

Why the Internet is missing an identity layer—and why SSI can finally provide one

by Alex Preukschat and Drummond Reed

“The Internet was built without an identity layer.”

—Kim Cameron, Chief Architecture of Identity, Microsoft
The Laws of Identity, May 2005¹

What did Kim Cameron—Microsoft’s Chief Architect for Identity from 2004 until 2019—mean by that quote? What is an “identity layer?” Kim gives an answer in his groundbreaking series of essays called *The Laws of Identity*, published on his blog over a series of months in 2004 and 2005:

The Internet was built without a way to know who and what you are connecting to. This limits what we can do with it and exposes us to growing dangers. If we do nothing, we will face rapidly proliferating episodes of theft and deception that will cumulatively erode public trust in the Internet.

What Kim was saying is that when the internet was initially developed in the 1960s and 1970s by the U.S. military (sponsored by DARPA²), the problem it was designed to solve was how to interconnect **machines** to share information and resources across multiple networks. The solution—packet-based data exchange and the TCP/IP protocol—was so

¹ <http://www.identityblog.com/?p=352>

² The Defense Advanced Research Projects Agency

brilliant that it finally enabled a true “network of networks.” And the rest, as they say, is history.

What Kim was driving at, however, is that with the Internet’s TCP/IP protocol, you only know the **address of the machine** that you are connecting to. That tells you nothing about the **person, organization, or thing** controlling that machine and communicating with you.

This seems like a fairly easy problem to solve—after all, people and organizations built the Internet, and we control (or at least we think we do) all the “things” that are using it. So, how hard could it be to design a simple, standard way to identify the person, organization, or thing you are dealing with over the Internet?

The answer turns out to be: *very, very hard*.

Why? In a nutshell, the original Internet was not very big. The people using the network were mostly academic computer scientists. Most of them knew each other, and they all needed access to expensive machines and sophisticated technical skills to even participate. So even though the Internet was designed to be decentralized without single points of failure, early on it was effectively a relatively small club.

Needless to say, that has changed completely. There are now billions of people and multiple billions of devices on the Internet, and almost everyone is strangers. In this environment, the unfortunately truth is that there are many, many people who want to **deceive you** about who or what you are dealing with over the Internet. Identity (or the lack of it) is, in fact, one of the main sources of cybercrime.

Internet Identity Workshop

Shortly after Kim published his *Laws of Identity*, in the fall of 2005, a group of about 60 people working on all aspects of the Internet identity problem met at a workshop in Berkeley, California. It was organized by Kaliya Young (known online as Identity Woman³), Phil Windley, and Doc Searls. It was called, fittingly enough, the *Internet Identity Workshop (IIW)*, and its purpose was to see if a community of people working together could finally solve the problem of the Internet’s missing identity layer.

Many of the original attendees⁴ could regale you with stories about that first meeting and how it became the birthplace of OpenID 2.0, the first major protocol for “user-centric identity”. But the unanimous conclusion was that more meetings were needed—and more frequently than once a year. So it was agreed to start meeting every six months.

The Computer History Museum in Mountain View, California, was chosen as the location for the next meeting in the spring of 2006—and it has remained so ever since. This means roughly 200 people from around the world have been banging their collective heads against the wall since 2005 just to solve the problem of knowing and trusting who or what you are

³ <http://www.identitywoman.net/>

⁴ Contributors to this book who attended the first IIW include Drummond Reed, Phil Windley, Doc Searls, Kaliya Young and _____

dealing with over the Internet.

Each of the major evolutionary steps in Internet identity technology and protocols—SAML (Security Assertion Markup Language), OpenID Connect, OAuth (Open Authorization), UMA (User Managed Access), SCIM (Simple Cloud Identity Management), FIDO (Fast Identification Online)—has been either born or nurtured at IIW.

And what have been the results so far?

How bad has the problem become?

Recall the final sentence of Kim Cameron's 2005 prediction about the Internet's missing identity layer: *"If we do nothing, we will face rapidly proliferating episodes of theft and deception that will cumulatively erode public trust in the Internet."*

Despite all the efforts of the IIW community—and everyone else working on Internet identity—the lack of a breakthrough solution has proved Kim's prognosis true in spades. Never mind that by 2017, the average business user had to keep track of 191 passwords⁵ or that username/password management has become the most hated consumer experience on the Internet. That's just an inconvenience.

Where's the foul? The deeper damage is in cybercrime, fraud, economic friction, and the ever-growing threats to our online privacy.

The litany of statistics goes on and on:

- IBM President and CEO Ginni Rometty described cybercrime as "the greatest threat to every profession, every industry, every company in the world."⁶
- Global cybercrime damages are predicted to cost \$6 trillion annually by 2021.⁷
- Over 90% of American consumers believe they have lost control of how their personal information is collected and used by all kinds of entities.⁸
- In 2016, 3 billion Yahoo accounts were hacked in one of the biggest breaches of all time.⁹
- 63 percent of network intrusions are the result of compromised user passwords.¹⁰
- The Equifax breach has cost the company over \$4 billion in total.¹¹

⁵ <https://www.securitymagazine.com/articles/88475-average-business-user-has-191-passwords>

⁶ <http://www.forbes.com/sites/stevemorgan/2015/11/24/ibms-ceo-on-hackers-cyber-crime-is-the-greatest-threat-to-every-company-in-the-world/>

⁷ <https://www.comparitech.com/vpn/cybersecurity-cyber-crime-statistics-facts-trends/>

⁸ <http://www.pewresearch.org/fact-tank/2018/03/27/americans-complicated-feelings-about-social-media-in-an-era-of-privacy-concerns/>

⁹ <https://www.oath.com/press/yahoo-provides-notice-to-additional-users-affected-by-previously/>

¹⁰ <https://www.microsoft.com/en-us/cloud-platform/advanced-threat-analytics>

¹¹ <http://time.com/money/4936732/equifaxs-massive-data-breach-has-cost-the-company-4-billion-so-far>

- According to a 2014 study from CTRL-Shift¹², the cost of identity assurance processes exceeds £3.3 billion per annum in the UK alone.

Our failure to solve the Internet identity problem is reaching the breaking point. Either we fix it, or the very future of the Internet is in doubt.

The breakthrough: blockchain

Like many disruptive innovations, a breakthrough in digital identity has come from a very unexpected direction: cryptocurrency. When Satoshi Nakamoto first published *Bitcoin: A Peer-to-Peer Electronic Cash System* in October 2008,¹³ no one expected it could also inspire a fundamental transformation in how we think about identity and establish trust online.

Yet in fact identity and money have been very closely intertwined for centuries—a history explored in rich and entertaining detail in David Birch’s 2014 book *Identity is the New Money*.^{14, 15} So it is not surprising that by 2015, when Bitcoin’s decentralized blockchain model had started to capture the attention of industry and the world press, that it finally came to the attention of the Internet identity community.

Co-author Drummond Reed, an attendee at every one of the Internet Identity Workshops, tells the story this way:

Shortly before Spring 2015 IIW, a number of us in the space had become convinced by Christopher Allen, co-author of the original SSL protocol, that we should look closely at using blockchain technology to solve the Internet identity problem. At the opening dinner of that IIW I approached IIW co-founder Phil Windley about the idea, expecting to have a long conversation with him about it. All he said was, "Yes, let's do it."

So at that IIW, which is entirely an open-space conference, we organized five or six sessions over the three days just to talk about how we could use blockchain for identity. Interest was so strong that we put together an informal working group and began meeting weekly online for the next six months. At the Fall 2015 IIW we reported out on our findings, and there was so much enthusiasm that by the end of that IIW, a number of us were convinced we had to build it.

IIW was not the only identity community taking notice of blockchain. On December 16, 2015, the U.S. Department of Homeland Security Science & Technology division published a Small Business Innovation Research (SBIR) grant topic entitled: *The Applicability of Blockchain Technology to Privacy Respecting Identity Management*.¹⁶ It explained:

...recent innovations around crypto-currencies point to a potential answer to this

¹² <https://www.ctrl-shift.co.uk/insights/2014/06/09/economics-of-identity/>

¹³ https://en.wikipedia.org/wiki/Satoshi_Nakamoto

¹⁴ Read the chapter about money and identity for more details.

¹⁵ <https://www.amazon.com/Identity-Money-Perspectives-David-Birch/dp/1907994122>

¹⁶ <https://www.sbir.gov/sbirsearch/detail/867797>

dilemma [of Internet identity]. Of particular interest is the underlying technology of the 'bitcoin' crypto-currency, which is called the blockchain. The blockchain is in effect a common, public ledger, which utilizes cryptographic mechanisms to verify transactions and information in a decentralized manner.

The potential applicability of blockchain technology goes beyond crypto-currencies (which is simply an application built on top of that technology) to many other uses such as smart contracts, provenance and attribution, distributed validation of information, and more.

This SBIR topic is focused on determining and demonstrating if classic information security concepts such as confidentiality, integrity, availability, non-repudiation and provenance as well as privacy concepts such as pseudonymity and selective disclosure of information can be built on top of the blockchain to provide a distributed, scalable approach to privacy respecting identity management.

Here was a U.S. government agency—an agency that specializes in cybersecurity—proposing that the same principles of blockchain technology that power Bitcoin could, potentially, solve critical problems with the Internet's missing identity layer. And the reason: it could enable us to move from centralized to **decentralized** digital identity systems.

Since then under the U.S. Department of Homeland Security (DHS) Science & Technology Silicon Valley Innovation Program (SVIP) has made grants to various local and international organizations in support of the technologies covered in this book. There is a direct parallel between DARPA's bootstrapping of the decentralized Internet and DHS helping to bootstrap a decentralized digital identity infrastructure.

However this time the U.S. is far from the only government in the mix. In fact, most of the major world economic hubs are supporting the development of decentralized infrastructure. For example, the European Union is exploring decentralized digital identity via a number of initiatives,¹⁷ the Chinese government have made Blockchain technologies a national priority,¹⁸ and Korea has created public/private consortia specifically for decentralized identity.¹⁹

Why is decentralization so important?

¹⁷ See the International Association for Trusted Blockchain Applications <https://inatba.org/>, <https://www.eublockchainforum.eu/> or the European SSI Framework (eSSIF) <https://ssimeetup.org/understanding-european-self-sovereign-identity-framework-essif-daniel-du-seuil-carlos-pastor-webinar-32/>

¹⁸ <https://www.coindesk.com/president-xi-says-china-should-seize-opportunity-to-adopt-blockchain>

¹⁹ <http://didalliance.org/>

The three models of digital identity

This question is best answered by showing the evolutionary progression across three basic models of digital identity.²⁰

The centralized identity model

The first model, **centralized identity**, is the easiest to explain. It is the model we have long had with identifiers or credentials, for example government ID numbers, passports, identity cards, driving licenses, invoices, Facebook logins and so on. All of these are issued by centralized governments or trusted institutions like banks or telecom companies. In fact, it is so prevalent in the real world that the centralized model can be divided into two types:

- The Scandinavian model
- The Continental Identity model

In the *Scandinavian model* private companies (financial and telecom firms) provide a centralized digital identity service to interact with the government (TUPAS in Finland, BankID in Sweden, and so on.) In the *Continental model* in Europe, governments provide digital identity services to companies allowing interaction with their citizens.²¹

The centralized model is also the original form of Internet identity—and the one that in many cases we still use today. You establish an identity by registering an **account** (typically a username and password) with a website, service, or application. For this reason it is also called **account-based identity**.

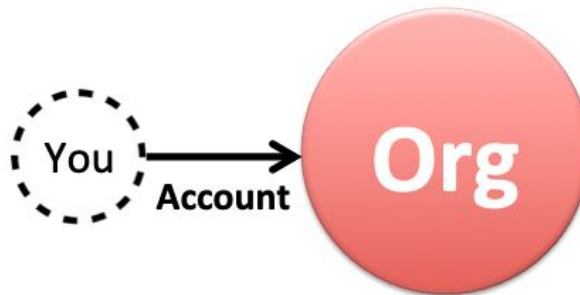


Figure 1.x: The relationship of an individual to a website (or application) under the Internet's original centralized, account-based identity model

In this diagram, "You" are a dotted circle because, in the world of centralized identity, the real "You" doesn't exist without an account in some centralized system. The real "You" gets to plug into the website, service, or application because the Org is lending you credentials that represent you with limited controls and permissions. At the end of the day, these

²⁰ <https://medium.com/evernym/the-three-models-of-digital-identity-relationships-ca0727cb5186>

²¹ All these centralized models are described in the excellent 2016 World Economic Forum report, *A Blueprint for Digital Identity* http://www3.weforum.org/docs/WEF_A_Blueprint_for_Digital_Identity.pdf

credentials belong to the Org.

Delete all your accounts at sites, services, and apps and “You” would disappear from the Internet completely.

That is only one of the many problems with centralized identity. Others include:

- The burden of remembering and managing all the usernames and passwords (and in some cases other multi-factor authentication tools such as one-time codes) falls entirely on **you**.
- Every site enforces their own security and privacy policies and they are all different (a classic example is the maddingly different rules about passwords—minimum length, special characters allowed, and so on.)
- None of your identity data is portable or reuseable anywhere (in fact users are warned to never reuse passwords at all).
- The centralized databases of personal data used to manage centralized identity are giant honeypots that have lead to some of the biggest data breaches in history.

The federated identity model

To alleviate some of these problems, the Internet identity industry moved quickly to develop a new model called **federated identity**. The basic idea is simple: insert a service provider, called an **identity provider** or IDP, in the middle.

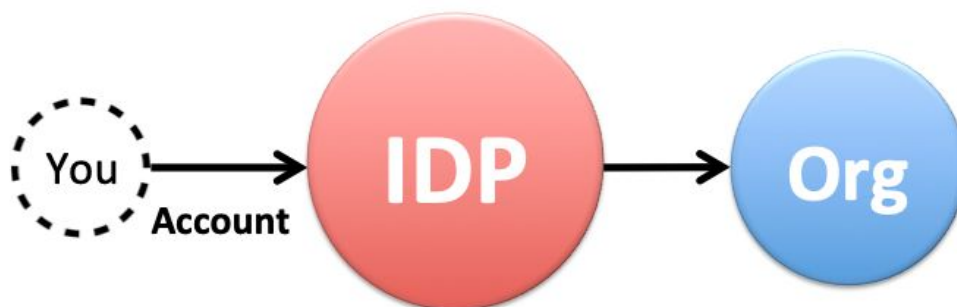


Figure 1.x: The three-way relationship involved in the federated identity model

Now, you can just have one identity account with the IDP, and they in turn can log you in and share some basic identity data with any site, service, or app that uses that IDP. The collection of all the sites that use the same IDP (or group of IDPs) is called a **federation**. Within a federation, each of the Orgs is often called a **relying party**, or RP.

Three generations of federated identity protocols have been developed since 2005—SAML, OAuth, and OpenID Connect. And they have all had some real success. Using these protocols, SSO (Single Sign-On) is now a standard feature of most corporate intranets and extranets.

Federated identity also started to catch on in the “consumer Internet”, where it began to be called **user-centric identity**. Using protocols like OpenID Connect, **social login** buttons from Facebook, Google, Twitter, LinkedIn, etc. are now a standard feature on many consumer-facing websites.

