A Self-Sovereign Identity Architecture

by Manu Sporny and David Longley

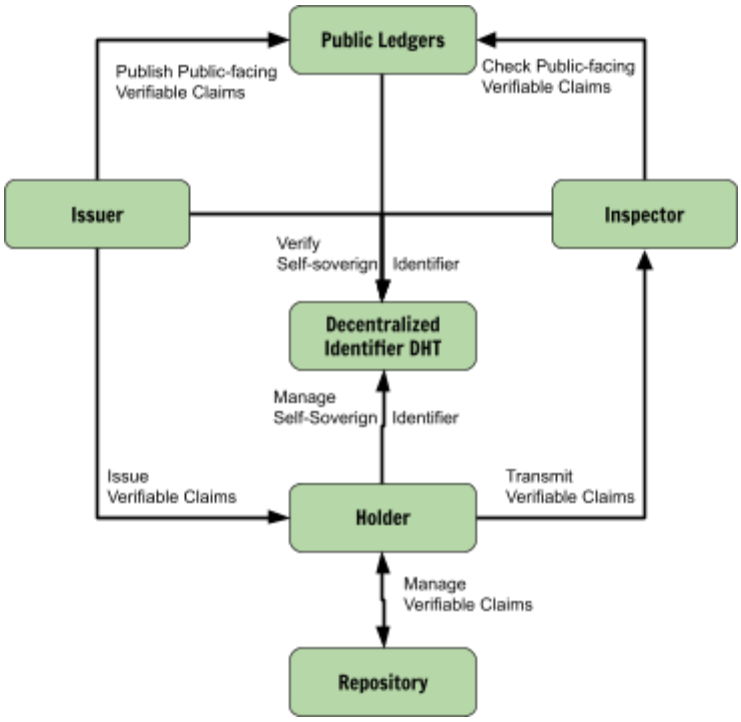
A Topic Paper from the ID2020 Design Workshop

Overview

There is currently no widely used self-sovereign, privacy-enhancing standard for expressing and transacting verifiable claims (aka: credentials, attestations) via the Web. It is asserted that being able to do this must be one of the fundamental building blocks of identity on the next generation Web. There is work that is being incubated at the World Wide Web Consortium (W3C) in the Credentials Community Group as well as the Verifiable Claims Task Force, a part of the W3C Web Payments Interest Group, to identify what a self-sovereign architecture would look like for the Web as well as a number of technical requirements of such an architecture. This topic paper outlines that proposed architecture and its primary components and actors.

A Proposed Self-Sovereign Identity Architecture

The following diagram provides an overview of the self-sovereign identity architecture.



Architecture Components

Issuers provide verifiable claims to people and organizations (e.g. ETS, Pearson, Walmart, Verisys, Target, NACS (retailers), New Zealand Government, Bloomberg, and IMS Global member companies).

Repositories store and curate verifiable claims on behalf of people and organizations (e.g. Accreditrust, Verisys, Bill and Melinda Gates Foundation, and Deutsche Telekom).

Inspectors request verifiable claims from people and organizations in order to give them access to protected resources (e.g. Walmart, Target, NACS (retailers), Bloomberg, New Zealand Government, Education Institutions (IMS Global member companies), Financial Institutions, and customers of Issuers today).

Holders receive verifiable claims from issuers, store them at repositories that they trust, and provide them to inspectors in order to get access to protected resources (e.g. Citizens, Employees, Professionals, Aid Recipients, Legal Guardians, and Property Owners).

Public Ledgers store public-facing verifiable claims (e.g. proof of publication, proof of existence, proof of revocation, etc.).

Decentralized Identifier DHT stores self-sovereign identifier documents for the purpose of claiming self-sovereign identifiers, updating the basic management information associated with a self-sovereign identifier, and verifying the cryptographic authenticity of information associated with a self-sovereign identifier.

Topic Discussion

Is this architecture the best design for a self-sovereign identity ecosystem?

A whitepaper that follows on page four provides more information on the desired technical goals, capabilities, and requirements.

