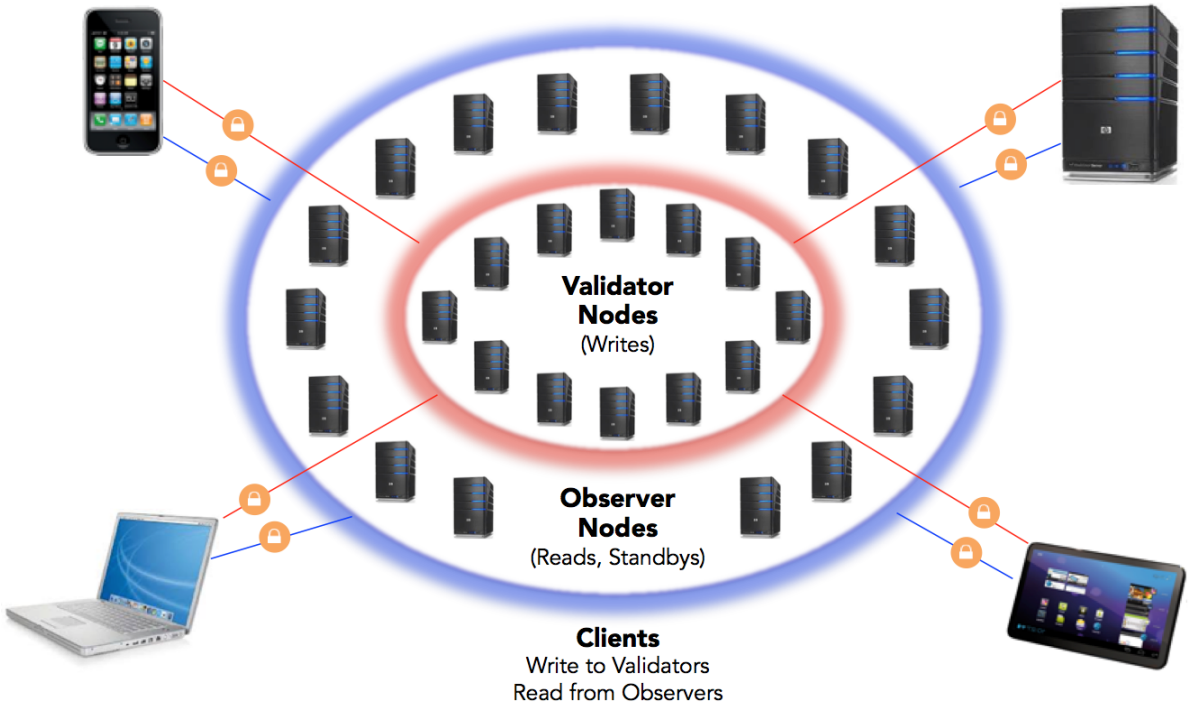

Figure 3: The two basic node types in the Sovrin ledger

Validator nodes run the Plenum consensus protocol to validate new Sovrin transactions. Every "write" to the Sovrin ledger must be sent to a validator node. In the early days of the Sovrin network, all nodes will be validators, since they will be able to handle both the read and write load of the network.

Observer nodes will be needed as the network scales. From the standpoint of Sovrin clients, an observer node is simply a read-only copy of the Sovrin ledger. Observer nodes are designed to fulfill three functions:

1. **Offloading read requests.** As with DNS, LDAP, and other large-scale identity systems, read requests are typically an order of magnitude more numerous than write requests. Sovrin observer nodes enable demand for Sovrin identity records to scale without affecting the performance of validator nodes running the Plenum consensus protocol.
2. **Hot standbys.** Since observer nodes are also operated by Sovrin stewards, they can be swapped into service as an active validator node should there be failure or compromise of another validator node.
3. **Push subscriptions.** Various actors may wish to subscribe to specific Sovrin ledger events. Observer nodes provide a means of pushing event notifications to subscribers without increasing load on the validator nodes.

One of the primary goals of the Sovrin Trust Framework is to ensure diffuse trust for validator and observer nodes, i.e., sufficient geographic diversity, jurisdictional diversity,

industry diversity, etc. For example, while a specific steward may operate any number of observer nodes, at any one point in time that steward may only operate a single active validator node.

Multiple Specialized Ledgers

Although it is convenient to think of Sovrin as a single distributed ledger for self-sovereign identity, it is in fact a combination of four different "fit for purpose" ledgers. Each of these is a public ledger running an instance of the Plenum consensus protocol to accomplish a specific task. In order of importance, these four ledgers are:

1. **The Identity ledger** is the primary ledger—the system of record for all identity records written by Sovrin identity owners.
2. **The Pool ledger** is the system of record for what Sovrin nodes are permitted, at any one point it time, to serve as validator or observer nodes. The Pool ledger stores the outcome of node votes on the Voting ledger. This ledger plays a key role in verified node discovery (see the following section).
3. **The Voting ledger** is where votes among trustees and stewards are held to propose, confirm, or revoke different permissions, e.g., whether a node is permitted to serve as a validator or observer node.
4. **The Config ledger** holds network-wide configuration data set by the Sovrin Foundation Technical Governance Board and approved by the Board of Trustees. Examples include:
    a. How many trustee votes are required to approve a new trustee.
    b. How many steward votes are required to approve a new steward.
    c. What time intervals should be used by nodes when posting throughput metrics.

Note that validator nodes process all write transactions on Sovrin regardless of type. Transactions are not sent to a specific ledger, rather they are sent to a validator node, and once validated, the validators update the appropriate ledger depending on the transaction type.