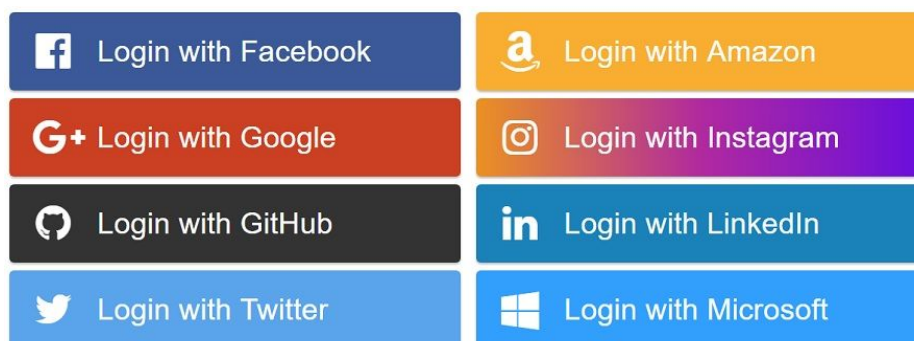


Figure 1.x: Examples of the proliferation of “social login” buttons to try to ease the pain of Internet identity for average mortals

Despite all the work that has gone into federated identity since 2005, unfortunately it has still failed to provide us with the Internet’s missing identity layer. There are numerous reasons:

- There isn’t one IDP that works with all sites, services, and apps. So users need accounts with multiple IDPs, and pretty soon they start forgetting which IDP they used with which site, service, or app.
- Because they have to serve so many sites, IDPs must have “lowest common denominator” security and privacy policies.
- Many users—and many sites—are uncomfortable with having a “man in the middle” of all their relationships, particularly being able to surveil users login activity across multiple sites.
- Large IDPs represent some of the biggest honeypots for cybercrime ever created.
- IDP accounts are no more portable than centralized identity accounts. If you leave an IDP like Facebook or Twitter, all those account logins are lost.
- Lastly, due the security and privacy concerns, IDPs are not in a position to help users securely share some of their most valuable personal data, e.g., passports, government identifiers, health data, financial data, etc.

The decentralized identity model

The **decentralized identity model** inspired by blockchain²² technology first began surfacing in 2015. Since then it has accelerated rapidly, assimilating new developments in cryptography and decentralized systems and spawning new decentralized identity standards such as Verifiable Credentials (VCs) and Decentralized Identifiers (DIDs) that we will explain in more detail in Part 2.

However the most important difference in this model is that it is no longer account-based. Instead it returns to a direct relationship between you and another party—only now the two of you are **peers**. Neither of you “controls” the relationship with the other. This is true whether the other party is a person, an organization, or a thing.

²² FIDO (Fast IDentification Online) uses a hybrid approach where connections are peer-to-peer, but key management is performed centrally by the FIDO Alliance rather than by a blockchain.

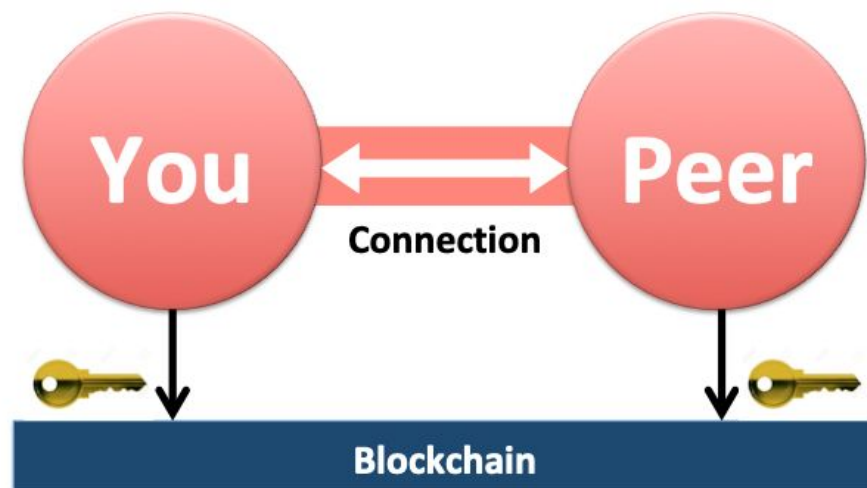


Figure 1.x: The peer-to-peer relationship that is enabled by the decentralized identity model—returning people to direct, private connections secured by public/private key cryptography

In a peer-to-peer relationship, neither of you has an “account” with the other. Rather you both share a **connection**. Neither of you fully “owns” this connection. It is like a string that you both are holding—if either one of you let go it will drop. But as long as you both want it, the connection will persist.

Peer-to-peer connections are inherently decentralized because any peer can connect to any other peer anywhere—that is exactly how the Internet itself works. But how does this become an identity layer? And why does it need blockchain technology?

The answer lies in **public/private key cryptography**—a way of securing data via mathematical algorithms based on cryptographic keys held by each party. Instead of using blockchain technology for cryptocurrency, identity management uses it for **decentralized public key infrastructure (DPKI)**. In the next few chapters we’ll go into this in greater detail, but in essence blockchain technology and other decentralized network technologies can give us a strong, decentralized solution for:

1. Exchanging public keys directly to form private, secure connections between any two peers.
2. Storing some public keys on public blockchains in order verify the signatures on **digital identity credentials** (aka **verifiable credentials**) that peers can exchange to provide proof of real world identity.

Ironically, this means the best overall analogy for the decentralized identity model is in fact *exactly the way we prove our identity to each other every day*: by getting out our wallet and showing the credentials we have obtained from other trusted parties.

With decentralized digital identity, we just do this with *digital* wallets, *digital* credentials, and *digital* connections.



Figure 1.x: The essence of decentralized digital identity: moving the utility and portability of physical identity credentials to our digital devices

Why "self-sovereign"?

As the decentralized digital identity model started to catch on, it quickly developed the moniker **self-sovereign identity (SSI)**. The origins of this term are further explained in a landmark essay on SSI by Christopher Allen explains included in the final section of this book. But why did a term so loaded with connotations start spreading so quickly—to the point where it is now used by top industry analyst firms²³ and leading digital identity conferences²⁴ around the world?

Such rapid adoption is highly unusual for a term that some people, some companies, and even some governments are uncomfortable with. In 2018, no less an industry luminary than Kim Cameron himself wrote a blog post entitled, *Let's find a more accurate term than 'Self-Sovereign Identity'*.²⁵ In it he says:

...the term "Self-Sovereign" seems to create a lot more confusion than understanding.

He even quotes himself, at the 2018 ID2020 conference on digital identity for the humanitarian sector, as saying:

The term 'self-sovereign' identity makes me think of hillbillies on a survivalist kick.

Others in the industry—including some of the authors of this book—agree with Kim that the term has serious issues. And yet it seems to keep spreading.

Why? What is it about the term that is so "sticky" (to use the term popularized by Chip and

²³ <https://blogs.gartner.com/blog/category/all/?c=self-sovereign-identity>

²⁴ The European Identity Conference, hosted every spring by the EU analyst firm Kuppinger Cole, has a track called "Self-Sovereign Identity".

²⁵ <https://www.identityblog.com/?p=1693>

Dan Heath in *Made to Stick*).²⁶

Let's start with the term "sovereign". This is not a word most of us use in everyday speech, so it has some cachet by itself. By definition it is a powerful word—sometimes used as a direct synonym for "king". To quote from the Wikipedia article:²⁷

*The roles of a sovereign vary from **Monarch** or **Head of state** to head of municipal government or head of a chivalric order. As a result, the word sovereign has more recently also come to mean independence or autonomy.*

The other frequent connotation is **sovereign nation** or **sovereign state**—one that "is neither dependent on nor subjected to any other power or state".²⁸

²⁶ <https://smile.amazon.com/Made-Stick-Ideas-Survive-Others-dp-1400064287/dp/1400064287/>

²⁷ <https://en.wikipedia.org/wiki/Sovereign>

²⁸ https://en.wikipedia.org/wiki/Sovereign_state

Add the word “self” in front of it and suddenly the meaning stands out: “a person that is neither dependent on nor subjected to any other power or state”.

That is clearly the source of the “survivalist” connotation that Kim Cameron referred to. In fact the Dictionary.com definition of “self-sovereignty” is:

the quality or state of being sovereign, or of having supreme power or authority; the status, dominion, power, or authority of a sovereign; royal rank or position; royalty.

Given that, what individual would not want to be “self-sovereign”? Conversely, what government might not have serious concerns about its citizens flocking en masse to a new technology called “self-sovereign”?

But of course the term we’re discussing is not “self-sovereign”, but “self-sovereign identity”. When you apply that term to a person, following our logic above, it would literally translate to:

A person’s identity that is neither dependent on nor subjected to any other power or state.

Ah-ha. Now we can finally understand why the term has become so popular so fast. When it comes to expressing one’s personal identity, there are many individuals around the world who would find that definition very attractive. Ironically, there are also many self-sovereign nations who would gladly help endow their citizens with the power of self-sovereign identity. As we will see in our use-case chapter about Canada, *governmental identity does not compete with self-sovereign identity*—in fact the two are highly complementary. Each enhances the other.

But we can also see why the term might be controversial—why, despite its power, it sometimes gets in the way of understanding the value of the decentralized identity model.

In fact there is one more issue with the term that often causes misunderstanding: **self-sovereign identity is not just for people**. It is also for organizations and things—literally *anything* that needs identity on the Internet.

This is why, as authors, we have a simple suggestion that we will follow in this book. As the name of any technology becomes popular, it is frequently reduced to an acronym—a shorthand. TCP/IP, DNS, URL, SSL, FIDO—many of us use these terms every day without even knowing (or caring) what they stand for.

In short, **the acronym becomes the term**. And that’s what we suggest: throughout this book, we will use the acronym *SSI* as simple, neutral, memorable name for this new model for digital identity.²⁹

²⁹ Co-author Alex Preukschat named his webinar series SSI Meetup for this very reason. <https://ssimeetup.org/>

Why is SSI so important?

In many ways, answering this question is the whole point of this book. The chapters in front of you will lay out how and why SSI will affect almost everything we do on the Internet—day in and day out. Some of these changes may actually be as deep and profound as the Internet itself was in the 1980s and 1990s—and the Web was after that.

Many of us today take those two technological advances for granted. Yet if you stop to think about it (if you were even alive back then), the work lives, social lives, and even political lives of billions of people have been radically transformed by the Internet and the Web. If that seems like an exaggeration, stop to consider that *seven out of the ten most valuable companies in the world today* would not even exist without the Internet and the Web.³⁰

Forecast (.calloutHead)We predict that the impact of SSI technology—and the uncounted new patterns of trusted interactions it will enable across all walks of life—will be equally profound.(.calloutStyle)

The fundamental reason is grounded in what we just explained about the term “self-sovereign identity”: it represents a **shift in control**. We started out trying to find a solution to the Internet’s missing identity layer; what we ended up discovering was that to solve those problems we needed a shift in control from the **centers of the network**—the many “powers that be”—to the **edges of the network**—where all of us exist and interact as peers.

³⁰ <https://www.statista.com/statistics/263264/top-companies-in-the-world-by-market-value/>

This shift in control is wonderfully captured in this diagram from Tim Bouma, author of the chapter on Identity in Canada:

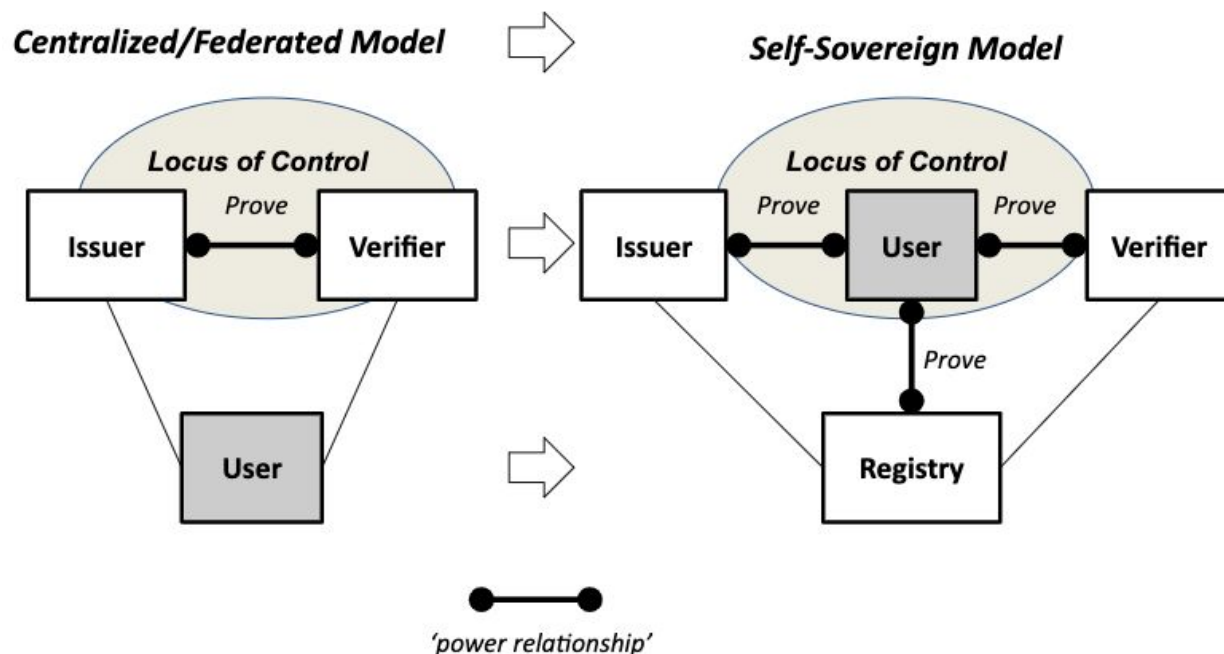


Figure 1.x: The shift in control that happens in the transition from centralized or federated identity models to the self-sovereign model—the model that truly puts the individual at the center.

In the centralized and federated identity models, the locus of control is with the issuers and verifiers in the network. In the decentralized self-sovereign identity models, the locus of control shifts to the individual identity owner, who can now interact as a full peer with everyone else.

This highlights a critical factor about SSI that is not often true of other technological advances—**it is about more than technology**. It has important business, legal, and social dimensions as well. Up to this point we have referred to SSI as if it were a single movement—one driven by the communities like IIW (described above), Rebooting the Web of Trust (RWOT),³¹ MyData³² and the W3C Decentralized Identifier³³ and Verifiable Credentials Working Groups.³⁴

However, as SSI becomes more mainstream we predict that different flavors and visions of SSI will be implemented. These differences will depend on the needs, wants and priorities of different communities implementing SSI. The devil will be in the detail about how different SSI architectures are designed to be interoperable—and there will be contrary views on

³¹ <http://www.weboftrust.info/>

³² <https://mydata.org/>

³³ <https://www.w3.org/2019/did-wg/>

³⁴ <https://www.w3.org/2017/vc/chapter.html>

what can and cannot be considered SSI.

Market drivers for SSI

One way to understand the possible differences in SSI architectures is by looking at what is driving demand for them. We can classify these into three broad categories:

1. Business efficiency and customer experience
2. Resistance to the surveillance economy
3. The sovereign individual movement

The market for business efficiency and customer experience market is focused on security, cost savings, and convenience.

As we will discuss extensively in Chapter 4, this is the primary market demand driving SSI in its early stages. Corporations, governments, universities, NGOs—they all want to improve the security of data, reduce costs by improving workflows, and be more competitive by offering their customers a better user experience.

This is primarily a disruption to the existing Identity and Access Management (IAM) marketplace, and like most disruptive technologies, it will give rise to new companies, new business models, and new subsegments within the IAM market.

Resistance to the surveillance economy is a reaction to the prevailing business model and tactics of some of the most dominant companies on the Internet today.

This is the market driver for certain governments, privacy-conscious individuals and a select number of corporates who want to do more than just ride the next wave of technological innovation and growth—they also want strategically weaken the business model of the global aggregators, resellers and distributors of personal data. Demand in this segment is a mix of ideology (privacy advocates for example), consumer sentiment and strategic positioning. Certain governments such as the European Union with its General Data Protection Regulation (GDPR) are leading this movement because it puts them on a more level playing field with the Internet giants, particularly as exponential technology changes continue to disrupt the “old world”.