THIS PAGE INTENTIONALLY BLANK

SELF-SOVEREIGN IDENTITY ARCHITECTURE WHITEPAPER FOLLOWS

A Self-Sovereign Identity Architecture

by Manu Sporny and David Longley
with content from the W3C Verifiable Claims Task Force and Credentials Community Group

A White Paper from the ID2020 Design Workshop

Abstract

There is currently no widely used self-sovereign, privacy-enhancing standard for expressing and transacting verifiable claims (aka: credentials, attestations) via the Web. It is asserted that being able to do this must be one of the fundamental building blocks of identity on the next generation Web. There is work that is being incubated at the World Wide Web Consortium (W3C) in the Credentials Community Group as well as the Verifiable Claims Task Force, a part of the W3C Web Payments Interest Group, to identify what a self-sovereign architecture would look like for the Web as well as a number of technical requirements of such an architecture. This whitepaper outlines that architecture and its technical requirements.

The Problem

There is currently no widely used self-sovereign, privacy-enhancing standard for expressing and transacting verifiable claims (aka: credentials, attestations) via the Web. As a result, these problems are a reality today:

- There is no standard that makes it easy for users to assert their verifiable qualifications to a service provider (e.g. I am over the age of 21, I am a citizen of country X, I have an account at Bank Y, my loyalty card number is Z, I am a Chartered Financial Analyst, etc.). As a result, manual input and fraud on the Web are higher than desired.
- In existing attribute exchange architectures (like SAML, OpenID Connect, Login with SuperProviderX, etc.), users, and their verifiable claims, do not independently exist from service providers. This means users can't easily change their service provider without losing or fragmenting their digital identity. This leads to vendor lock-in, identity fragility (duplication, confusion, and inaccuracy), reduced competition in the marketplace, and reduced privacy for all stakeholders.
- There is no interoperable standard capable of expressing and transmitting rich verifiable claims that cuts across industries (e.g., finance, retail, education, and

> healthcare). This leads to industry-specific solutions that are costly, inefficient, proprietary, and inhibit users' ability to manage their digital identities in a cohesive way.

This problem statement has been circulated among 83 commercial, government, and non-governmental organizations with 48 responses demonstrating that there is broad support for the problem statement and goals listed in this document.

Terminology

Listed below is special terminology used throughout this document:

Claim - A statement made by an entity about a subject.

Verifiable Claim - A claim that can be cryptographically proven to be non-repudiable, authentic, and unaltered by bad actors.

Credential - A set of claims that refer to a qualification, achievement, personal quality, aspect of an identity such as a name, government ID, preferred payment processor, home address, or university degree typically used to indicate suitability.

Identity - A set of credentials that can be used to identify a particular entity such as a person, organization, concept, or device. An entity may have multiple identities associated with it.

Self-Soverign Identity - A form of identity that attempts to balance transparency, fairness, and support of the commons with protection for the individual. A list of requirements for self-sovereign identity are outlined in Christopher Allen's blost post about the subject.

Ecosystem Participants

The envisioned self-sovereign identity architecture has at least the following actors:

Issuers provide verifiable claims to people and organizations (e.g. ETS, Pearson, Walmart, Verisys, Target, NACS (retailers), New Zealand Government, Bloomberg, and IMS Global member companies).

Repositories store and curate verifiable claims on behalf of people and organizations (e.g. Accreditrust, Verisys, Bill and Melinda Gates Foundation, and Deutsche Telekom).

Inspectors request verifiable claims from people and organizations in order to give them access to protected resources (e.g. Walmart, Target, NACS (retailers), Bloomberg, New Zealand Government, Education Institutions (IMS Global member companies), Financial Institutions, and customers of Issuers today).

Holders receive verifiable claims from issuers, store them at repositories that they trust, and provide them to inspectors in order to get access to protected resources (e.g. Citizens, Employees, Professionals, Aid Recipients, Legal Guardians, and Property Owners).