Node Discovery and Authentication

Sovrin validator nodes will always be able to authenticate Sovrin clients via their digital signatures. But Sovrin clients to also be able to authenticate that they are talking to the real Sovrin validator nodes.

To provide this mutual authentication, a set of genesis transactions are shipped in the code of the Sovrin client (which is why it is important for an identity owner to make sure they are using an authentic Sovrin client). These genesis transactions determine the starting point for a bootstrap set of validator nodes (nodes from the most stable and trusted Sovrin stewards). The Sovrin client interacts with that set of bootstrap nodes to obtain current information about the state of the Pool ledger, and verifies that information using audit proofs, consistency proofs, and a consensus protocol. The client then uses a special "catch-up"

algorithm (in fact the same algorithm used by validator nodes) to efficiently reproduce the current state of the Pool ledger and verify that it is authentic.

At that point the client has full knowledge of the full Pool ledger and is able to interact and verify transactions as needed with multiple authentic validator nodes.

Sovrin Agents

Sovrin agents form the “middle layer” of Sovrin architecture as shown in Figure 4:

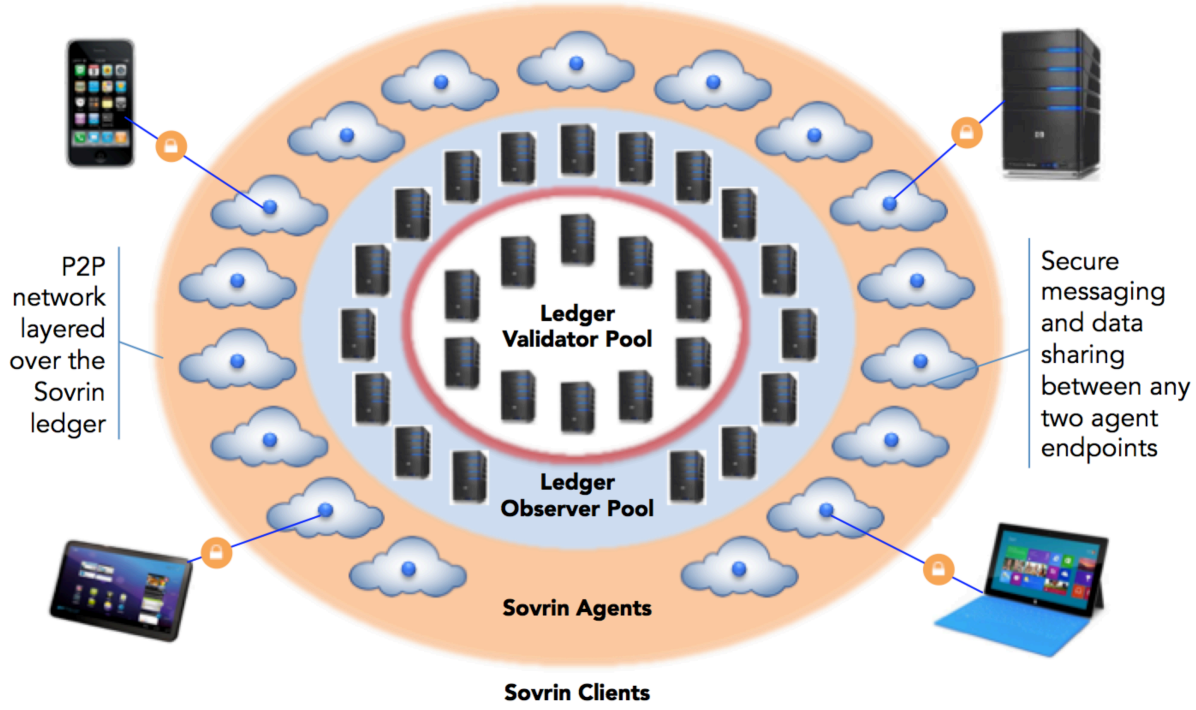

Figure 4: Sovrin agents are the P2P endpoints of the Sovrin network

Both Sovrin agents and Sovrin clients act as clients of the Sovrin ledger. The primary difference is that a Sovrin agent is not just a client; it is also a server—or at least a server process—that has an addressable network endpoint. In general this endpoint will have the same high availability as other critical network infrastructure (routers, DNS, email, etc.)

Agents serve the following core functions in the Sovrin network:

1. **Persistent P2P messaging endpoints.** Sovrin clients operated on edge devices (smartphones, laptops, cars, etc.) typically do not have their own messaging endpoints, but are provided those endpoints by network services running on their behalf (IP routers, domain name servers, email servers, web servers, etc.) Sovrin agents are the network service that make Sovrin clients addressable. This new type of network addressability has several unique advantages as further described the *Agent Endpoints* section below.

2. **Coordination endpoints for multiple clients.** Looked at from the other direction—the viewpoint of the identity owner—Sovrin agents are a solution to the challenge of coordinating messages and state across multiple Sovrin clients operating on multiple edge devices (smartphones, laptops, cars, etc.) This is not dissimilar to other cloud messaging/storage/sharing services such as Apple iMessage, Apple iCloud or Dropbox, except it has the full security, privacy, portability, and self-sovereignty advantages of the Sovrin network.

3. **Encrypted backup of Sovrin keyrings.** As described in *Sovrin Keys and Key Management*, below, an identity owner's Sovrin keychain is literally "the keys to their kingdom". So Sovrin agents will typically perform the service of maintaining an encrypted backup to simplify key recovery.