The sovereign individual movement is driven by people who want to take back more control over their lives and data.

Perhaps the best way to describe this SSI market driver is that *it aims to do for decentralized identity what Bitcoin aims to achieve for decentralized money*. Permissionless cryptocurrency technologies like Bitcoin aim to create fully decentralized economies that do not rely on any central parties for their operation. Individuals in this market segment want to apply the same philosophy to digital identity, opting for SSI architectures that maximize decentralization. To understand their motivations, we recommended reading *The Sovereign*

*Individual*³⁵ by James Dale Davidson and William Rees-Mogg. Ironically, it was published in 1997, well before Bitcoin or decentralization became a mainstream subject. However reading the book today makes its authors look prescient about the decentralization movement—a subject we will explore in greater detail in Part 3.

Because the biggest initial driver of SSI adoption is the first category of business efficiency and customer convenience, let's look at a few examples in specific market segments—some of which will also shed more light on the other two market drivers.

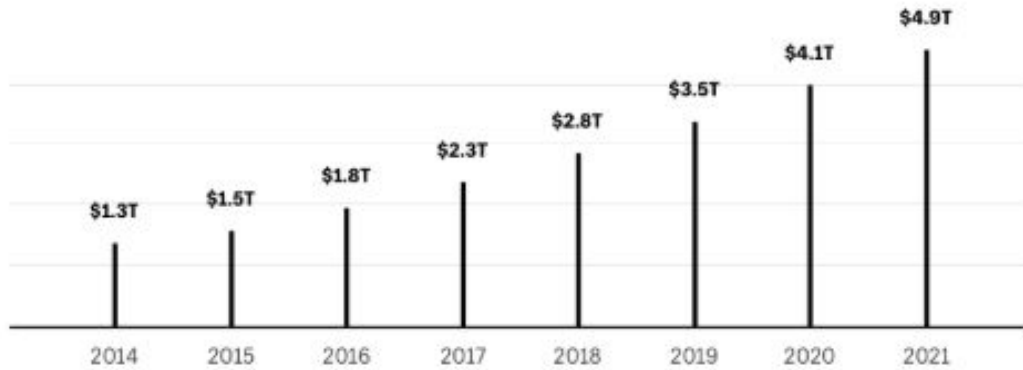
³⁵ https://www.goodreads.com/book/show/82256.The_Sovereign_Individual

Ecommerce

This chart from The Global Ecommerce Playbook shows the staggering growth of ecommerce.³⁶

Retail ecommerce sales worldwide

2014 to 2021 by trillions of USD



Data via eMarketer (Statista)

Every one of those dollars represents an electronic transaction over the Web, and every one of those transactions involves a digital identity of some kind—most of them centralized and some of them federated.

What happens when every consumer is equipped with an SSI digital wallet with which they can:

- Have near-instant passwordless registration and login at any SSI-enabled website or service?
- Be assured of strong security and privacy of every transaction?
- Automatically be warned if the site or service they are connecting to is not able to present its own trustworthy digital credentials?
- Provide payment directly out of their own digital wallet without having to fool with “checkout”, third-party wallet providers, or external payment gateways?
- Automatically maintain their own private personal log of digital receipts—and be able to provide proof of their purchases to any merchant or recommendation engine of their choice?

Not only will the average consumer’s experience of ecommerce be transformed, but the amount of friction that SSI will wring out of the global digital economy could be measured in **the hundreds of billions of dollars.**

³⁶ <https://www.shopify.com/plus/guides/global-ecommerce?itcat=plusblog&itterm=global-ecommerce-statistics>

Banking and finance

According to Citi's 2018 Mobile Banking Study, almost one-third of all U.S. adults are now using mobile banking. It's the most popular mobile app after social networking. And when it comes to millennials, the number is almost two-thirds.³⁷

Nearly all of this mobile banking activity is from dedicated apps provided by banks, credit unions, and other financial institutions directly to their customers. Some of these are award-winning for their usability, security, and privacy features. But they are still dedicated apps with their own logins and passwords that work with just a single financial service provider.

What happens when individuals gain the superpowers of an SSI digital wallet that:

- Can work with any financial institution that supports SSI—and have strong instant passwordless authentication with all of them?
- Can both strongly prove their identity and store and exchange their money from the same wallet using the same connections?
- Can digitally provide the trusted credentials necessary to pass KYC ("Know Your Customer") and AML ("Anti-Money Laundering") checks required of every financial institution?
- Can digitally share all the information—all third-party verified—necessary to apply for a loan or mortgage in seconds?
- Can do single-party or multi-party digital signatures to authorize important transactions—up to millions of dollars—with full cryptographically-protected audit trails?

These breakthrough benefits are not fictional at all—they are being worked on already today. For example, the global credit union industry has formed a consortia called CULedger whose first initiative is to introduce the world's first global digital credential of credit union membership.³⁸

³⁷

<https://www.prnewswire.com/news-releases/mobile-banking-one-of-top-three-most-used-apps-by-americans-2018-citi-mobile-banking-study-reveals-300636938.html>

³⁸ <http://www.culedger.com/>

Healthcare

Practice Fusion, the largest cloud-based electronic health record (EHR) provider in the United States, provides the following statistic about EHR adoption:

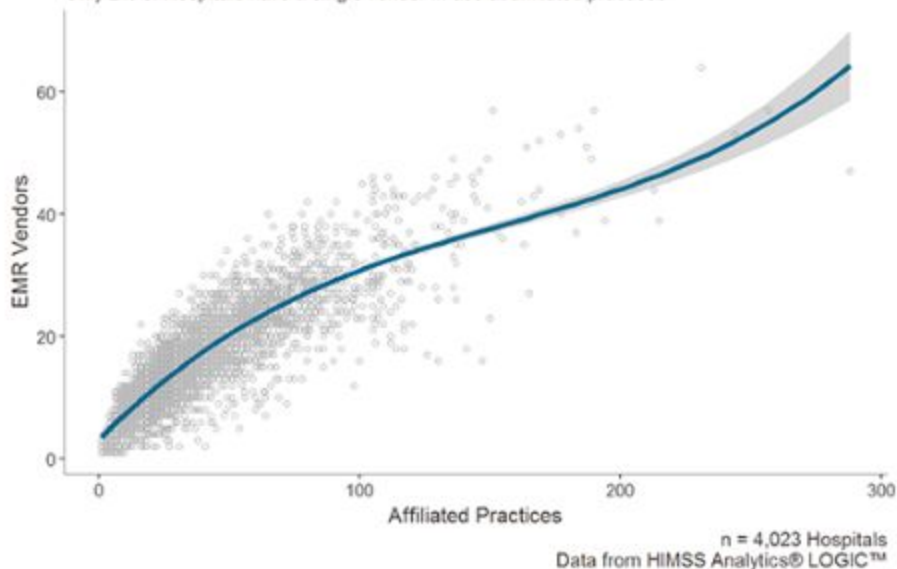
*Less than a decade ago, nine out of ten doctors in the U.S. updated their patients' records by hand and stored them in color-coded files. By the end of 2017, approximately 90% of office-based physicians nationwide will be using electronic health records (EHRs).*³⁹

Now, by contrast, Healthcare IT News provides this graphic about the state of EHR interoperability showing the reality of how easy it is to actually move an EHR from one doctor to another:⁴⁰

The average hospital has 16 disparate EMR vendors in use at affiliated practices

75% of hospitals are dealing with 10+ disparate outpatient vendors

Only 2% of Hospitals have a single vendor in use at affiliated practices



Healthcare IT News sums up the problem this way:

*The thorny matter of interoperability in healthcare, as it is or has historically been in other industries, is almost all-consuming among technology vendors and their clients. Indeed, a big part of the problem is exactly how many EHR companies are out there and, more specifically, the average number of platforms hospitals are running today.*⁴¹

³⁹ <http://www.usatoday.com/story/news/nation/2014/03/19/stateline-electronic-health-records/6600377/>

⁴⁰ <https://www.healthcareitnews.com/news/why-ehr-data-interoperability-such-mess-3-charts>

⁴¹ <https://www.healthcareitnews.com/news/why-ehr-data-interoperability-such-mess-3-charts>

What's the solution? Healthcare IT News is very clear what is required (emphasis added):

*Achieving interoperability among different EHR platforms is so difficult, in fact, that the Centers for Medicare and Medicaid Services working with the Office of the National Coordinator for Health IT, the federal agency charged with leading public and private healthcare organizations toward interoperability, essentially retooled the meaningful use EHR incentive program to focus on enabling **a more unified view of patient data**. Health IT shops across America, meet "promoting interoperability."*

*"There were three steps originally. Get as many hospitals and medical groups to purchase a viable EHR, then to meaningfully use that EHR, and the direction for the third step was to focus on qualitative value and quality measures," Icenhower added. "Now they're saying instead of that, '**we skipped the step where the patient is the center of the universe and their data is spread across different systems,**' so ONC shifted to **focusing on the patient**."⁴²*

Keep in mind, all of this is in the context of healthcare IT systems for doctors, hospitals, and medical institutions. It doesn't even contemplate yet how the patient could actually participate in being "the center of the universe" of their healthcare data. How much easier would the whole problem of EHR portability be if patients used their own SSI digital wallets to:

- Instantly obtain copies (either on their phone, or securely stored in a private cloud) of their EHR records immediately after any medical procedure?
- Securely and privately share their EHR in seconds with the doctors and nurses of their choice?
- Provide secure, legally valid, auditable consent for medical procedures—for themselves, family members, dependents, etc.—directly from their smartphone or other networked device?
- Have a lifetime history of vaccinations, allergies, immunities, etc. available in a verifiable electronic record to share in seconds—in person or remotely—with schools, employers, doctors, nurses or anyone who needs to verify it?

And we haven't even touched yet how your personal EHR might be used by your own apps on your own devices for you own healthcare. Or the impact of being able to securely and anonymously share your medical data with universities and medical researchers who can use it to advance the state of public health⁴³ for all of us.

⁴² <https://www.healthcareitnews.com/news/why-ehr-data-interoperability-such-mess-3-charts>

⁴³ Dive into these topics in the chapters about Health and supply-chain management in the pharmaceutical industry.

Travel

Anyone who has done international travel knows the fear in the pit of your stomach when you deplane and take the (often very long) walk to customs. The questions flood your mind: “How long will the line be?” “What if it takes hours?” “Will I miss my connecting flight?”

Governments and airport authorities around the world have been knocking their heads trying to figure out how to remove that particular friction from international commerce. Programs like Global Entry and CLEAR in the U.S., Nexus in Canada, Registered Traveller in the UK, and others around the world are dedicated systems designed to do a deep check on your identity credentials, enroll your biometrics, and thereby give you a fast track through customs.

But these programs cost tens of millions of dollars to set up, tens or hundreds of dollars to enroll in, and require dedicated facilities and personnel in every airport.

How much simpler will international travel become when you can travel with a smartphone holding an SSI digital wallet that can:

- Produce an instant proof of **any digital credential you possess** in a single QR code on their phone just like a mobile boarding pass?
- Have that proof be cryptographically verified in less than 5 seconds in any airport in the world using standard SSI QR scanners that will become commodities?
- **At the very same time receive a proof** into your digital wallet of having passed through security and/or customs at that particular airport—thereby building a private, verifiable audit trail of your travel that you alone can prove at subsequent checkpoints in your trip?
- Do all of this with cryptographic proofs that do not disclose anything more than the information required to meet government regulations at every border or checkpoint?
- Have all the travel documents you need—airline tickets, train tickets, hotel reservations, dinner reservations—flow automatically into your digital wallet so they are always there when you need them?

It might sound more like the world of Harry Potter than our world today, but this particular SSI magic carpet ride is in fact the goal of a growing number of international consortia and government agencies around the world. However, the SSI world has a number of challenges to tackle to reach its full potential.

Major challenges to SSI adoption

As authors, we believe SSI is a transformational step for the Internet that is already in motion. However we do not believe it is without major challenges. At a recent event called “The Future of Digital Identity” hosted by Citi Ventures, Vinod Baya, Director & Head of Emerging Technology, pointed out that there were three major challenges to the adoption of SSI:⁴⁴

⁴⁴ <https://www.citi.com/ventures/perspectives/opinion/digital-identity.html>

1. Building out the new SSI ecosystem
2. Decentralized key management
3. Offline access

Building out the new SSI ecosystem

SSI is a paradigm shift in our approach to digital identity. Some of its benefits can be realized by individual companies or communities, but the network effects will only be experienced when multiple industries, governments, and other ecosystems start accepting each others' credentials. This in turn depends on achieving real interoperability between the essential components of SSI as we will discuss in Chapter 2 and more deeply in Part 2. For example, we will need digital credentials to work across different digital wallets from different vendors—some of which will also integrate digital money (fiat or cryptocurrency for example). While this infrastructure is being built, it is still not mature, and there is much work to be done before it will be ready for Internet scale.

Decentralized key management

As we explained above, the key to SSI is literally **keys**—cryptographic key pairs where the SSI identity holder holds the private keys in his/her own digital wallet. Loss of those private keys would be tantamount to complete loss of the holder's digital identity. Key management has always been the Achilles heel in the adoption of cryptography and public key infrastructure (PKI). In fact it has been so difficult that many experts believe it can only be handled by large enterprises and centralized service providers such as banks or government agencies.

The growth of cryptocurrencies has already led to some advancements in decentralized key management, and the rise of SSI is leading to much more research in this area. We believe this is one of the primary hurdles to the ultimate market success of SSI, which is why we cover this subject in detail in Part 2.

Offline access

SSI is based on digital credentials shared over a digital network. Yet there are many situations where we need to be able to prove our identity without having Internet access or a digital device. For example a Canadian Mounted Police might need to verify a driver's license in the Far North where access is simply not available. So SSI solutions need to be able to work offline or with intermittent or indeterminate connectivity. This major engineering challenge is being tackled by SSI practitioners but is far from a solved problem.

Exploring SSI with this book

Our goal in this chapter was to introduce you to the basic idea of SSI and give you a solid understanding of why we've reached a watershed in the evolution of Internet identity. The rest of this book is designed to help you deepen and broaden that understanding across the many uses cases that SSI fulfills, the industries it might transform, and the disciplines it impacts.

We will do that not only through our own voices, but also through the voices of some of the leading SSI experts from all over the world. They will each be sharing their own perspectives on all aspects of SSI—not just the technology, but the business and legal implications, the social impact, and even the philosophy.

Our goal in every case will be to bring you specific examples of how SSI will be used so you can ground your understanding in how it can be applied in your own work, family, company, school, industry, city, or country.

Here is the structure of the rest of the book:

The three remaining chapters in **Part One** will cover:

- **Chapter 2:** *Basic Building Blocks of SSI*—digital credentials, wallets, digital agents, decentralized identifiers, blockchains, and governance frameworks.
- **Chapter 3:** *Example Scenarios: How SSI Works*—seven examples of how the building blocks are put together to solve hard problems of online trust
- **Chapter 4:** *The SSI Scorecard: Major Features and Benefits of SSI*—5 categories summarizing the 25 key benefits of SSI infrastructure.

We recommend reading these chapters sequentially as they apply to anyone interested in SSI, regardless of your whether your focus is technical, product, business, or policy.

In Part Two we will dive into SSI technology for those readers who want to understand more deeply how it works. While these chapters do not go all the way down to the code level, they do cover all aspects of SSI architecture, and should provide a solid technical foundation for architects, developers, system administrators, and anyone who wants to understand the underlying elements of the SSI “stack”.