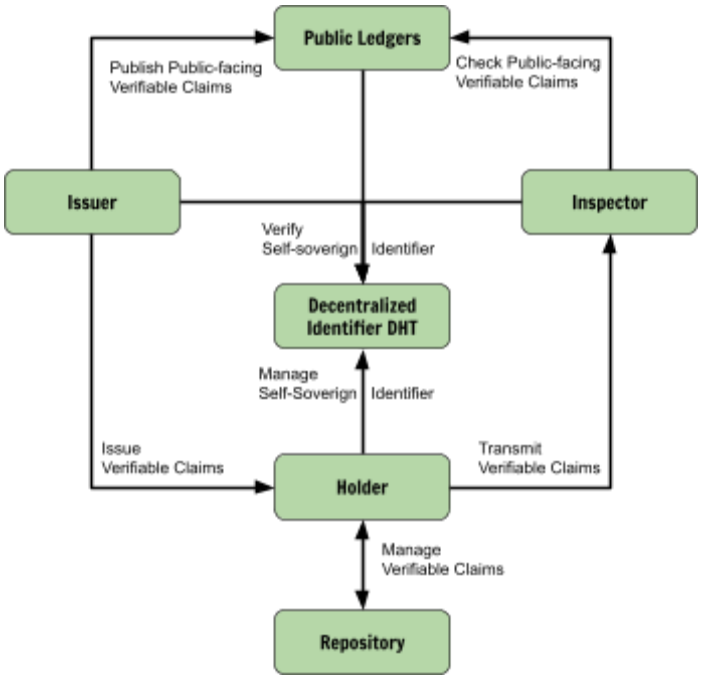
Public Ledgers store public-facing verifiable claims (e.g. proof of publication, proof of existence, proof of revocation, etc.).

Decentralized Identifier DHT stores self-sovereign identifier documents for the purpose of claiming self-sovereign identifiers, updating the basic management information associated with a self-sovereign identifier, and verifying the cryptographic authenticity of information associated with a self-sovereign identifier.

Subjects are entities that a verifiable claim is about.

An Overview of the Self-Sovereign Identity Architecture

The following diagram provides an overview of the self-sovereign identity architecture.



Desired Goals

In order to address the entirety of the Problem Statement identified by this document, at least the following four architectural goals must be met:

Goal 1: Create a standard way for users to assert their verifiable qualifications to a service provider, producing benefits such as:

- Enhancing website usability by removing the need to manually enter verifiable claims.
- Improving the detection of fraud, such as false claims and identity theft, by establishing a standard way to cryptographically verify 3rd party claims.

Goal 2: Ensure that users and their claims can be independent from service providers, producing benefits such as:

- Improving operational efficiency, by reducing operating costs (for example), for verifiable claim issuers and inspectors as a result of a common set of technology for expressing and verifying claims.
- Reducing vendor lock-in by ensuring that verifiable claims are portable from one claims repository to another.
- Enhancing some aspects of privacy and unlinkability for the subject of a verifiable claim.

Goal 3: Ensure that there is an interoperable standard capable of expressing verifiable claims that cuts across industries, producing benefits such as:

- Reusability of the machine-readable language that expresses verifiable claims (aka vocabularies) so that a single vocabulary may suit the needs of a broad set of stakeholders, and
- Extensibility of the vocabularies so that a particular industry vertical may build extensions on top of existing vocabularies to suit their industry-specific needs.
- Composability of verifiable claims to express an aspect of one's identity in a granular way.

Goal 4: Re-use existing attribute-exchange protocols or create a new protocol that is capable of transmitting the data in a privacy-enhancing way, producing benefits such as:

- Transferability of verifiable claims from issuer to repository to inspector.

- Portability of verifiable claims at the instruction of the holder from one repository to another
- Increased competition among issuers, repository, and inspector services.

Note: The “new protocol” portion of Goal 4 is controversial and thus was not included in the Verifiable Claims survey mentioned previously in this document.