4. **Encrypted data storage and sharing.** Similar to Apple iCloud, Sovrin agents can simplify and automate the process of storing and sharing data, only in this case the data can all be encrypted and managed by the identity owner's own Sovrin keychain. See the *Containers and Data Storage* section below.

Note that Sovrin agents are not strictly required by Sovrin architecture; any Sovrin client can talk directly to the Sovrin ledger, and in certain cases this is recommended in order to ensure the integrity of specific transactions or verify the integrity of Sovrin agents. Some excellent reasons to ensure that Sovrin clients are never fully dependent on Sovrin agents are discussing in the DPKI: Decentralized Public Key Infrastructure paper published by attendees of the first Rebooting the Web of Trust workshop.

Agencies

Although the Sovrin ledger is operated cooperatively by steward organizations around the world, and thus does not need to be "hosted" by Sovrin identity owners, Sovrin agents require a host. An identity owner can always be their own host, as they can today with an email server, Web server, file server, etc. Or an identity owner can choose a third-party hosting service, called an *agency*.

Much as we have commercial email, Web, and file hosting providers today, the emergence of Sovrin infrastructure will kickstart a competitive market for commercial agencies that offer Sovrin agent hosting services to individuals, organizations, governments, and any other Sovrin identity owners.

Agent Endpoints

While the Sovrin ledger is permanently addressable via all of its validator node endpoints as described above, Sovrin identity owners—people, organizations, things—are not directly addressable on the network—let alone by permanent address. Instead we are assigned those endpoints by the different network services we use (mobile phone numbers for our phones, IP addresses for our computers, domain names for our websites, email addresses for our email, etc.)

With all of these network services, your address is dependent on your service provider. This dependency is a form of lock-in—so much so that in some cases its anti-competitive nature has led to regulatory requirements for portability, e.g., mobile phone number portability regulations in the United States.

Still, ultimately, with all of these services, your network address is ultimately delivered via a service provider over which you do not have control. In other words, these addresses are not self-sovereign.

With Sovrin, you as an identity owner finally have a network address fully under your control, for life. A self-sovereign address.

In fact, you have as many as you legitimately need. This is a vitally important feature of Sovrin privacy architecture. As a rule, as a Sovrin identity owner you may have as many distinct Sovrin identities registered on the Sovrin ledger as you need to keep the contextual separation you want. For example, you may want to keep the identity you use to share medical information separate from the identity you use to share political views separate from the identity you use for your work.

If all of those Sovrin identities shared the same agent endpoint, they would all be fully correlate-able. To avoid this correlation, each separate Sovrin identity needs a separate agent endpoint. This service—of automatically providing different private agent endpoints for different Sovrin identities—is expected to be a standard feature of commercial Sovrin agencies. In many ways it will be modern day equivalent of the unlisted phone number services that telcos began offering back in the 1950s.

Of course Sovrin agent endpoints are vastly more powerful than phone numbers. Not only can they support much more sophisticated forms of privacy protection, but they can use the Sovrin ledger, Sovrin key management, and P2P protocols to dynamically negotiate and provision endpoints for almost any kind of communications, from secure messaging, file sharing, and data sharing to VoiP, SIP, and WebRPC connections.

Finally, Sovrin agent endpoints in combination with semantic data interchange protocols like OASIS XDI can offer groundbreaking new discovery services—ways to find and connect with resources (people, organizations, things, databases, bots) that are far more sophisticated, granular, secure, and private than the web searches we typically perform today.

Containers and Data Storage

A second key feature of Sovrin agents is maintaining the identity owner's off-ledger "container" of Sovrin identity data. This logical container can contain literally any information owned or controlled by the identity owner, including the owner's:

- **Identity graph** of Sovrin identity claims (see the *Sovrin Claims* section)
- **Relationship graph** of Sovrin connections (the Sovrin equivalent of a user's address book, contact lists, and social graphs on various social networks)

- **Reputation graph** of Sovrin reputation claims (see the *Sovrin Trust Framework* section).
- **Data graph** of files, photos, videos, and any other digital asset or artifact that the identity owner wishes to keep "under management".

Again, this is a *logical* container because part of the responsibility of a Sovrin agent—if the identity owner wishes it—is to manage where and how this data is actually stored, encrypted, backed up, shared, and otherwise managed online. In other words, a Sovrin agent may not need to actually physically store any data at all, but just manage, coordinate, and lock/unlock the identity owner's data from other data sources and providers.

Again, this is another area where commercial agencies will compete to offer the fastest, safest, most private, and most convenient data storage and management options. This is also an area where new decentralized, encrypted file management systems like IPFS and databases like BigChainDB may play a very significant role.

This approach to personal data management under the identity owner's control and consent is wonderfully synergistic with the vision articulated in the MyData white paper from the Finnish Ministry of Transport and Communications:

> *In the MyData architecture, data flows from a data source to a service or application that uses the data. It is important to understand that within the MyData infrastructure, the flow of consents or permissions is separate from the actual flow of data. The MyData account should not be confused with personal data storage (PDS) solutions that enable storing data in a secure place under the direct control of an individual custodian. The primary function of a MyData account is to enable consent management – the data itself is not necessarily streamed through the servers where the MyData account is hosted.*
>
> *Application Programming Interfaces (APIs) enable interaction between data sources and data users. MyData compliant APIs provide data in a machine readable format and also enable the data sources and users to exchange information with the MyData account. As a result, it is possible to build a centralized dashboard where the individual may grant access and give or cancel permissions for multiple data sources and services. Any service provider can build a MyData API and enable their service to be connected with MyData accounts directly. If the service does not have a MyData-compliant API, it can be connected via a MyData proxy service.*
>
> *Standardized MyData architecture makes the accounts interoperable and allows individuals to easily switch operators. This is major element contributing MyData's trustworthiness. Interoperability is the core advantage provided by MyData, but it is also the core challenge. Interoperability within the data management system can be understood as functioning similarly to interoperability in mobile telephone networks. Both systems require a common network that connects distributed nodes. Global interoperability and transferability of MyData accounts (and thus individual's consents) between operators requires further standardization and design on e.g. trust networks, data formats, and semantics.*