- **Chapter 5:** SSI architecture—the big picture
- **Chapter 6:** Basic cryptography for SSI
- **Chapter 6:** Verifiable credentials
- **Chapter 7:** DIDs (Decentralized Identifiers)
- **Chapter 8:** Portable digital wallets and decentralized key management
- **Chapter 9:** SSI governance frameworks

In Part Three, we'll broaden the focus to use cases for SSI that cross traditional industry boundaries and encompass larger technological, legal, social, or political infrastructure. It will explore how the decentralization technologies powering SSI are rooted in even larger shifts of philosophy, society, and culture. In this part we will discuss the various points of

view—historical, political, sociological—on what is and is not considered SSI and why. We believe this part is highly relevant for everyone, but if your focus is primarily on SSI technology you can choose to skip it.

- **Chapter 10:** Controlling your identity with open source
- **Chapter 11:** Cypherpunks: the origin of decentralization
- **Chapter 12:** Identity for a peaceful society
- **Chapter 13:** Centralization vs decentralization believers
- **Chapter 14:** The evolution of the SSI community
- **Chapter 9:** Identity is money

In Part Four, we will look at how SSI will impact different categories of business and industry—with chapters written by individual experts in every one of these verticals. Each chapter will end with a SSI Scorecard summary of the impact of SSI on that particular vertical market. This part is especially relevant for technologists and product managers who need to convey to their business leaders why SSI matters to their business units—why it might be an opportunity, threat or disruption to your current business model.

- SIMPLIFYING AND IMPROVING OUR DIGITAL LIVES
 - Digital Banking just got faster and more trustworthy with SSI
 - Animal care, adoption, and accountability just became crystal clear
 - Self-sovereign identity for digital publication
- PERSONAL DATA CONTROL AND DIGITAL INCLUSION
 - NGOs and financial inclusion powered by SSI
 - Open democracy and electronic voting
 - Personal agency and the empowered customer
- EFFICIENT AND EFFECTIVE COMPANIES AND ORGANISATIONS
 - The energy sector gets a new relationship model
 - Supply-chain management powered by SSI in Pharma
 - SSI in Healthcare: A new era of trusted relationships between individuals, organizations and clinical things
- NEW JOBS AND NEW BUSINESS
 - Insurance firms reinvent themselves with SSI
 - Gaming and identity
- DIGITAL GOVERNMENT
 - Canada: The meaning of trust for identity

In the Appendix we have collected some of the landmark essays published about SSI that will help you to keep on expanding your vision of the SSI ecosystem.

- Christopher Allen on *The Path to Self-Sovereign Identity*
- Jason Law and Daniel Hardman on *The Three Dimensions of Identity*
- Phil Windley on *The Identity Metasystem*.

Our philosophy in composing this book is that successful developers, product managers, and business leaders need the capacity to look beyond their defined areas of responsibility to see the bigger picture, understand the cross-disciplinary currents, and assimilate major markets changes into their teams. SSI is one of those cases—an exponential technology that demands a mix of visions and skill sets to shape it into the future we want for the

world.

We hope you will enjoy learning about SSI as much as we love working in this space.

The basic building blocks of SSI

by Drummond Reed, Rieks Joosten, and Oskar van Deventer

As we explained in chapter 1, SSI is relatively new, having only emerged onto the Internet stage in 2016. At one level, SSI is a set of principles about how identity and personal data control should work across digital networks. At another level, SSI is a set of technologies that build upon core concepts in identity management, distributed computing, blockchain or Distributed Ledger Technology (DLT), and cryptography.

In many cases these core concepts have been established for decades. What's new is how they are put together to create a new model for digital identity management. The purpose of this chapter is to quickly familiarize you with these seven basic building blocks from a conceptual and technical point of view before we show how they are applied to different example scenarios in chapter 3.

1. Verifiable credentials (aka digital credentials)
2. Issuers, holders, and verifiers
3. Digital wallets
4. Digital agents and hubs
5. Decentralized identifiers (DIDs)
6. Blockchains
7. Governance frameworks (aka trust frameworks)

Note that we will go much deeper into these building blocks in *Part 2: SSI Architecture*.

Verifiable credentials

In chapter 1 we summarized that the essence of decentralized identity is “to move the utility and portability of physical identity credentials to our digital devices.” This is why the

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

concept at the very heart of SSI is **verifiable credentials**.

First, what exactly do we mean by the term “credential?” It obviously refers to the pieces of paper, plastic (or, in some cases, metal) that you carry around in your wallet to prove your identity, for example, driving licenses, government IDs, employment cards, credit cards, and so on (figure 2.1).

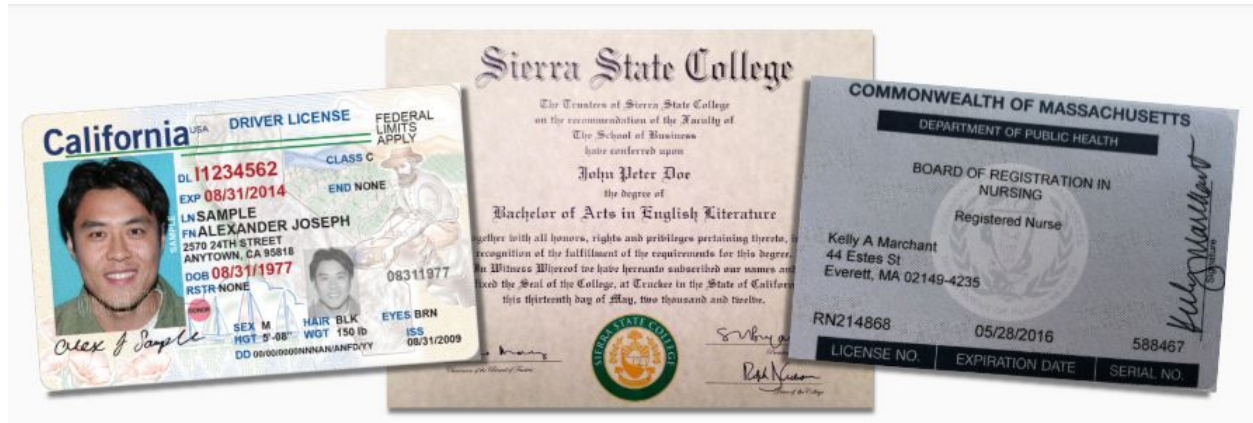


Figure 2.1: Common examples of credentials—not all of which fit in a traditional wallet.

But as this figure illustrates, not all credentials fit in your physical wallet. The term “credentials” extends to any (tamper-resistant) set of information that some authority claims to be true about you, and that enables you to convince others (who trust that authority) of these truths. For example:

- A birth certificate issued by a hospital or vital statistics agency proves when and where you were born and who were your parents.
- A diploma issued by a university proves you have an educational degree.
- A passport issued by a government of a country proves you are a citizen.
- An official pilot’s license proves you can fly a plane.
- A utility bill proves you are a registered customer of the utility that issued the bill.
- A power of attorney issued by the appropriate authority within a jurisdiction proves that you can legally perform certain actions on behalf of another person.

In one form or another, every credential contains a set of **claims** about the **subject** of the credential. Those claims are made by a single authority, which in SSI is called the **issuer** of the credential. The entity (person, organization, or thing) to whom the credential is issued, i.e, the one who will keep it in their digital wallet, is called the **holder** of the credential. Note that the subject of the credential is usually the same as the holder, but there are important exceptions to this rule-of-thumb that we will talk about later in the book.

The claims in a credential can state anything about the subject, such as **attributes** (age, height, weight, etc.), **relationships** (mother, father, employer, citizenship, or others), or

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

entitlements (medical benefits, library privileges, membership rewards, legal rights, and so on).

To qualify as a credential, the claims must be **verifiable** in some way. This means a **verifier** must be able to determine the following:

- Who issued the credential
- That it has not been tampered with since it was issued
- That it has not expired or been revoked

With physical credentials, this is typically accomplished through some proof of authenticity embedded directly in the credential itself (like a watermark, hologram, or some other special printing feature). It can also be done by checking directly with the issuer that the credential is valid, accurate, and current. But this manual verification process can be difficult and time-consuming — a major reason why there is a worldwide black market in falsified credentials.

This brings us to one of the fundamental advantages of verifiable credentials: using cryptography and the Internet, they can be **digitally verified** in seconds. This verification process can answer the following four questions:

1. Is the credential in a standard format that can be electronically processed by the verifier and does it contain the data the verifier needs?
2. Does it include a valid digital signature from the issuer (thus establishing its origin and that it has not been tampered with in transit)?
3. Is the credential still valid, that is, not expired or revoked?
4. If applicable, does the credential (or its signature) provide cryptographic proof that the holder of the credential is the subject of the credential.¹

¹ This feature can be supported by zero-knowledge proof (ZKP) cryptography.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Figure 3.2 is an illustration from Manu Sporny, co-editor of the Verifiable Credentials Data Model 1.0 specification, showing how a digital credential can be structured to answer these four questions. The first part of the verifiable credential package is a unique identifier for the credential—just like the unique number shown in the figure that appears on your driving license or passport. The second part is metadata describing the credential itself. In this example, this is the expiration date for the driving license credential. The third part is the claims contained in the credential—in the image, these are all the other items of data (name, date of birth, sex, hair color, eye color, height, weight).

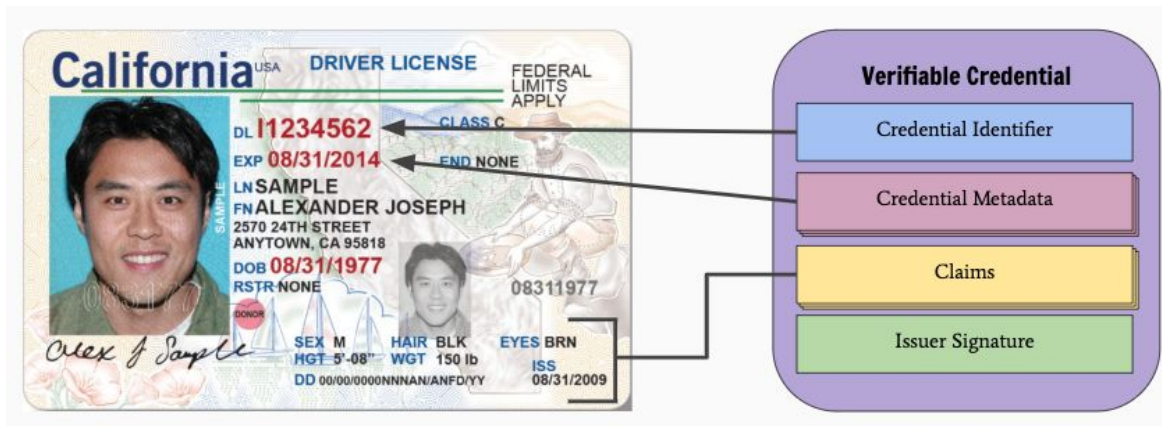


Figure 2.2: A mapping of three of the four core components of a W3C verifiable credential to the physical equivalent. The fourth component—the issuer’s digital signature—can only be produced in a physical credential by some form of watermark, hologram, or other hard-to-forge seal.

The equivalent of the credential subject’s signature (the one that appears below the person’s picture in the driving license) is a digital signature created using cryptography.

Issuers, holders, and verifiers

Figure 3.3 illustrates the terminology defined by the W3C Verifiable Claims Working Group² for the three primary roles involved with exchange of verifiable credentials.³

1. **Issuers** are the source of credentials—every credential has an issuer. Most issuers are organizations such as government agencies (passports), financial institutions (credit cards), universities (degrees), corporations (employment credentials), NGOs (membership cards), churches (awards), etc. However individuals can also be issuers, and so can things—for example a properly equipped sensor could issue a digitally-signed credential about a sensor reading.
2. **Holders/Provers** request verifiable credentials from issuers, hold them in the holder’s digital wallet (below), and present **proofs** of claims from one or more credentials when requested by verifiers (and approved by the holder). Although we most commonly think of individuals as holder/provers, holders/provers can also be organizations using enterprise wallets, or things in the sense of the Internet of Things (IoT).
3. **Verifiers** can be anyone—person, organization, or thing—seeking trust assurance of some kind about the subjects of credentials. Verifiers request proofs from holders/provers of one or more claims from one or more verifiable credentials. If the holder agrees (and the holder always has that choice), the holder’s agent will respond with a proof the verifier can then verify. The critical step in this process is verification of the issuer’s digital signature, typically accomplished using a **DID**, discussed in more detail later in this chapter.

² The Working Group has the name “W3C Verifiable Claims Working Group” even though the specification it is creating is called “Verifiable Credentials Data Model 1.0”. For more details on the terminology and standards for verifiable credentials, see the Verifiable Credentials Primer (<https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/topics-and-advance-readings/verifiable-credentials-primer.md>) by Manu Sporny and the W3C Verifiable Credentials Data Model 1.0 (<https://www.w3.org/TR/verifiable-claims-data-model/>).

³ From Appendix G of the Sovrin Glossary V2.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

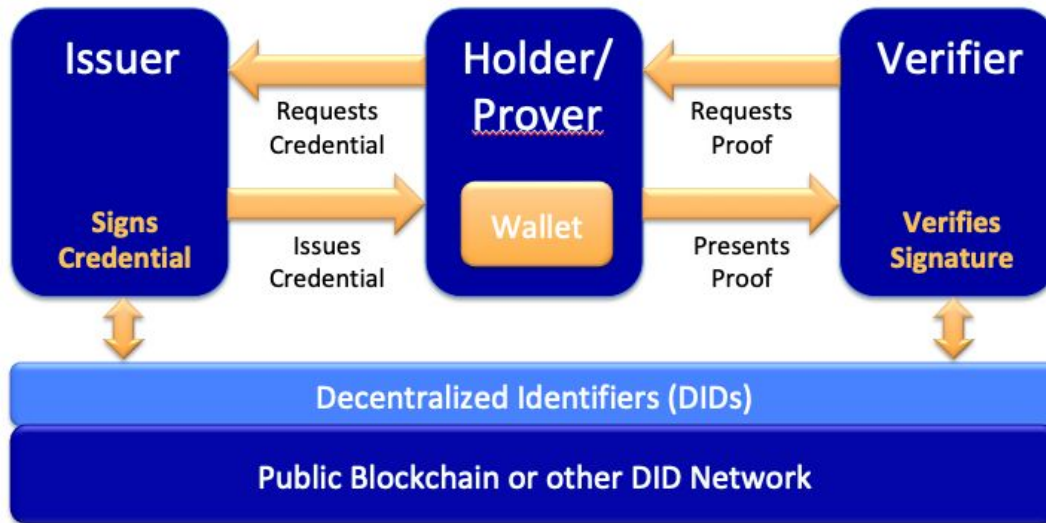


Figure 2.3: The primary roles involved with exchange of verifiable credentials. Digital signature verification is the part of the process enabled by public or private blockchain networks.

The relationship between issuers, holders/provers, and verifiers is often referred to as the *trust triangle* because it is fundamentally how human trust relationships are conveyed over a digital network. Figure 3.4 illustrates how **verifiable credentials only convey trust if the verifier trusts the issuer**. This doesn't mean the verifier needs to have a direct business or legal relationship with the issuer. It just means that the verifier is willing to make a business decision ("Will I accept this credit card?", "Will I board this airline passenger?", "Will I admit this student?") based on the trust assurance the verifier has in the issuer.