Desired Ecosystem Capabilities

There are a number of desirable capabilities for the ecosystem:

- Verifiable Claims are self-sovereign. This refers to an architecture where:
    - Holders broker the transmission of their information between issuers and inspectors.
    - Holders receive and store verifiable claims from issuers through an agent that the issuer does not need to trust.
    - Holders provide verifiable claims to inspectors through an agent that inspectors needn't trust; they only need to trust issuers.
    - Verifiable Claims are associated with self-sovereign identifiers, not particular services; holders can decide how to aggregate verifiable claims and manage their own identities.
    - Holders can control and own their own identifiers.
    - Holders can control which verifiable claims to use and when.
    - Holders may freely choose and swap out the agents they employ to help them manage and share their verifiable claims.
    - Holders that share verifiable claims are not required to reveal the identity of the inspector to their agent or to issuers.
- A standard, machine-readable data model (with corresponding format) is needed for expressing verifiable claims that can be extended with minimal coordination.
- Independent issuance, storage, and cryptographic verification of verifiable claims.
- A standard mechanism for requesting verifiable claims.
- The ability to revoke previously issued verifiable claims.
- Web Browser APIs for storing and consuming verifiable claims.

Verifiable Claims Lifecycle Examples

In an attempt to provide an overview of how the actors in this ecosystem participate, the following use cases are provided. Note that there are more use cases with different

(automated) flows, but we've found the following non-automated use cases to resonate most with first-time readers:

Acquiring a self-soverign identifier:

1. The holder uses software to cryptographically claim a self-sovereign identifier on a public DHT or Ledger of some kind.
2. The holder, via their software, associates the self-soverign identifier with a credential repository in the public DHT or Ledger.

Issuing a verifiable claim:

1. The holder visits an issuer website.
2. The holder requests a verifiable claim from the issuer. The holder's self-sovereign identifier is made available to the issuer so that it may be associated with any issued verifiable claim (binding the claims to a identifier).
3. If the holder meets the issuer's requirements, the issuer issues a verifiable claim to the holder. The issuer may optionally place public information associated with the verifiable claim (identifier validity information, proof of publication, proof of existence, revocation information, etc.) into a public ledger.
4. The holder stores the verifiable claim in their preferred repository.

Inspecting a verifiable claim:

1. The holder visits an inspector website and attempts to access a protected resource.
2. The inspector requests a verifiable claim from the holder in order to grant access.
3. The holder, via their software, retrieves the appropriate verifiable claim from their repository and delivers it to the issuer.
4. The issuer checks the validity of the verifiable claim first through cryptographic validation and then through semantic validation. The issuer may optionally check information associated with the verifiable claim in a public ledger.
5. If the holder's verifiable claim meets the issuer's requirements, the holder is given access to the protected resource.

Architecture Benefits to Ecosystem Participants

A self-sovereign architecture provides the following benefits to ***issuers***:

- Level competitive playing field (not just a few super-providers)
- Ability to participate in a broader ecosystem resulting in common tooling to issue verifiable claims
- Avoidance of vendor-specific solutions and lock-in
- Potential for reduced infrastructure needs due to user-centric architecture

A self-sovereign architecture provides the following benefits to ***repositories***:

- Level competitive playing field (not just a few super-providers)
- Ability to participate in a broader ecosystem resulting in common tooling to store verifiable claims
- Higher-stakes verifiable claims being stored resulting in more value-added services

A self-sovereign architecture provides the following benefits to ***inspectors***:

- Ability to participate in a broader ecosystem resulting in common tooling to consume verifiable claims
- Richer set of verifiable claims to choose from, resulting in better understanding of the customer
- Increased ability and choice to trust authenticity of verifiable claims

A self-sovereign architecture provides the following benefits to ***holders***:

- No identity provider lock-in
- Digital claims that can be used in more than one location
- Ability to aggregate verifiable claims as cohesive digital identities
- Privacy-enhanced sharing mechanism
- Control of confidential information
- Elimination of repetitive input at websites
- Reduction in the need to input personally identifiable information (PII)
- Better usability for sites that need to collect data to perform checks (regulatory compliance)
- Cost-reductions through verifiable claim persistence and machine verifiability

Meeting the Requirements of Self-Sovereign Identity

This section analyses how the self-sovereign identity architecture meets the principles for self-sovereign identity proposed by Christopher Allen in his blog post about the subject:

Existence. *Users must have an independent existence. Any self-sovereign identity is ultimately based on the ineffable "I" that's at the heart of identity. It can never exist wholly in digital form. This must be the kernel of self that is upheld and supported. A self-sovereign identity simply makes public and accessible some limited aspects of the "I" that already exists.*

The architecture supports this principle by enabling users to have multiple digital identities that they fully control. Each digital identity can be an aspect of their "real identity".