Data Portability

A Sovrin agent service, whether hosted directly by an identity owner or by a commercial provider, is an instantiation of precisely these MyData principles, and in particular the principle of data interoperability and portability. On this score, Sovrin architecture contributes the following key features:

- **Sovrin identities are 100% "portable"** because they are completely self-sovereign, i.e., they do not exist in any service provider's system or database.
- **Sovrin agent endpoints are always under the identity owner's control**, so the identity owner can port their Sovrin agent service to a new host at any time.
- **Sovrin keys and keychains are portable across any device** that can host a Sovrin client application.
- **Sovrin data graphs should be maintained in a system-independent semantic graph format** such as JSON-LD or OASIS XDI, so the data itself can be ported across data stores and systems without loss of semantics.

As a general rule, if it's not portable, it's not truly self-sovereign, so all aspects of Sovrin network architecture must be prioritize portability as much as security, privacy, and trust.

Sovrin Clients

Sovrin clients are the "last mile" of Sovrin architecture—the point at which control actually sits in the identity owner's hands. From a technical standpoint, there are several features of Sovrin client architecture that are vitally important.

Keychains

First and foremost, a Sovrin identity owner's private keys—the keys that literally unlock their Sovrin identity and everything that can be done with it—become the owner's single most important digital asset. So the single most critical function of a Sovrin client is to manage and protect the identity owner's Sovrin keychain.

This keychain is similar in many ways to the OS-specific keychains currently offered by modern operating systems such as Apple OSX and Microsoft Windows. What distinguishes the Sovrin keychain as a cryptographic data structure is that it is designed from the ground up to be completely self-sovereign—independent of any specific OS, device, application, or network—even the Sovrin network.

To do its job in maintaining the security and privacy of the identity owner's Sovrin transactions, every Sovrin client must maintain a copy of all—or at least a portion of—the owner's Sovrin keychain. For more about this function, see *Sovrin Keys and Key Management* below.

Local Containers

The second job of the Sovrin client is to maintain a local copy of all or a portion of the identity owner's Sovrin data container. Again, as with a Sovrin agent, this is a logical container, so part of the function of a Sovrin client (especially a full-featured client) is to manage how this data is securely physically stored on a particular device by a particular operating system.

Since the storage capacity and bandwidth of different devices varies dramatically, it may be that none of an identity owner's Sovrin clients has a complete local copy of the owner's Sovrin data container. So another function of a Sovrin client is to manage how the portion of the container stored on that particular device is shared and if necessary synchronized across the owner's set of Sovrin clients. This is an excellent example of where a Sovrin agent can assist, either by enabling secure messaging between the clients, by maintaining an encrypted backup of the container, or both.

First-Time Provisioning

Just like an email or file sharing client, a Sovrin client must first be connected with the network before it can do its work. This step, called *provisioning*, is critical from a security and network integrity standpoint because it involves authenticating the Sovrin client to the network and vice versa.

In the case of an identity owner that does not yet have a Sovrin identity and is provisioning their first Sovrin client, the provisioning process involves requires connecting with an existing Sovrin identity owner called a **trust anchor**. Trust anchors have permission to add new identities to the network—see the *Sovrin Trust Framework* section. Thus new identity owners must have an existing (or develop a new) trust relationship with a trust anchor in order to register.

For first-time provisioning of a new Sovrin identity, the basic steps are:

1. The identity owner first uses the Sovrin client (possibly in conjunction with another client, such as a browser or smartphone) to connect to and independently prove his/her identity to the website or web service of the trust anchor.
2. The trust anchor returns a challenge token to the Sovrin client.
3. The Sovrin client generates a new cryptographic keypair and adds it to the identity owner's Sovrin keychain. The public (verification) key is used to create the identity owner's Sovrin identifier (see the *Sovrin Identifiers* section). The private (signing) key is used to sign the challenge token.
4. The Sovrin client returns the signed challenge token to the trust anchor.
5. The trust anchor verifies the signature on the challenge token, thereby authenticating the new key pair, and then proceeds to register the new Sovrin identifier on the Sovrin ledger.

6. The trust anchor signs a receipt message for the registration with the trust anchor's own private key and returns it to the Sovrin client.

7. The Sovrin client verifies the trust anchor's signature by checking the Sovrin ledger to verify the trust anchor's Sovrin identity.

Once this process is complete, the identity owner is registered on the Sovrin ledger and is ready to begin using a Sovrin agent. In some cases, the trust anchor may also be an agency providing Sovrin agent services. In this case the provisioning process above will also include provisioning of a new Sovrin agent. In other cases, the identity owner may wish to host his/her own Sovrin agent, or choose to register a Sovrin agent with a different agency. In those cases, the identity owner will use the Sovrin client to go through a similar process to authenticate the Sovrin client to the host agency and vice versa. Once authenticated in both directions, the agency will provision a new Sovrin agent.

Once the identity owner has both a Sovrin identity and a Sovrin agent, the Sovrin client is provisioned and ready to start creating connections and sharing data.