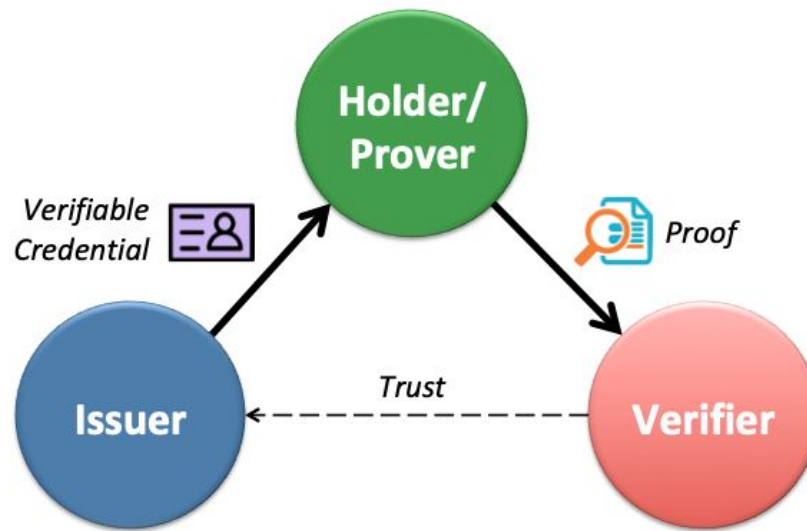


Figure 2.4: The “trust triangle” at the heart of all human trust relationships in the SSI ecosystem.

Note that the trust triangle describes only one side of a business transaction. In many business transactions, both parties will request information from the other. So in a single transaction, both parties will play the roles of holder and verifier. Also, many business transactions may result in a new credential issued from one party to the other—or even two new credentials, one in each direction.

An example of this is shown in figure 2.5, which describes the series of steps a consumer goes through buying an expensive holiday trip from a travel company:

- The consumer wants to verify that the travel company has insurance against bankruptcy.
- The travel site wants to verify that the consumer is older than 18.
- After payment, the travel site sends the tickets to the consumer.
- After the trip, the consumer confirms that he is a satisfied customer of the travel company.

Each of these pieces of information can be transmitted as a verifiable credential with a digital signature from the issuing party. It shows how both parties intermittently perform all roles in the trust triangle.

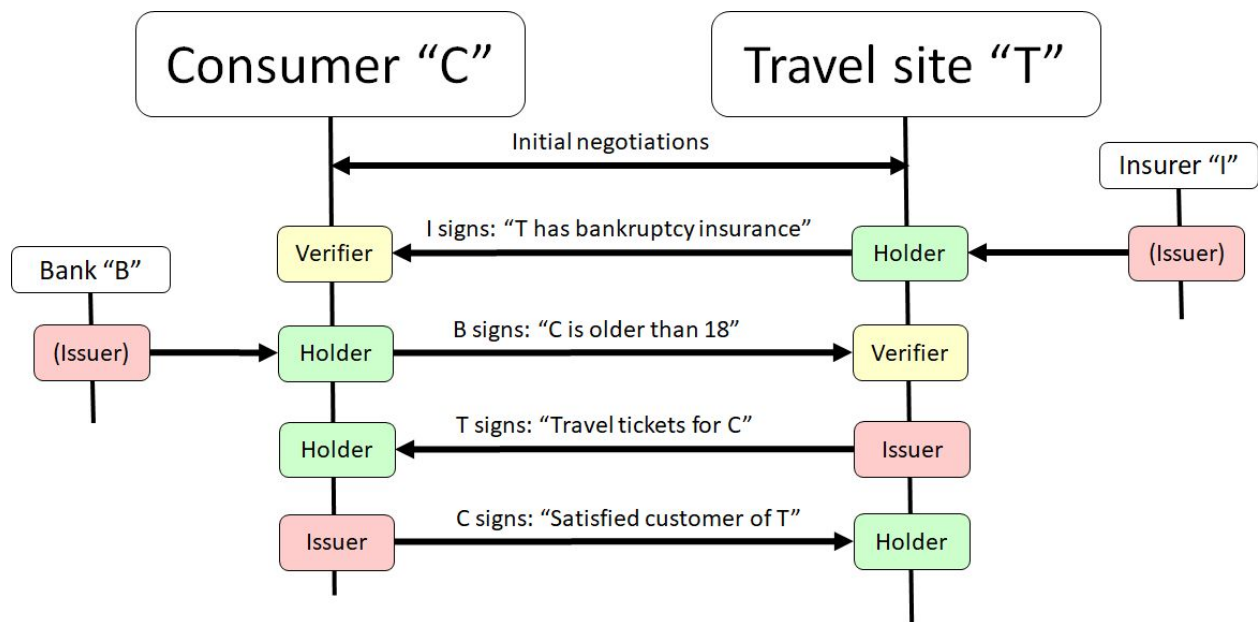


Figure 2.5: A typical multi-stage transaction where multiple verifiable credentials are used and created and both parties play the role of issuer, holder, and verifier.

Digital wallets

In the offline world, we typically store credentials in a physical wallet—it keeps them all in one place, protects them by keeping them close to our body, and makes them easy to carry around and access when we need them. The job of a digital wallet is no different:

1. Store your credentials, keys/keycards, bills/receipts, etc.
2. Protect them from theft or prying eyes.
3. Keep them handy—easily available and portable across all your devices.

Unfortunately, there have been so many attempts at digital wallets over the years that many developers stopped using the term. But two trends have brought the concept back into vogue. The first is **mobile wallets**, particularly those built-in to the operating system of smartphones to hold credit cards, tickets, boarding passes, and other typical financial or travel credentials.

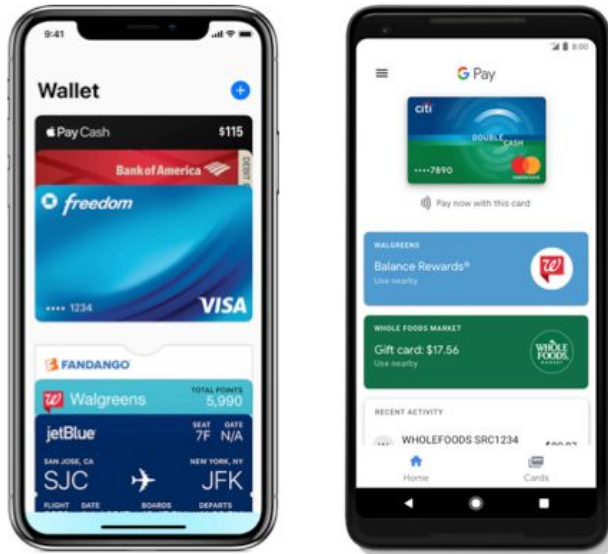


Figure 2.6: Two of the most popular smartphone mobile wallets—Apple Wallet and Google Pay. They are widely used because they come built-in to every Apple and Google phone.

The second is **cryptocurrency wallets**. Every purchaser of a cryptocurrency like bitcoin, ether, litecoin, and others needs either a **server-side wallet**, in which keys are stored by a broker such as Coinbase, or a **client-side wallet** (edge wallet) that is either a dedicated hardware wallet or an app that runs on one or more of the user's devices (smartphone, tablet, laptop, etc.).



Figure 2.7: A typical dedicated hardware wallet for cryptocurrencies. This one is called Ledger Nano S and has its own secure display.

For SSI, any of these general forms of a wallet will work. But there are also some key differences, including the following:

1. **An SSI digital wallet should implement open standards** for portable,

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

self-sovereign verifiable credentials and other sensitive private data.

2. **An SSI wallet works with a digital agent** to form connections and perform credential exchange (discussed in the next section).

For SSI to fulfill its full potential instead of proprietary wallets from different vendors, where each wallet uses that vendor's own APIs and verifiable credential designs, we will need general-purpose digital wallets that function much more like the real-world wallets we carry in our pockets or purses. In other words:

- **The wallet should accept any standardized verifiable credential** (or other standardized information artefact) just like you can fit any paper or plastic credential of the right size into your physical wallet.
- **You can install the wallet on any device you regularly use**, just like you can put a physical wallet in the pocket of any clothes you wear or purse you carry. Unlike physical wallets, however, your digital wallets should automatically stay "in sync" across your different devices the same way your email and messaging apps do today.
- **You can back up and move your keys and verifiable credentials between digital wallets as needed**—even from different vendors—just like you could move your physical credentials from one wallet to another.
- **You should have the same basic experience no matter what wallet⁴ you use**—even across different wallets from different vendors—because this is critical to being able to use your wallets safely and securely.

Adam Gunther, former Executive Director for Blockchain Trusted Identity at IBM and now VP Identity at Equifax, expresses this last point as, "**one wallet, one experience**"⁵ — one standard way of managing your verifiable credentials and trust relationships no matter what wallet or device you are using. Not only will it be the easiest experience for end-users, but it will also be the safest approach, because it makes it much harder to try to fool or "phish" an identity owner into doing the wrong thing.

Kim Cameron, Microsoft's Chief Identity Architect, felt this was so important that he made it the last of his seven Laws of Identity.⁶ He called it "Consistent Experience Across Contexts." He compares it to what we have to learn to know how to drive. The worldwide automotive industry has standardized the controls for driving (e.g., steering wheel, accelerator, brakes, turn signals) across all makes and models of cars to minimize the learning experience and maximize safety for drivers. The reasons are obvious: a car that does not operate the way a driver normally expects could end out killing the driver or others. We need the same level of

⁴ For more about interoperability and standardization across SSI digital wallets, please see the chapter on digital wallets in Part 2. That chapter also answers the other most frequently-asked question about digital wallets: what happens if I lose my phone? (Without spoiling the punch line, the answer is: by automatically maintaining an encrypted backup, your digital wallet will actually be much safer and more secure from loss, theft, or hacking than any physical wallet you can carry.)

⁵ <https://www.ibm.com/blogs/blockchain/2019/04/episode-1-the-future-of-protecting-your-wallet-and-identity/>

⁶ <http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

attention to the safety of our digital lives.

Digital agents and hubs

The second major difference between a digital wallet and a physical wallet is how it operates. A physical wallet is “operated” directly by a person—the owner. He or she sets it up, adds credentials as they are created or acquired, selects credentials to be presented to a verifier when a proof is required, and moves the wallet from pocket-to-pocket or purse-to-purse when the owner needs the wallet in different places.

Because people don't "speak" in digital bits and bytes, digital wallets require software to operate them. In SSI infrastructure, this software module is called a **digital agent**, or just **agent**. As figure 2.8 illustrates, you can think of an agent as a digital guardian that "wraps" around your digital wallet to protect it and makes sure that only you, the person responsible for your verifiable credentials and cryptographic keys, can take them in and out to use them.

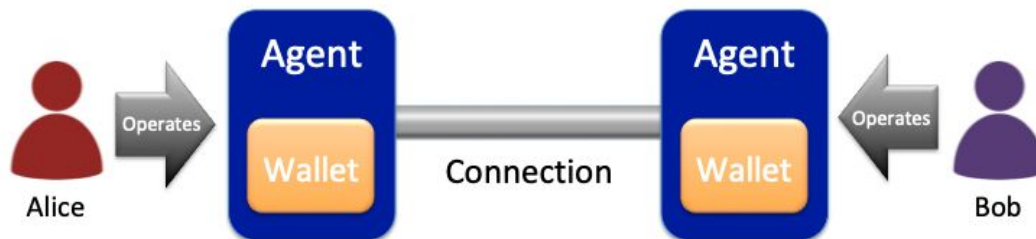


Figure 2.8: In SSI infrastructure, every digital wallet is “wrapped” by a digital agent that acts as a software guardian, making sure that only the wallet’s controller (typically the identity owner) can access the stored verifiable credentials and cryptographic keys.

In SSI infrastructure, an agent has a second job in addition to helping identity owners manage their wallets. Using instructions from their owners, agents “speak” to each other over the Internet to **form connections** and **exchange credentials**. They do this via a **decentralized secure messaging protocol** that is designed from the ground up for private communication between digital agents.

Figure 3.8 is a high-level overview of how agents and their wallets form connections and communicate in the SSI ecosystem.⁷ There are two general categories of agents based on where they are located: **edge agents** operate at the edge of the network, on identity owner’s local devices. **Cloud agents** operate in the cloud, where they are hosted either by standard cloud computing platform providers or specialized cloud service providers called **agencies**.

⁷ From Appendix F of the Sovrin Glossary V2.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

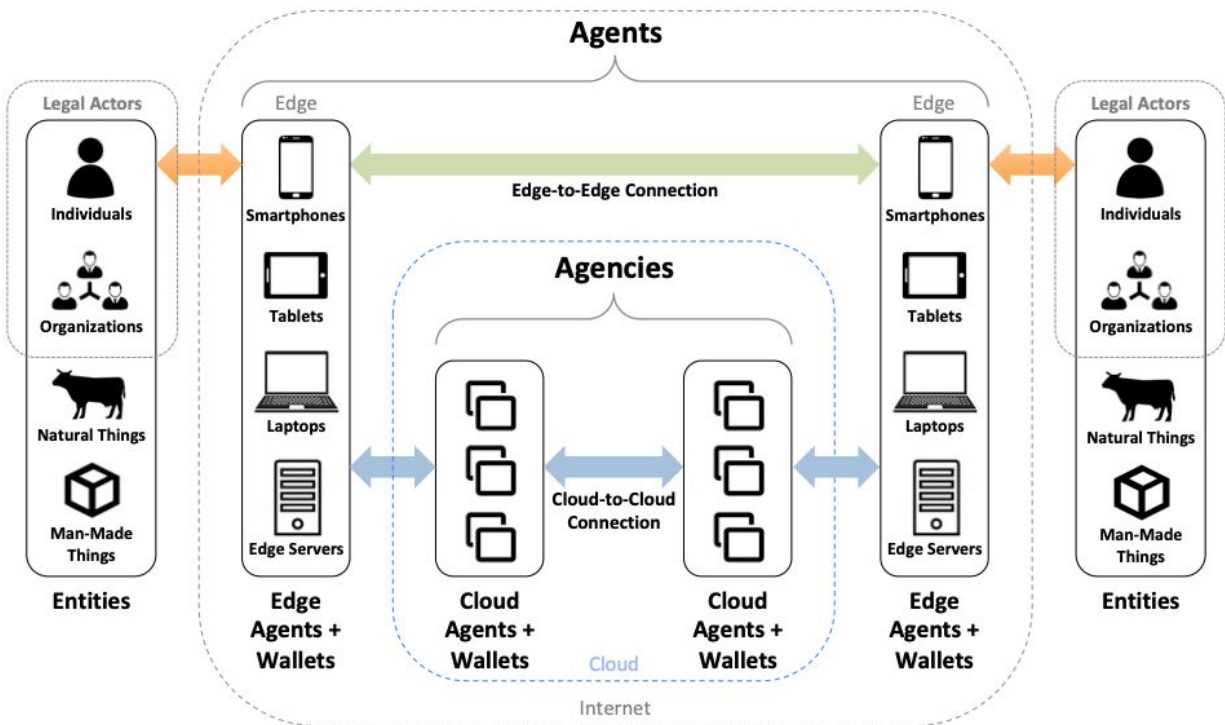


Figure 2.9: An overview diagram of the role of agents and wallets in the SSI ecosystem. Identity owners interact both directly with edge agents operating on local devices and indirectly with cloud agents operating remotely in the cloud.

Cloud agents can also be designed to store and synchronize other data on behalf of an identity owner, e.g., files, photos, financial records, medical records, asset records, etc. These encrypted data stores are called **identity hubs, digital hubs**, or just **hubs**.⁸ They can serve as the backbone for digital life management applications and services that process and maintain digital identity data of all kinds for the lifetime of an individual—and beyond, i.e., a digital hub can be of great assistance in the management and settlement of a digital estate after its owner has passed away.⁹

So the next question is: how do agents find each other to connect and communicate on behalf of their identity owners?

⁸ The Decentralized Identity Foundation and Hyperledger published a joint paper on how agents and hubs work together:
<https://medium.com/decentralized-identity/rhythm-and-melody-how-hubs-and-agents-rock-together-ac2dd6bf8cf4>
⁹ Agents, hubs, decentralized secure messaging, and other aspects of decentralized identity architecture are covered in more detail in the technology architecture chapter.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Decentralized Identifiers (DIDs)

What made the Internet possible was a new addressing system that allows any device on the Internet to form a connection with and send data packets to any other device on the Internet. Figure 3.10 is a diagram of an IP (Internet Protocol) version 4 address.