Control*. Users must control their identities. Subject to well-understood and secure algorithms that ensure the continued validity of an identity and its claims, the user is the ultimate authority on their identity. They should always be able to refer to it, update it, or even hide it. They must be able to choose celebrity or privacy as they prefer. This doesn't mean that a user controls all of the claims on their identity: other users may make claims about a user, but they should not be central to the identity itself.*

The architecture supports this principle partially via the following:

- The Decentralized Identifier DHT ensures that user's control their identifiers.
- Portability of verifiable claims from repository to repository ensures that repositories cannot lock users into their service.
- A design requirement of the system is that user's must provide consent before their claims are used.

The architecture partially fails to support this principle because repositories may share their customer's data with 3rd parties without consent. It's an open question on whether or not all verifiable claims should be encrypted. If we do, we fully support this principle but then are unable to address fully automated use cases. We assert that this is a tension that all self-sovereign systems will have.

Access*. Users must have access to their own data. A user must always be able to easily retrieve all the claims and other data within his identity. There must be no hidden data and no gatekeepers. This does not mean that a user can necessarily modify all the claims associated with his identity, but it does mean they should be aware of them. It also does not mean that users have equal access to others' data, only to their own.*

The architecture supports this principle partially via the following:

- The Decentralized Identifier DHT ensures that user's control their identifiers.
- Portability of verifiable claims from repository to repository ensures that repositories cannot lock users into their service.

Transparency*. Systems and algorithms must be transparent. The systems used to administer and operate a network of identities must be open, both in how they function and in how they are managed and updated. The algorithms should be free, open-source, well-known, and as independent as possible of any particular architecture; anyone should be able to examine how they work.*

In theory, the Decentralized Identifier DHT will have multiple open source implementations if successful. That said, there will also be multiple closed source implementations. The best method to achieving this principle is to ensure that open standards are available early in the process and interoperability testing is a constant for the Decentralized Identifier DHT and the Ledger portions of the architecture.

Persistence. *Identities must be long-lived. Preferably, identities should last forever, or at least for as long as the user wishes. Though private keys might need to be rotated and data might need to be changed, the identity remains. In the fast-moving world of the Internet, this goal may not be entirely reasonable, so at the least identities should last until they've been outdated by newer identity systems. This must not contradict a "right to be forgotten"; a user should be able to dispose of an identity if he wishes and claims should be modified or removed as appropriate over time. To do this requires a firm separation between an identity and its claims: they can't be tied forever.*

The proposed architecture provides a firm separation between an identifier and its claims. The identifier is managed through the Decentralized Identifier DHT. The claims are managed via issuers. Identifiers managed via the Decentralized Identifier DHT are designed to be long-lived.

Portability. *Information and services about identity must be transportable. Identities must not be held by a singular third-party entity, even if it's a trusted entity that is expected to work in the best interest of the user. The problem is that entities can disappear — and on the Internet, most eventually do. Regimes may change, users may move to different jurisdictions. Transportable identities ensure that the user remains in control of his identity no matter what, and can also improve an identity's persistence over time.*

The proposed architecture ensures that identities are transportable:

1. The Decentralized Identifier DHT ensures that user's control their identifiers and thus the repositories associated with the identifier are under the control of the user.
2. Portability of verifiable claims from repository to repository ensures that repositories cannot lock users into their service and a design requirement of the protocol is such that a repository never knows if another repository is asking for data or not.
3. Porting data from one repository to another is a part of the protocol, and repositories that don't support it will eventually get bad marks from consumer reporting agencies.