Provisioning Additional Sovrin Clients

Part of the convenience of self-sovereign identity is that an identity owner can use it across all of his/her devices, apps, services, and networks. To experience this convenience, the identity owner needs to have a Sovrin client installed and provisioned on each of the owner's devices.

Provisioning of a new Sovrin client is simpler and faster because it does not need to involve a trust anchor or provisioning of a new Sovrin agent. Rather the identity owner can simply perform the authentication process between the new Sovrin client and the owner's current Sovrin agent. The identity owner herself can then approve adding the new Sovrin client from one of the owner's existing provisioned Sovrin clients, e.g., approve adding a new Sovrin client on a laptop from the owner's existing smartphone, or vice versa.

Usability

The final vital feature of Sovrin clients is usability. Simply put, to gain wide adoption, Sovrin clients must make using a Sovrin identity as easy as using an ordinary browser, email client, or chat client, while at the same time ensuring that the user's Sovrin identity and keychain are secure, private, and protected from many common user errors and acts of God, including social engineering, loss of a device, or compromise of a private key.

This is a tall order, but it is also another place where both open source and commercial providers of Sovrin client software will compete, just as browser, email, and chat app vendors compete on usability, security, and privacy. With Sovrin infrastructure, this competition will be a virtuous cycle that produces better and better Sovrin clients that both protect identity owners and make it very easy and intuitive to use a Sovrin identity.

Sovrin Identifiers

As a fully decentralized global ledger, Sovrin architecture relies on **decentralized identifiers** (DIDs) for every Sovrin identity record. This concept of decentralized identifiers is not new; the structure of UUIDs (universally unique identifiers) was first developed in the 1980s and later became a standard feature of the Open Software Foundation's Distributed Computing Environment. UUIDs achieve global uniqueness without a centralized registry service by using an algorithm that generates 128-bit values with sufficient entropy that the chance of collision are infinitesimally small.

The Sovrin Technical Governance Board intends for Sovrin identifiers to conform with the DID Identifiers and Objects Specification currently being developed based on requirements identified by members of the Rebooting the Web of Trust group with support from an SBIR (Small Business Innovation Research) grant from the U.S. Department of Homeland Security. This section explains more about the three types of decentralized identifiers to be supported by the DID specification.

DIDs (Decentralized IDentifiers)

A pure DID is a decentralized identifier with no cryptographic properties—specifically a version 4 UUID serialized as defined in section 3 of RFC 4122. An example:

```
did:76d0cdb7-9c75-4be5-8e5a-e2d7a35ce907
```

There is one primary use case for a pure DID: when a Sovrin identity record needs to be generated for an identity owner who does not currently have the capability of generating or storing her own keys. There are many examples of this use case, including trust anchors assigning Sovrin identities on behalf of refugees or other members of at-risk populations who do not have their own computing devices or storage.

CIDs (Cryptographic IDentifiers)

A CID, also called a **cryptonym**, is a globally unique identifier that is algorithmically generated from cryptographic key material so that it has specific cryptographic properties. For example, A CID may be a public key, the hash of a public key, or a truncated hash of a public key.

There will be multiple schemes for CIDs as cryptography evolves; thus the syntax is intentionally extensible. At the same time, all CIDs are required to be globally unique without the need for centralized registration, so CID schemes are forward-compatible.

The syntax and cryptographic properties of a particular CID scheme are defined by its associated specification. The format of a CID must follow the rules defined in section 13.6.2 (Cryptographic Identifiers) of the OASIS XDI Core 1.0 specification:

1. The CID scheme name MUST begin with "cid-x:", in all lowercase, where x is an integer greater than or equal to 1. This integer is the CID scheme number.

2. The value of the string following the CID scheme name MUST be a valid JSON string value defined by the specification for that CID scheme number.

Example:

```
cid-1:MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCABMC
```

The OASIS XDI Technical Committee defines CID schemes and assigns CID scheme numbers. CID schemes MAY also be defined by other standards bodies. For purposes of disambiguation, this specification will use CID scheme numbers assigned by the OASIS XDI Technical Committee.

The proposed **cid-1** scheme is summarized as follows:

- The starting value is a 256 bit Ed25519 public key.
- This value is then base58Check encoding to produce a 44 character string.

Aliases

DIDs and CIDs both achieve global uniqueness without the need for a central registration authority. However, as Zooko's Triangle illustrates, this comes at the cost of human memorability. In some use cases, it is desirable to be able to discover the DID record for an entity from a conventional address for the entity, such as a mobile telephone number, an email address, a Twitter handle, a URI, etc.

An alias is a hash of a conventional address produced using a specified normalization algorithm and a specified hashing algorithm. These algorithms must be specified in a separate alias specification. In addition, because the conventional address behind an alias is not algorithmically generated but registered with a registration authority, a principal must prove ownership of an alias prior to including it in a DID object.

Aliases represent a tradeoff between convenience and privacy. When a DID or CID record is indexed by one or more aliases, it enables discovery of the identity owner by parties who know the conventional address. Because it is a hash, it does not reveal the conventional address directly, however this is relatively weak privacy protection because an alias can be easily discovered via dictionary attacks. Therefore aliases should only be registered only with the permission of the identity owner, and ideally they should either be subject to access controls that meet the principal's privacy requirements or salted (e.g., combined with a passcode) in such a way that the identity owner retains control of who can perform a lookup.

For all these reasons, it is unclear whether aliases will be supported by Sovrin, or if so, whether they will be introduced at a certain point of maturity has been reached. That decision will be made by the Sovrin Foundation Board of Trustees with the input of the Technical Governance Board.