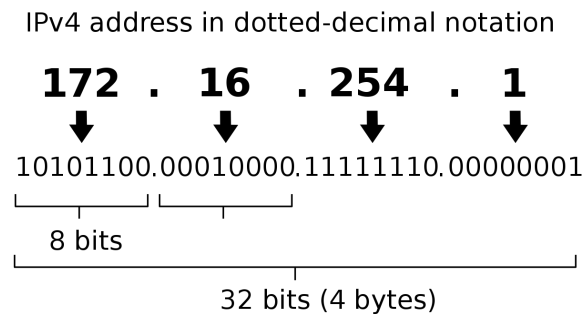


Figure 2.10: Example of an IP address—the addressing format that let any device on the Internet talk to any other device on the Internet.¹⁰

But as we explained in the opening pages of this book, knowing the IP address of a machine on the Internet doesn't tell you anything about **the identity of the person, organization, or thing controlling that machine**. To do that, the controller (the identity owner) needs to be able to provide *proof* about their identity, attributes, relationships, or entitlements. And that proof has to be verifiable in some way.

For many years we've had a technology for creating digital proofs: **public/private key cryptography**.¹¹ The owner of a private key uses it to sign messages, and anyone else can verify this signature using the owner's corresponding public key. The signature verification shows that the signature was created by the owner of the private key and that the message has not been tampered with since.

To rely on this verification, however, the verifier must know the correct public key for the owner. So, for decentralized messaging between digital agents and wallets to be secure—and for agents to be able to send cryptographically-verifiable proofs of verifiable credentials to each other—we need a very strong, secure, scalable way for identity owners and their agents to **prove ownership of their public keys**.

Just like with the Internet, the answer was a new identification system. Only this time the identification system needed to be designed from the ground up for digital agents and their public keys. And because it needs to be extremely secure, this new type of identifier needed

¹⁰ From https://en.wikipedia.org/w/index.php?title=IP_address&oldid=888840206

¹¹ We explain the basics of public/private key cryptography in our cryptography chapter.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

the following four core properties:

1. **Permanent.** The identifier needs to be able to never change, no matter how often the identity owner moves, uses different service providers, or uses different devices.
2. **Resolvable.** The identifier needs to be able to retrieve not just the current public key or keys for the identity owner, but also the current addresses to reach the owner's agent or agents.
3. **Cryptographically verifiable.** The identity owner needs to be able to prove, using cryptography, that he, she, or it has control of the private key for each public key associated with that identifier.
4. **Decentralized.** Unlike X.509 certificate trees that rely on centralized registries under the control of single authorities, this new type of identifier must be able to avoid single points of failure by using decentralized networks such as blockchains, distributed ledgers, distributed hash tables, distributed file systems, peer-to-peer networks, etc.

It is the last of these four properties that gave this new address its name: DID (decentralized identifier). Given its different purpose, a DID has a much different structure than an IP address. An example DID is shown in figure 3.11.

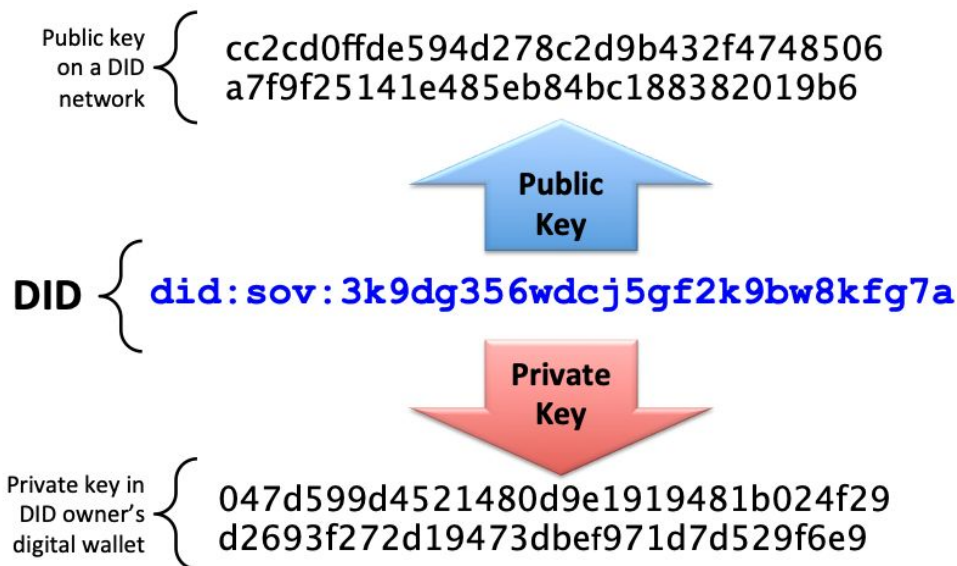


Figure 2.11: An example of a DID (Decentralized Identifier) on the Sovrin DID network. A DID functions as the identifier of a public key on a blockchain or other decentralized network. In most cases it is also the identifier for locating an agent for the entity identified by the DID.

Just as the original research for the Internet was sponsored by a U.S. government agency (DARPA), the original research and development for the first DID specification was done under a contract with the U.S. Department of Homeland Security (DHS). Published in late

2016, it was subsequently contributed to the W3C Credentials Community Group (CCG) to start the process of becoming an official standard.¹² That process resulted in the launch of the W3C DID Working Group in September 2019.¹³

DIDs are designed so they can take advantage of any modern blockchain, DLT, or other decentralized network via a **DID method** that is written specifically for that target system. The DID method defines the following four atomic operations on any DID:

1. How to **create** (write) the DID and its accompanying **DID document** (the file containing the public key(s) and other metadata describing the DID subject) to the target system.
2. How to use the DID to **read** (look up) the DID document from the target system.
3. How to **update** the DID document for a DID, e.g., to rotate a public key.
4. How to **deactivate** a DID by terminating its usage (usually by updating its DID document to contain no information).

The DID in the previous figure uses the Sovrin DID method (`did:sov:`) written for the Sovrin SSI network.¹⁴ The informal DID Method Registry¹⁵ maintained by the W3C CCG includes the DID methods for more than 30 different target decentralized networks, including (at the time of publication) three methods for Bitcoin and five for Ethereum. It also includes two methods (`did:peer:` and `did:git:`) that do not even need a distributed ledger because they work entirely peer-to-peer.¹⁶

Just as any two devices with their own IP addresses can use the TCP/IP protocol stack to form a connection and exchange data, any two identity owners with DIDs can use the SSI protocol stack to form a cryptographically secure connection to exchange data. The basic concept of DID-to-DID **connections** (shown in figures 3.8 and 3.9) is not new—it is directly analogous to how connections work in many other networks. But DID-to-DID connections bring five powerful new properties to digital relationships:

1. **Permanent**—the connection will never break unless one or both parties want it to.
2. **Private**—all communications over the connection can be automatically encrypted and digitally signed.
3. **End-to-end**—the secure connection has no intermediaries.
4. **Trusted**—the connection supports verifiable credential exchange to establish trust to any level of assurance.
5. **Extensible**—the connection can be used for any other application that needs secure, private, reliable digital communications.

¹² See the DID chapter for more about DIDs, DID documents, DID methods, and the DID specification.

¹³ <https://www.w3.org/2019/did-wg/>

¹⁴ <https://sovrin-foundation.github.io/sovrin/spec/did-method-spec-template.html>

¹⁵ <https://w3c-ccg.github.io/did-method-registry/>

¹⁶ <https://dhh1128.github.io/peer-did-method-spec/index.html>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Blockchains

A DID can be registered with any type of decentralized network—or even exchanged peer-to-peer. So why would someone choose to register a DID on a blockchain? What do blockchains provide that other types of electronic identifiers and addresses we have been using for decades—telephone numbers, domain names, email addresses—do not?

The answer is deeply rooted in cryptography, databases, and networking. Blockchains are **highly tamper-resistant transactional distributed databases that no single party controls**. This means they can provide an authoritative source of data that many different peers can trust without any single peer being in control. Blockchains intentionally trade off many other standard features of distributed transactional databases—performance, efficiency, scalability, searchability, ease of administration—to solve *one really hard problem*—trusted data that does not need to rely on a central trusted authority.

Figure 3.12 illustrates the fundamental design differences in design between traditional databases—whether centralized or distributed—and blockchains.

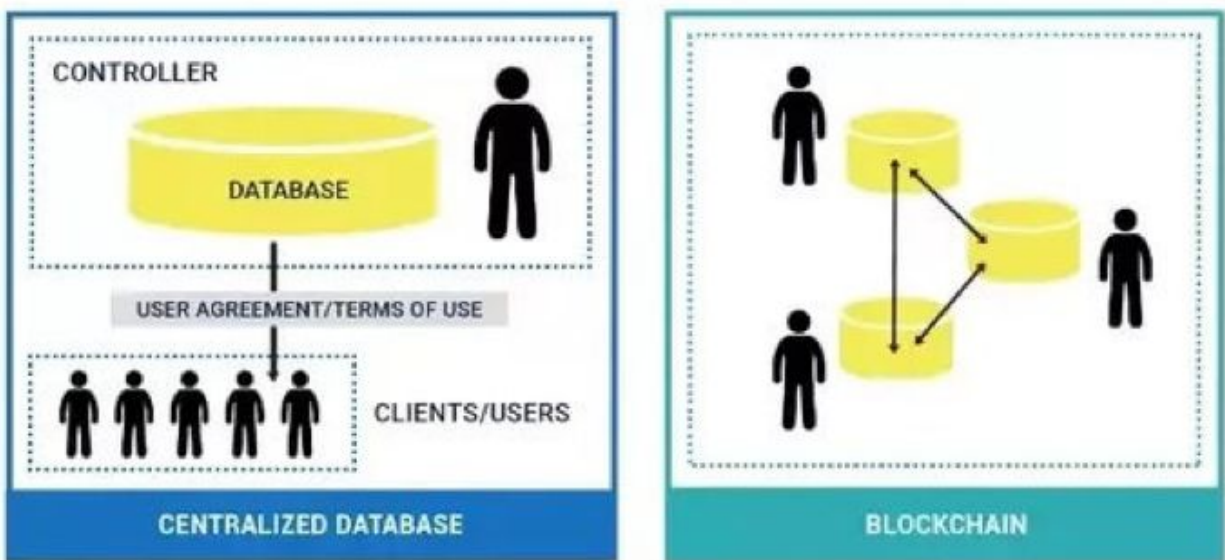


Figure 2.12: The fundamental innovation of a blockchain as a transactional database is that it has no centralized administrator or controller—it uses cryptography to allow many different peers to each control their own transactions, while making it nearly impossible for anyone to alter any transaction once it has been written to the blockchain.

It is important to understand how blockchains accomplish the feat of removing the need for a central authority while still maintaining very high trust in the integrity of the data. It's not

just cryptography, it is a **triple play of cryptography**:

1. **Every transaction (the writing of a new record) to a blockchain is digitally signed.** This is how control of the distributed database is spread out across all the peers—every peer manages its own private keys and digitally signs its own transactions with the blockchain, and no new transaction is accepted unless the signature is verified.
2. **Transactions are grouped into “blocks” that are cryptographically hashed and linked to the previous block.** This is the step that creates an immutable “chain” of ordered transactions.
3. **Every new “block” is cryptographically replicated across all peer nodes on the blockchain network, each of which is run by a different party.** This is the step performed by the **consensus protocol** at the heart of many blockchains. There are many different consensus protocols—some of which (like Bitcoin) involve cryptographic proof-of-work called “mining”. But regardless of the specific protocol, in the end every peer node in the network ends up with a copy of the latest “block”, and they all agree on that copy. The fact that each node is run by a different party combined in some cases with clever game-theory makes it far less likely that 51% or more of the nodes will collude in order to try to attack the blockchain (and have some chance of rewriting its history).

This cryptographic triple play is what makes a blockchain so hard to attack—modifying even a single bit of a single transaction that has already been recorded to the blockchain and replicated to all the peer nodes requires breaking into tens, hundreds, or thousands of machines and changing them all at once.

From the standpoint of SSI—and specifically for registering and resolving the DIDs and public keys that enable digital wallets and digital agents to securely communicate and exchange verifiable credentials—the differences between the various types of blockchains (permissionless, permissioned, hybrid, etc.)¹⁷ do not matter much. A DID method can be written to support pretty much any modern blockchain or other decentralized network.

What matters is that blockchains solve a problem that has never had a solution in the history of cryptography: they are globally distributed databases that can serve as a source of truth for public keys without being subject to single points of failure or attack. This is what provides the strong foundation needed for the ubiquitous adoption of the verifiable digital credentials at the heart of SSI.

¹⁷ We will explain more about the differences between Blockchain and DLTs in the decentralization believer chapter about Blockchain and DLT.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Governance frameworks

The ultimate goal of all SSI infrastructure is to achieve a mutually acceptable level of trust between any two parties interacting on the Internet—a goal that for many types of transactions is simply not possible today. With SSI, the foundation for this trust layer is first laid by **cryptographic trust**, i.e., by rooting publicly-resolvable DIDs and public keys for credential issuers in a decentralized network. This anchors the trust triangle shown in figure 3.4 and again in figure 3.13, so verifiers can rely on the digital signature of an issuer.

But cryptographic trust is not always sufficient to achieve **human trust**. For example, while Bitcoin uses cryptographic trust and game theory to power its system of value exchange, by itself it does not provide any solution to the challenges of Know Your Customer (KYC), Anti-Money Laundering (AML), or Anti-Terrorism Financing (ATF) regulations for money transmission. But cryptographic trust in the Bitcoin blockchain network *could* be used to anchor a DID for an issuer of verifiable credentials to parties making Bitcoin transactions. Provided that the issuer has enough human trust in their authority (for example, a credit union, bank, insurance company, or government agency), those verifiable credentials may be used by verifiers to satisfy KYC or AML regulations.

This layering of human trust on top of cryptographic trust is how SSI delivers the full power of verifiable credentials. But trusting in credentials from one issuer at a time doesn't scale. This was the same problem faced in the early days of credit cards in the 1960s. Each major bank tried to issue its own brand of credit card, and merchants were overwhelmed—they couldn't handle dozens of different credit cards from different banks.

So credit card adoption didn't take off until banks got together and formed *credit card networks*—Visa and MasterCard being the two best known. Each of these is governed by a set of business, legal, and technical rules known as a **governance framework** (also known especially in the digital identity industry as a **trust framework**). The entity that creates and administers a governance framework is known as the **governance authority**.

A governance framework creates the second trust triangle shown in the lower half of figure 3.12. This illustrates how a governance framework can make a verifier's job easier: when presented with proof of a credential from an issuer the verifier does not know, the verifier can request a second proof from the issuer (now acting as a holder/prover) proving that issuer is authorized under a governance framework the verifier trusts. This proof comes from another verifiable credential issued by the governance authority to the issuer. This approach of "recursive trust triangles" can work for any size trust community—even Internet-scale trust communities where verifiers do not directly know all the issuers (e.g., MasterCard and Visa).