Interoperability. *Identities should be as widely usable as possible. Identities are of little value*

if they only work in limited niches. The goal of a 21st-century digital identity system is to make identity information widely available, crossing international boundaries to create global identities, without losing user control. Thanks to persistence and autonomy these widely available identities can then become continually available.

The proposed architecture is being designed at a community group in the World Wide Web Consortium and utilizes the design of the Web, which places massive interoperability as a core requirement of any standard put through the W3C process.

Extensibility of the architecture, in the types of claims that can be expressed, is a cornerstone of the work. At present, Linked Data in the form of JSON-LD has been identified as the primary extensibility mechanism.

The participants in the design come from a variety of industries: technology, education, healthcare, government. The broad participation from different market verticals helps ensure that the system will be broadly interoperable and thus the identities should be as widely usable as possible.

***Consent**. Users must agree to the use of their identity. Any identity system is built around sharing that identity and its claims, and an interoperable system increases the amount of sharing that occurs. However, sharing of data must only occur with the consent of the user. Though other users such as an employer, a credit bureau, or a friend might present claims, the user must still offer consent for them to become valid. Note that this consent might not be interactive, but it must still be deliberate and well-understood.*

A design requirement of the architecture is that a user must provide consent before their claims are used. That said, a repository may act as a bad actor and share claims without user consent. The architecture supports consent as much as it can without additional regulations and law to prevent abuse.

***Minimalization**. Disclosure of claims must be minimized. When data is disclosed, that disclosure should involve the minimum amount of data necessary to accomplish the task at hand. For example, if only a minimum age is called for, then the exact age should not be disclosed, and if only an age is requested, then the more precise date of birth should not be disclosed. This principle can be supported with selective disclosure, range proofs, and other zero-knowledge techniques, but non-correlatibility is still a very hard (perhaps impossible) task; the best we can do is to use minimalization to support privacy as best as possible.*

The proposed architecture has the ability to atomize claims, meaning that it is possible for the disclosure of claims to be minimized to the point that only a single data point is shared. We do have a number of concerns around privacy, as the architecture is designed to be as privacy-enhancing as possible while delivering a robust set of commercializable properties. We agree with Christopher Allen's assertion that non-correlatability may be an impossible task and we should keep that in mind when discussing what is truly possible when we explore the privacy-enhancing aspects of the architecture.

Protection*. The rights of users must be protected. When there is a conflict between the needs of the identity network and the rights of individual users, then the network should err on the side of preserving the freedoms and rights of the individuals over the needs of the network. To ensure this, identity authentication must occur through independent algorithms that are censorship-resistant and force-resilient and that are run in a decentralized manner.*

The proposed architecture supports this design requirement as the Decentralized Identifier DHT is meant to be developed as an open standard, with open algorithms that are censorship-resistant and force-resilient.

Open Questions

We would like to enagage the ID2020 Design Summit (aka Rebooting Web of Trust Workshop II) to explore at least the following open questions:

1. Is this the simplest architecture for addressing the Problem Statement? If not, what can be removed?
2. Does this architecture provide strong commercial incentives to participate for all actors? Does this architecture provide strong commercial incentives to maintain and upgrade the shared DHT and Ledger infrastructure?
3. What would the best technical solution for the Decentralized Identifier DHT look like? Is it a DHT? Is it a Ledger? IPFS? Blockstack?
4. What would the most easily standardizable solution for the Decentralized Identifier DHT look like? Where is the best place to standardize that technology?
5. Are current cryptographic expression mechanisms up to the task? Are Linked Data Signatures of interest to the group?
6. Is a unified data model with different syntaxes the best approach?
7. Is the use of Linked Data or JSON-LD a non-starter? If so, how are we going to support the extensibility requirement?
8. What could a repository do to prevent migration to another repository?

9. Do we have a list of envisioned attacks on each part of the system that we could start documenting?
10. If the architecture fails, what components can be re-used in future attempts at solving the problem statement?