Example of an alias based on a SHA 256 hashing algorithm:

```
sha-256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934
ca495991b7852b855
```

Sovrin Keys and Key Management

Sovrin keychains were already introduced above in the Sovrin Clients section. In this section we will go into more detail about how Sovrin serves as a decentralized public key infrastructure (DPKI). For more information about DPKI architecture, see the DPKI paper.

The basic key management architecture of Sovrin as a DPKI is illustrated in Figure 5.

Figure 5: The basic key management architecture of Sovrin DPKI

The “key” to this architecture is that neither keys, agent endpoints, or Sovrin identifiers provide any inherent correlation between an identity owner’s different Sovrin identities. Each identity and its corresponding agent endpoints and key(s) are distinct and, short of compromise or out-of-band information, cannot be correlated without permission from the identity owner. Also, the type of key and its associated controls can be different with each Sovrin identity; it is not necessary to have a completely homogeneous key architecture.

Key Discovery

Every distributed ledger technology (DLT) is a “cryptographic triple play”, i.e. the very definition of a DLT is a distributed database in which:

1. Every transaction is cryptographically **signed**.
2. Every transaction is cryptographically **chained** (with a hash of the previous transaction, whether individually or in a block).
3. Every transaction is cryptographically **replicated** (via a consensus protocol).

So cryptography is “baked in” to DLT distributed database much more deeply than it is in, say, DNS. While this does makes the distributed data structures more complex, it has the advantage that you get key discovery—the ability to locate and verify the public key of another identity owner—essentially for free.

For example, with Sovrin architecture, every CID record has an associated public (verification) key. The same is true of every DID record once the originating trust anchor publishes the associated CID. So all that is necessary to discover the current public key for any Sovrin identity is to look up the most recent DID or CID record. (“Most recent” is very important for purposes of key rotation and revocation, below.)

Key Rotation and Revocation

Any PKI must be able to deal with how identity owners can associate new keys with their identity under normal circumstances (key rotation) and especially if an old key has been compromised (key revocation). With Sovrin identity records, key rotation and revocation can be as simple as the identity owner writing a new Sovrin DID or CID identity record using the old key that replaces it with the new key. Of course, if a key has been stolen, the attacker may attempt to revoke it and replace it with a new key under the attacker’s control. This is one scenario where key recovery is required.

Key Recovery

In addition to key compromise, key recovery is also required when an identity owner has lost her keys, for example by loss of a device (if the owner’s Sovrin keychain only lived on a single device), or loss of an entire set of devices. In any of these cases, the identity owner no longer has access to the keychain and can no longer unlock her Sovrin identity or change any of her identity records.

For this contingency, Sovrin key management relies on a “web of trust” architecture as described in the *Sovrin Trust Framework* section. This requires the identity owner to designate their own set of **trustees**—anywhere from a single person or organization to dozens if necessary—that the identity owner trusts to assist in key recovery if and only if the identity owner asks for it. If that event occurs, a minimum number of the trustees (for example, 3 out of 5, or whatever **threshold** originally specified by the identity owner) must sign a new identity record transaction. The Sovrin validator nodes will verify this transaction against the most recent identity record before a specified **timelock** period.

The purpose of the timelock is to thwart an attacker who compromises an identity owner’s keychain and immediately tries to change the owner’s identity records, including her designated trustees, thus preventing key recovery. As long as the identity owner is able to trigger key recovery by her trustees within the holdback period, the identity owner will be able to recover control of her identity records and begin using a new set of keys.

This is one of many use cases where it is very useful for identity owners and others to be able to actively subscribe to specific events on the Sovrin ledger, such as updates to their own identity records. Subscriptions are one of the features of observer nodes, discussed above. Just like notifications from websites about password changes, notification of Sovrin identity record updates can alert identity owners to compromises in their Sovrin keychains or other possible attacks.

Sovrin Claims and Claim Management

The attributes associated with a Sovrin identity are called **claims**. This term originated with claims-based identity, a way of asserting digital identity that is independent of any particular system that needed to rely on it. The formal definition of a claim is:

> a statement that one identity owner, such as a person or organization, makes about itself or another identity owner

Since a primary purpose of Sovrin identity management is to enable identity owners to easily assert and prove attributes of their identities—personal data, credentials, awards, degrees, etc.—claims management is at the heart of Sovrin architecture.

Verifiable Claims Architecture

The core concept of how claims can be issued by one identity owner to a second identity owner and then presented to a third identity owner in a way that they can be cryptographically verified as genuine is captured in Figure 6 from a presentation by the W3C Verifiable Claims Task Force (VCTF). (Note that the "Inspector" role in this diagram is called a **relying party** in Sovrin architecture, and that a Holder may or may not be the identity owner described by a claim—it may also be a trustee of the identity owner, such as the parent or guardian for a child.)