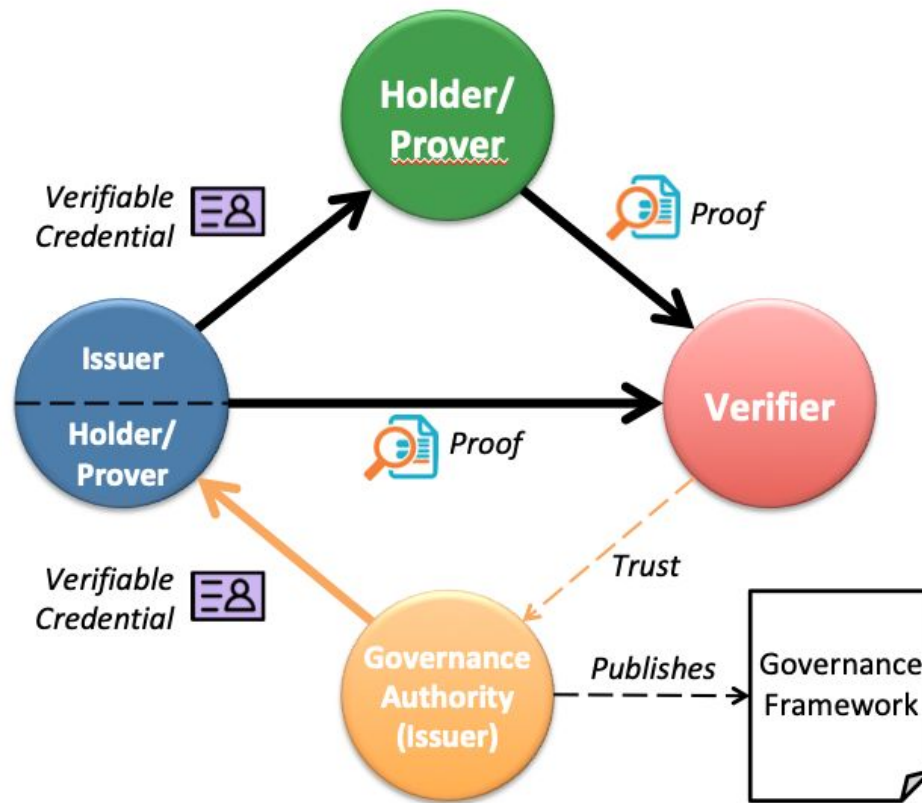


Figure 2.13: Governance authorities and governance frameworks represent a second trust triangle that enables verifiers to determine the authorized issuers for a specific set of verifiable credentials.

Governance frameworks are the “flip side” of a verifiable credential—they specify the policies and procedures that issuers must follow to issue that specific credential. In some cases they will also specify the terms and conditions to which holders must agree to obtain them. And when verifiers are paying issuers for the value of the credential, a governance framework can also specify liability policies, insurance requirements, and other legal and business variables that verifiers can factor into their trust decisions.

Governance frameworks are the secret to how verifiable credentials can scale to work in any size trust community, from a single city to an entire industry, or from one nation state to the Internet as a whole. As we will see in the balance of this book, these verifiable trust communities can be as transformative to our digital lives as credit card networks have been to the world of commerce.

Although an entire book can (and will) be written on each of these seven fundamental building blocks of SSI, the goal of this chapter was to introduce them so that anyone—technical or non-technical—can understand the basic role they play in SSI infrastructure. In fact we can summarize each building block in a single sentence.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

1. **Verifiable credentials** are the digital equivalent of the physical credentials we carry in our wallets to prove some aspect of our identities almost every day.
2. **Issuers, holders, and verifiers** are the three roles of the “trust triangle” that make credentials of any kind work: issuing the credential, holding it in a wallet, and verifying it when it is presented by the holder.
3. **Digital wallets** are the digital equivalent of our physical wallets for holding verifiable credentials on any modern computing device—smartphone, table, laptop, and so on.
4. **Digital agents** are the apps or software modules that enable us to use our digital wallets to obtain and present credentials, manage connections, and securely communicate and exchange verifiable credentials with other digital agents.
5. **Decentralized identifiers (DIDs)** are a new type of digital address that is both generated and verified using modern cryptography so DIDs do not require any kind of centralized registration authority.
6. **Blockchains** are globally distributed, cryptography-protected databases that can serve as a source of truth for DIDs and public keys without being subject to single points of failure or attack.
7. **Governance frameworks** are the set of business, legal, and technical rules for how SSI infrastructure will be used that are published by a particular governance authority that is trusted by the members of a particular trust community.

In Part 2 we will go into much greater technical detail about most of these building blocks. But first, in Chapter 3, we will put these building blocks to use by showing how their roles in a series of typical SSI usage scenarios.

Example scenarios of how SSI works

by Drummond Reed and Daniel Hardman

In chapter 2 we examined the core building blocks of SSI. In this chapter we'll show you how these building blocks are put together to implement SSI in seven example scenarios that progress from the very simple to the reasonably complex:

1. Alice and Bob form a connection after meeting in person at a conference.
2. Alice and Bob form a connection via Alice's blog.
3. Bob logs into Alice's blog to leave a comment.
4. Alice and Bob form a connection by meeting through an online dating site.
5. Alice applies for a bank account.
6. Alice buys a car.
7. Alice sells the car to Bob.

Each scenario illustrates basic patterns of SSI usage that you will see repeated in the many industry--specific SSI scenarios we will be exploring in Part Two.

A simple notation for SSI scenario diagrams

First, we want to introduce a simple notation for the diagrams in this chapter—a notation that illustrates how conceptually easy it is to apply SSI technology to standard business problems (even though there is real rocket science in the underlying cryptography and protocols). Our notation is inspired by the simple connector-style toys that can be found in almost any child's playroom.



Figure 3.1: SSI diagramming notation is inspired by simple connector-style toys.

As you will see, agents are the round connector pieces in this analogy and connections are the colored spars that connect together the various actors (people, organizations, things) who need to exchange credentials and proofs as required to establish trust. Figure 3.2 shows the ten icons used in our notation.

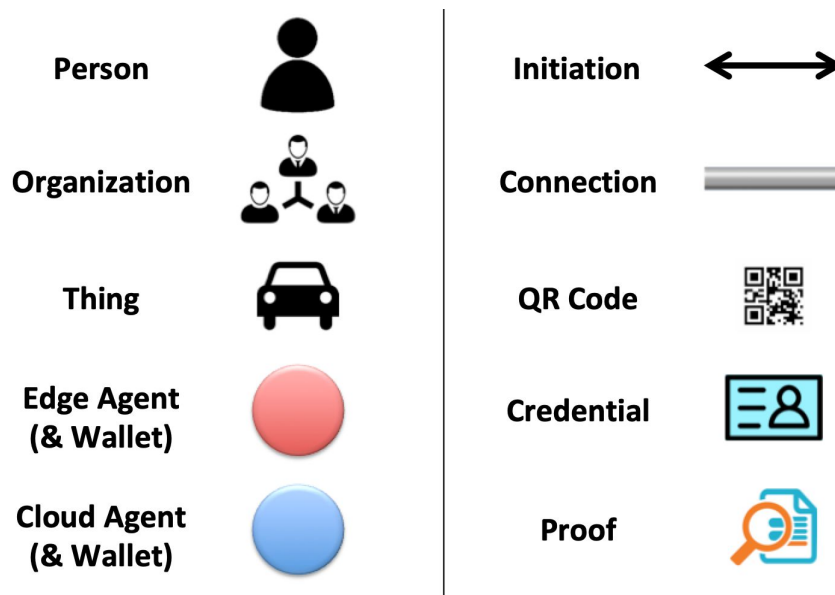


Figure 3.2: The simple set of icons we will use for SSI scenario diagrams in this chapter.

*Note: In these diagrams we will only refer to **agents**—every agent is assumed to be paired with a digital wallet and, where needed, with a digital hub.*

Our example scenarios will use the **Alice** and **Bob** characters that have become so iconic in cryptography and cybersecurity that there is an entire Wikipedia article about them.¹

¹ https://en.wikipedia.org/wiki/Alice_and_Bob

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Scenario #1: Bob meets Alice at a conference

Our first scenario is one of the most common interactions in business. Alice and Bob meet at a conference, discover some common interests, and want to stay in touch. In the pre-digital days, their exchange might look like this:



Figure 3.3: The pre-digital way of forming a business connection.

Nowadays, with the prevalence of mobile phones and business networks like LinkedIn, the ceremony looks more like the following figure.:



Figure 3.4: Entering business card info into mobile phone address books and/or LinkedIn.

Although entering contact info straight into a smartphone address book is good, establishing a connection on a service like LinkedIn is even better because that connection will persist even if you change jobs and phone numbers.

But what if Alice and Bob want to have a connection that persists *independent of any service provider or social network*? Or of any intermediary at all? A connection that will last

literally forever—or until either Alice or Bob breaks it, whichever comes first?

As described in chapter 2, an SSI can do that. The actual ceremony for forming the connection between Alice and Bob depends on the capabilities of their agents and smartphones. But one way is familiar to anyone who has used a mobile airline boarding pass: scanning a QR code.

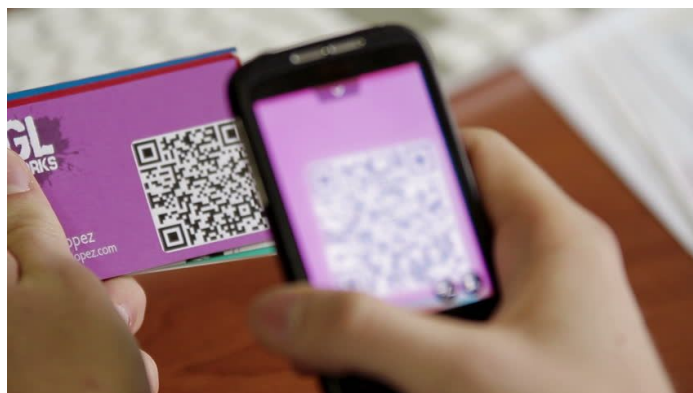


Figure 3.5: Forming a new SSI connection can be as easy as scanning a QR code.

Some people are already printing QR codes on their business cards for a similar purpose. But those QR codes usually take you to a service provider who can let you form a connection with the owner of the business card *on the service provider's own proprietary network*.

With SSI, you create a connection with **no intermediate service provider**. It's just Alice and Bob and nobody else. It doesn't matter whether Alice scans Bob's QR code or Bob scans Alice's. Either way, once both Alice and Bob approve it, a direct, private, peer-to-peer SSI connection is created between their two agents, as shown in figure 3.6

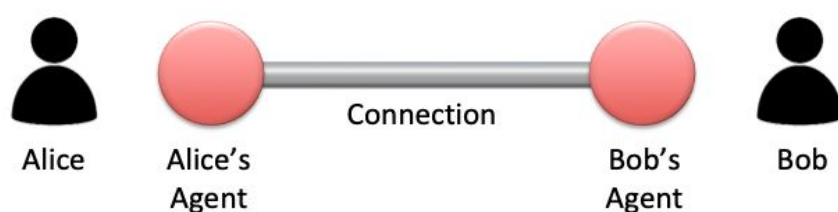


Figure 3.6: A simple connection between Alice and Bob implemented by their respective agents.

Of course figure 3.6 is an over-simplification. It doesn't show the specific hardware and networks being used or the DIDs and DID documents being exchanged. But conceptually it is an accurate picture of what exists after Alice and Bob have created their connection.

What's going on under the hood

For more technically-minded readers, figure 3.7 takes this scenario down another level of detail.

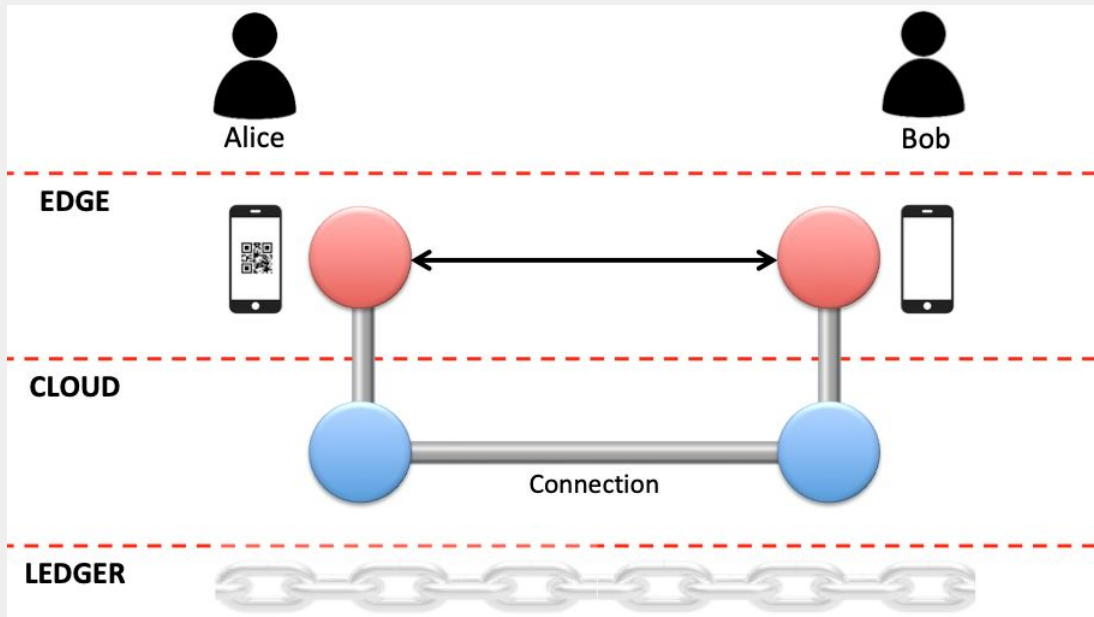


Figure 3.7: A more detailed view of how the connection is formed between Alice and Bob, showing what is happening with edge agents on their phones (red) and with cloud agents hosted at the agencies they each use (blue).

Figure 3.7 assumes the following:

1. Alice and Bob already have SSI edge agents (and wallets) set up on their smartphones.
2. Alice and Bob already have connections between their respective edge agents and cloud agents hosted by the agencies of their choice (it makes no difference if Alice and Bob use the same agency or different ones, just as it makes no difference if two email users use the same email service provider).
3. Alice is the one generating a QR code to be scanned by Bob (the process works identically in reverse).

Based on these assumptions, here are the details for how the connection is formed:

1. Alice clicks the button in her edge agent app to cause it to generate a QR code for a new connection—called a **connection invitation**. This request includes information about how Bob can reliably contact her over an encrypted channel. The data in this QR code is not secret and does not need strong security guarantees; the encrypted channel adds those guarantees later in the flow.

2. Before it generates the QR code, Alice's edge agent generates a nonce and sends it to Alice's cloud agent to notify it to expect a message associated with that nonce. This nonce will be included in the QR code, making the QR code unique to Bob—when Alice forms her next connection with Carol, the QR code will be different.
3. Bob uses his edge agent app to scan the QR code.
4. Bob's edge agent generates a unique new public/private key pair and a **peer DID**—a private pairwise pseudonymous identifier that will be known only to Alice—and saves it in his wallet.
5. Bob's edge agent composes a **connection request** message that includes the new peer DID, public keys, and service endpoints in a DID document (see Chapter 2).
6. Bob's edge agent sends his connection request message to Bob's cloud agent with instructions to forward both to the encrypted channel that Alice identified in her invitation.
7. Bob's cloud agent forwards the message.
8. Alice's cloud agent receives Bob's messages and pushes them to Alice's edge agent asking if Alice wants to confirm the connection.
9. Alice's edge agent prompts Alice to confirm the connection.
10. Alice clicks Yes.
11. Alice's edge agent saves Bob's half of the connection information in its wallet.
12. Alice's edge agent does the same thing Bob's did: generates a unique new public/private key pair and a peer DID that will be known only to Bob and saves it in her wallet.
13. Alice's edge agent now creates a **connection response** that is the mirror image of Bob's connection request. It contains Alice's own peer DID, public key, and service endpoints for the connection. It is encrypted for the public key(s) from Bob's DID doc, and is sent to the endpoint that Bob also specified there. Alice can use this message to update the service endpoint or keys that Bob has on file for her, so that the insecure information from the connection invitation is replaced with something no eavesdropper could know.
14. Alice's edge agent sends her connection response to Alice's cloud agent.
15. Alice's cloud agent sends the connection response to the service endpoint that Bob gave for Bob's cloud agent.
16. Bob's cloud agent forwards the connection response to Bob's edge agent.
17. Bob's edge agent saves the Alice half of the connection information in Bob's wallet.
18. Bob's edge agent notifies Bob that the connection is complete in both directions.