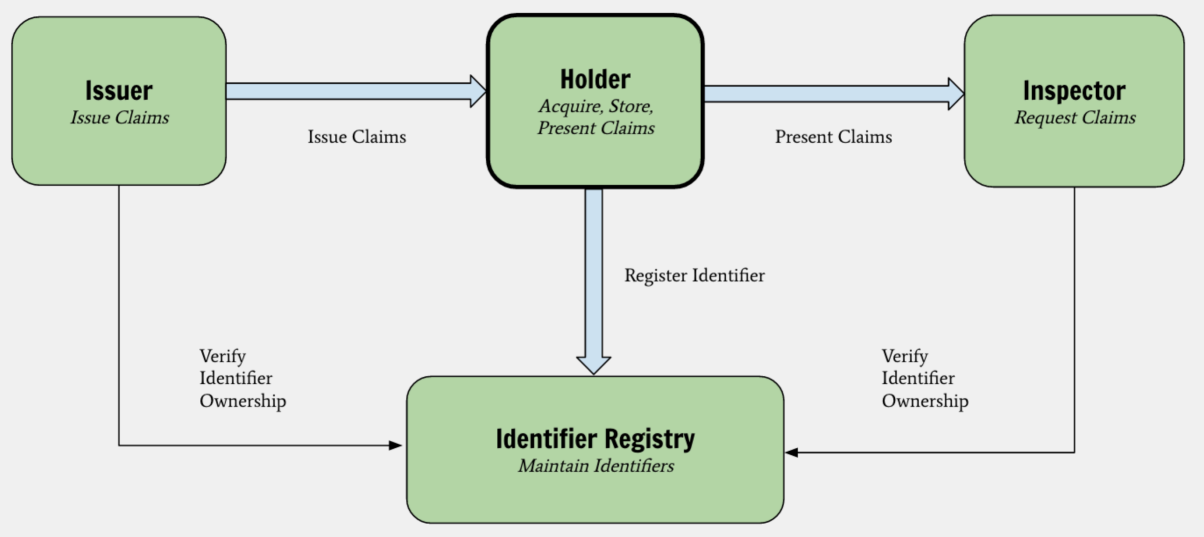

Figure 6: Verifiable claims architecture

The VCTF, led by JSON-LD pioneers Manu Sporny and David Longley, has spent four years laying the groundwork for verifiable claims to become an interoperable standard on the World Wide Web. The Verifiable Claims work is now on the verge of final approval as a W3C Working Group; further information will be in the next version of this white paper.

Public and Private Claims

For privacy reasons, Sovrin claims architecture distinguished between:

1. **Public (on-ledger) claims** are stored directly on the Sovrin ledger. Public claims may or may not be in plain text (see the next section), but regardless of whether they are in plain text or encrypted, the existence of the claims (and thus any correlation is it possible to derive from them) is publicly visible on the ledger. Public claims should NOT contain personally-identifiable information (PII), regardless of whether they are encrypted or not.

2. **Private claims** are claims that are stored off-ledger by the associated Sovrin agent, in a private container that is not publicly visible, searchable, or correlatable. Private claims are recommended for any claims that contain PII or whose posting on the public Sovrin ledger may raise a correlation risk.

Basic Sovrin Claim Types

Sovrin architecture supports five basic claim types. All of them store claims data as JSON objects, the standard data interoperability format for Sovrin.

1. **Cleartext claims** are directly readable, with no hashing or encryption. Public cleartext claims are intended for public identities with no expectation of privacy, e.g., claims about business and governmental identities that are a matter of public record (and which with Sovrin can be fully verified).

2. **Encrypted claims** contain an encrypted version of a cleartext claim, where the entire claim may be decrypted with a single symmetric or asymmetric private key.

3. **Hash signature claims** contain a specially encrypted tree of cleartext claims, where the identity owner can selectively reveal specific claims to specific relying parties.

4. **Proof of existence claims (aka POE claims or hash claims)** are simply hashes of digital objects that enable an identity owner to prove that a digital object existed at a point in time. POE claims are especially useful for proving consent as required under privacy regulations such as the EU General Data Protection Regulation (GDPR). When an identity owner gives consent to a relying party for a particular usage of personal data under GDPR, a hash of the consent receipt or link contract to which both parties agreed can be written to the ledger as a claim. The hash itself does not reveal any information about the consent—or even the relationship of the parties—but can be used by either party to prove consent was granted.

5. **Anonymous credentials** transmit claims information without actually containing either a cleartext or encrypted version of the claims data. Rather they are a cryptographic method of providing a proof about a claim. The classic example of an anonymous credential is a proof of age (i.e., “over 21”) that does not reveal the actual birthdate. See the Wikipedia article for more information.

Anonymous Credentials Architecture

Anonymous credentials (anoncred) are a major privacy-enhancing technology that have been the subject of extensive research and development by Microsoft (UProve) and IBM (Idemix) for over a decade. They have tremendous potential for preserving privacy while enabling the sharing of highly sensitive or correlate-able information.

The unique opportunity Sovrin represents is for anoncred to "go mainstream", i.e., to become a standard feature of a global public utility for identity management. The next version of this white paper will contain greater detail about Sovrin's anonymous credentials architecture. This subject also deserves its own white paper, which is on the roadmap for the Technical Governance Board. See the *Roadmap* section below.

Claim Semantics: Schemas, Ontologies, and Dictionaries

Another critical aspect of interoperability is claim semantics, i.e., how do Sovrin participants understand and process the contents of the JSON objects that carry the payload of Sovrin claims?

Again, as a global public utility expressly for the purpose of identity management and permissioned claims sharing, Sovrin represents an unprecedented opportunity to forge consensus around longstanding problems in semantic interoperability. Whether at the level of JSON schemas, RDF ontologies, or XDI dictionaries, this area will be another focus of the Technical Governance Board—and also the subject of another white paper on the TGB roadmap.

The Sovrin Trust Framework

As described in The Inevitable Rise of Self Sovereign Identity, the Sovrin Trust Framework is the core legal and technical governance document for the Sovrin Identity Network. It defines the rights and responsibilities of each of the participants in the network, including Sovrin stewards, trust anchors, and identity owners (see the Sovrin Glossary). Once completed, it will serve as the common contract between all Sovrin participants.

Most importantly, the Sovrin Trust Framework defines how the "permissioning" of a public permissioned ledger will actually work. This section provides a high-level description of this trust model.

Web of Trust Architecture

Many of the contributors to Sovrin have participated in the Rebooting the Web of Trust workshops organized by Christopher Allen, co-author of the SSL /TLS protocol. The genesis of these workshops was the 25th anniversary of PGP (Pretty Good Privacy), one of the original catalysts for the concept of a web of trust. To quote from the Wikipedia article:

> *[The web of trust] decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such).*

In fact, the "web of trust" model is essentially the way trust works between most people and organizations in the real world: we each choose who we trust, and then we use those relationships to decide about who else to trust. There is no one authority or hierarchy who decides who can trust whom. Yet this organic process still produces an overall social, economic, and legal trust fabric that in many (but not all) cases is strong enough to uphold modern society.

Seen from this perspective, the Sovrin web of trust model is simply an application of the same principle of decentralized peer-to-peer trust. From the standpoint of the Sovrin Foundation, whose responsibility is to maintain the integrity of the Sovrin ledger, this model only need to be applied to the selection of Sovrin stewards and trust anchors. For Sovrin identity owners and communities, this web of trust model may be extended to many other relationships and communities of any size, from families to clubs to nation states.

The web of trust model was also the inspiration for the Respect Trust Framework, the second trust framework listed with Open Identity Exchange, the international non-profit clearinghouse for trust frameworks. The Sovrin Trust Framework will build upon core principles of the Respect Trust Framework, including the concept of trust anchors.

Trust Anchors

When a web of trust model is applied to a P2P network, it has been mathematically proven that it can be defeated if there is no way to enforce a limit on the number of nodes (participants) in the network. This is due to the Sybil attack, also known as sock puppets. To quote from the Wikipedia article (emphasis added):

> *In a Sybil attack, the attacker subverts the reputation system of a peer-to-peer network by creating a large number of pseudonymous identities, using them to gain a disproportionately large influence. A reputation system's vulnerability to a Sybil attack depends on how cheaply identities can be generated,* ***the degree to which the reputation system accepts inputs from entities that do not have a chain of trust linking them to a trusted entity****, and whether the reputation system treats all entities identically.*

The highlight explains why the network needs a subset of members that are considered by other members to be trusted—the trust anchors. A more direct definition (again from Wikipedia):

> *A* ***trust anchor*** *is an authoritative entity for which trust is assumed and not derived.*

The Sovrin Trust Framework will define the qualifications for selection and ongoing verification of trust anchors, and thereby establish the basis for maintaining a sustainable, organic web of trust of permissions on the Sovrin network.

Trust Levels

In the design of the Respect Trust Framework, new members of the network (i.e., newly provisioned Sovrin identity owners) earn trust by building reputation with other members. The Sovrin Trust Framework will define how new members can graduate through a small set of trust levels to earn their way to trust anchor status, thereby organically growing the Sovrin web of trust.

Roadmap

Public Peer Review

This white paper has explained the fundamentals of Sovrin technical architecture as developed by the initial contributors to the Sovrin Foundation and the Sovrin open source code. We are confident this is a solid foundation for an open global public utility, but like any foundation, it is just the start of what we aspire to build together.

The next stage is to begin open public peer review of this architecture and the Sovrin open source code. We invite you to engage in this process both online via sovrin.org (where we will be publishing information about chat channels) and at the following industry events:

- The third Rebooting the Web of Trust Workshop; San Francisco, October 19-21.
- The Internet Identity Workshop (Oct 25-27); Computer History Museum, Mountain View, CA, October 25-27.
- The first meeting of the W3C Verifiable Claims Working Group; Computer History Museum, Mountain View, CA, October 27-28.

Growing the Technical Governance Board

A second key step is recruiting additional members to join the Sovrin Foundation Technical Governance Board. This board will be chaired by Jason Law, co-founder and CTO of Evernym, original developers of the Plenum consensus protocol and the Sovrin open source code that has been gifted to the Sovrin Foundation. If you are interested in participating in the TGB, please reach out to us via the Contact Page on sovrin.org.

Moving from Sandbox to Production

The third major step, expected to take the fourth quarter of 2016, is to grow and test the current sandbox version of the Sovrin network until the Technical Governance Board and the Board of Trustees agree that it is ready

Complete the V1 Sovrin Trust Framework

The final step before the Sovrin network can go into full production mode with Sovrin identities and claims is to complete and approve the first version of the Sovrin Trust Framework, since this will become the operating contract between the Sovrin Foundation, Sovrin stewards, Sovrin trust anchors, and all other Sovrin identity owners. This will be a highly collaborative effort between the Sovrin Trust Framework Working Group, the Technical Governance Board, and the Board of Trustees. Please contact us if you would like to be directly involved in this effort.

Additional White Papers

An ongoing activity of the Technical Governance Board will be to not only maintain this white paper, but publish new ones on all core aspects of Sovrin technical architecture and design. Our motto will be “publish early and often”, both to maximize transparency and public review.

The current list of additional white papers on our roadmap includes:

- **The Legal Foundations of Sovrin** exploring the legal implications of self-sovereign identity and how it will map to the requirements of different jurisdictions around the world.
- **Sovrin Security and Privacy By Design** will go deeper into key decisions about Sovrin security, privacy, and trust architecture.
- **Anonymous Credentials on Sovrin** will explain the vision and roadmap for integrating and promoting anoncred with Sovrin.
- **Semantic Interoperability on Sovrin** will tackle the hard issues with establishing common semantics and why Sovrin claims should us a standard semantic graph model.

Please contact us via the Contact Page on sovrin.org if you would like to see other topics covered by Sovrin Foundation white papers or if you would like to contribute to one.