Now Alice and Bob have a permanent, private connection they can use for cryptographically secure communications of any kind. Four notes about this:

1. **No verifiable credentials needed to be exchanged** for Alice and Bob to begin trusting each other because this connection was formed in person. This doesn't mean some type of credential exchange won't be needed in the future—for example if they are going to do a high-stakes business deal—but right now they have sufficient trust.

2. **No interactions were needed with a public ledger or blockchain.** The whole process took place using peer DIDs and DID documents that were generated and exchanged completely off-chain—good for both privacy and scalability. Public DIDs registered on a public blockchain are generally only needed for issuers of credentials that anyone needs to be able to verify.
3. **The connection is completely private and known only to Alice and Bob.** No intermediary service providers were involved except to host cloud agents and deliver encrypted messages. Those cloud agent hosting provider(s) only know that there was traffic between two agents. They can't "see" inside the messages to know who was talking to whom about what.²
4. **If either Alice or Bob—or both—move agencies, the connection moves with them.** Their respective agents just send the other an update to their connection information with the new private cloud agent address at the new agency.

Scenario #2: Bob meets Alice through her online blog

This scenario also results in Alice and Bob forming a connection. But this time they don't meet in person. Rather Bob discovers Alice's blog about her state-of-the-art website design business. Bob likes the content there and decides he'd like to contact Alice about creating a website for him.

Alice, being a cutting-edge web designer, has SSI-enabled her blog so that it can accept connection requests. This means her blog has its own cloud agent. Because this is Alice's personal blog, this cloud agent acts as another representative of Alice as an individual—it doesn't need to represent the web hosting site or anything else with a separate identity. This illustrates a key principle of SSI: **Alice can have as many agents as she wants**, each of which allows Alice to express her identity and form new relationships in a specific context—in this case her online presence as a web designer.

² This can be further privacy-protected by using [onion routing](#) between agents.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Figure 3.8 shows the agents and connections established in this scenario. The numbers indicate the order in which actions are taken.

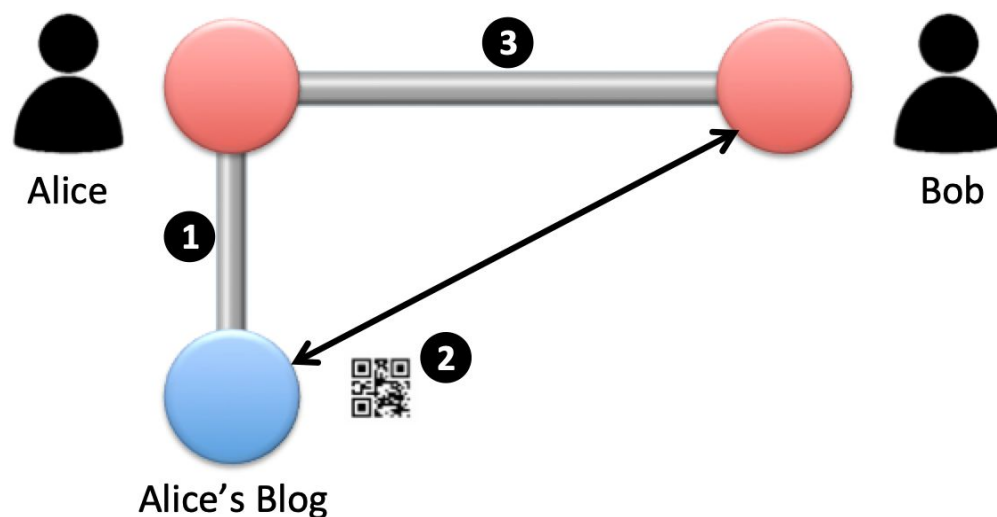


Figure 3.8: The scenario of Bob meeting Alice by first reading Alice's blog and then deciding to request a connection with her by scanning a QR code on her blog.

1. **Alice sets up and connects to a cloud agent for her blog.** Depending on her blogging service, this could take under a minute. (See the following sidebar if you are interested in the technical details.) Once Alice does this, her blog can now display a unique QR code to any reader who wants to request an SSI connection with Alice.
2. **Bob uses his edge agent to scan the QR code to request a connection.** Bob's edge agent app prompts him to accept the new connection with Alice and Bob presses Yes.
3. **The connection is formed between Bob and Alice.** At this point, the rest of this scenario plays out just like it did in example scenario #1. Bob's agent sends the connection request to Alice's agent; when Alice approves it, her agent reciprocates with a connection response and the connection is completed.

Note that, like scenario #1, this scenario also does not require any exchange of verifiable credentials. If Alice is willing to accept connection requests from any reader of her blog, her agent does not need to ask for any further identity information. But this also exposes Alice to receive spam connection requests. To guard against these, Alice's edge agent can require proofs of one or more common verifiable credentials that Alice trusts. This is described further in scenario #4.

What's going on under the hood

Alice hosts her blog on WordPress, so we'll assume WordPress has an SSI plug-in.³ When Alice installs this plug-in, it shows her a QR code needed to set up her cloud agent. Alice uses the edge agent app on her phone to scan the QR code. Her edge agent app then prompts her to approve setting up a cloud agent for her blog.

When Alice says yes, the edge agent sends a message to Alice's agency agent (not shown in Figure 3.8) requesting to: a) provision the new cloud agent, and b) form a connection with Alice's edge agent. When finished, the agency agent sends a connection response to Alice's edge agent with the peer DID and DID document containing the private network address and public key for encrypted communications with her new cloud agent.

Now Alice's blog has a new feature: the ability to display a unique QR code to any reader who wants to request a new connection with Alice.

Note that this process works the same way for anything that Alice might want to have an agent for—her LinkedIn page, her Facebook page, a website she has designed, even the signature on her emails. It can also work for anything offline that Alice wants to set up an agent for, such as her business card, or the placard for one of her paintings hanging in a local gallery.

Scenario #3: Bob logs into Alice's blog to leave a comment

Once Bob has a connection with Alice, Bob can use that connection to authenticate himself to Alice at any time. For example, if Bob later returns to Alice's blog and wants to leave a comment, Bob doesn't need to create an "account" with Alice's blog. **His connection is the account.** So not only does Bob NOT need to create a new username and password for Alice's blog—he'll never have to remember them either!

Called **passwordless login** (or **auto-authentication**), this is one of the headline features of SSI (see the complete roster of major SSI features and benefits in chapter 4). And it works the same way for any SSI-enabled website or application. Figure 3.9 shows the basic sequence.

³ Such a Wordpress plug-in is under development as this book is being written.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

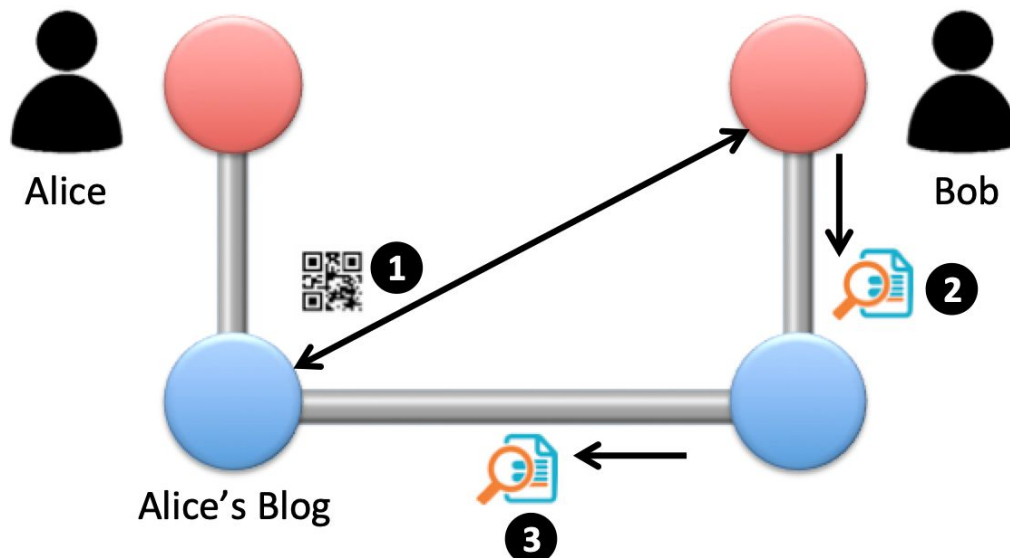


Figure 3.9: Bob uses a similar pattern to “log in” whenever he returns to Alice’s blog to leave a comment on one of her new posts

1. **Alice’s blog generates the QR code.** Just as in example #2, the first step is for Alice’s blog to generate a QR code for a reader to scan if they need to authenticate (e.g., to leave a comment, or do anything else that requires authorization). This time, when Bob scans the QR code, his edge agent recognizes that Bob already has a connection with Alice. So Bob’s edge agent asks Bob to confirm that he wants to authenticate (or, if Bob has told his edge agent to skip those kinds of confirmations, his edge agent will just proceed).
2. **Bob’s edge agent generates, signs and sends a proof.** This proof is signed with the private key Bob’s edge agent keeps *for this private connection only*. The proof is then sent to his cloud agent to forward to the private cloud agent connection address he already has for Alice.
3. **Alice’s cloud agent receives and verifies the proof.** Alice’s cloud agent verifies the signature using the public key it has for her private connection with Bob (shared by Bob when the connection was formed—and updated by Bob whenever his edge agent rotates his keys). If the signature verifies, Bob is “logged in”.

What is particularly powerful about auto-authentication is that it can easily be “tuned” to request the proofs needed to meet the level of authentication a verifier requires for a particular transaction. For example, leaving a comment on a blog is a relatively low-risk activity, so it is enough if Bob can prove he has the private key for the connection (which only he knows). But if Bob was asking his credit union to make a \$100,000 transaction, the QR code produced by the credit union could request a much stronger proof that this is really Bob.

SSI agents can also produce the authentication tokens needed for other Web authentication

protocols like OpenID Connect and WebAuthn. For more details about SSI-based authentication, see _____.

Scenario #4: Bob meets Alice through an online dating site

Once again this scenario will result in Alice and Bob forming a connection just like the one created in examples #1 and #2. But this time the introduction will come via an intermediary matchmaker: an online dating site. This means neither Alice or Bob have any pre-established trust relationship, so this is our first scenario that will require a verifiable credential.

Figure 3.10 is a diagram of the steps involved.

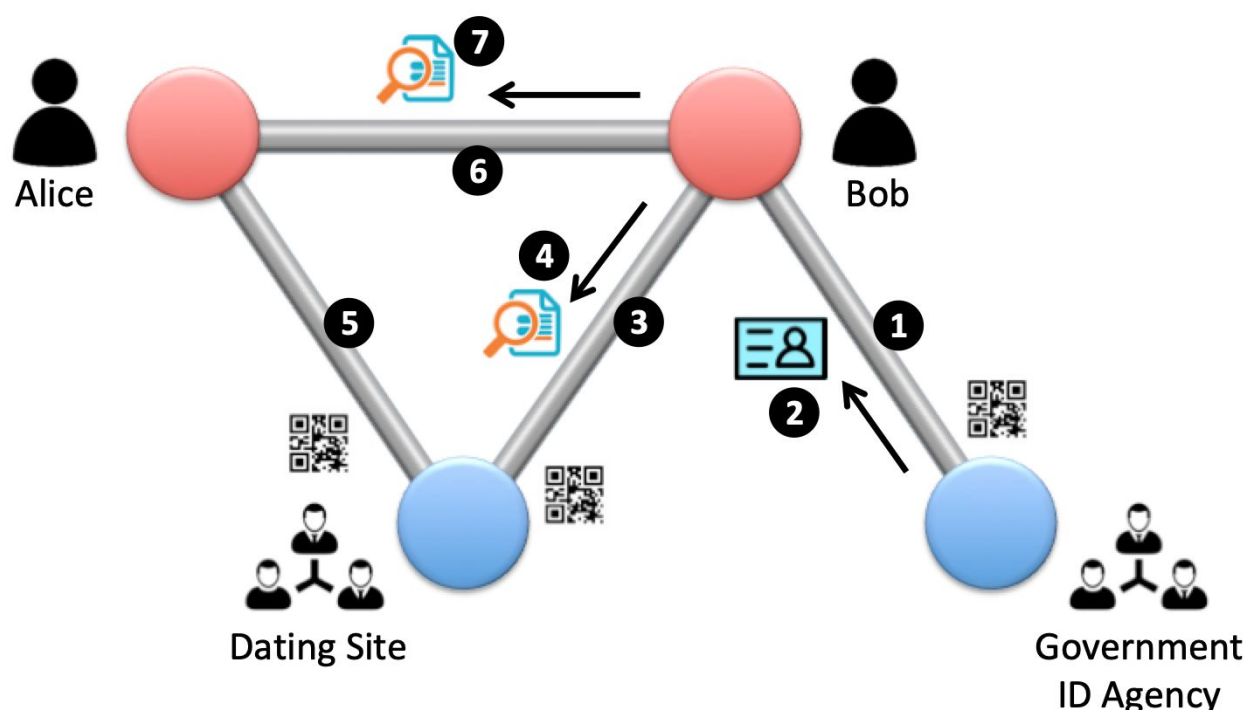


Figure 3.10: The scenario of Alice and Bob meeting through an online dating site—and Bob proving his authenticity to Alice over their newly formed connection.

1. **Bob connects to the Government ID Agency.** Bob knows the Dating Site requires proof of identity from a government ID, so first he forms a connection with the Government ID Agency cloud agent by scanning a QR code on its website as described in scenario #2.
2. **Bob requests a Government ID credential.** Bob's edge agent prompts him for the data (or other steps he must take) to prove his identity to the Government ID Agency (which could involve the Agency requesting proofs of other verifiable

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

credentials from Bob). Once Bob meets the Agency's policies for issuing a new Government ID credential, the Agency sends the credential to Bob's edge agent to store in Bob's wallet.

3. **Bob connects to the Dating Site** as per scenario #2.
4. **The Dating Site requests proof of Bob's Government ID credential.** Bob scans a QR code requesting a proof of the Government ID credential. Bob's edge agent prompts Bob to approve sending the proof. Bob agrees, and his edge agent generates the proof, signs it with the private key for this connection, and sends it to the Dating Site's cloud agent. The Dating Site's cloud agent verifies the proof by looking up the DID of the Government ID Agency on a public blockchain such as Bitcoin, Ethereum, or Sovrin in order to retrieve the DID document with the public key. If the proof verifies, Bob is good to go to use the Dating Site.
5. **Alice joins the Dating Site** by forming a connection the same way. The Dating Site may or may not require Alice to send proof of a credential (not shown in Figure 3.10). Alice then shares some profile data (which, if already stored in other credentials she has in her wallet, could be sent in a proof from her edge agent).
6. **Bob requests a connection with Alice.** Bob discovers Alice's profile on the Dating Site and requests a connection by scanning a QR code—just like he would have done via her blog in scenario #2. But this is a different context, so Alice is more cautious. Her edge agent requires Bob to share a proof of his Government ID credential. Note that Alice could rely on the fact that the Dating Site already has a policy requiring a male member to provide proof of a Government ID credential. But with SSI, *Alice can request that proof directly from Bob*. She doesn't have to worry if the Dating Site is not screening members properly.
7. **Bob sends Alice proof of his Government ID credential.** Bob can do this in one click because his edge agent does all the work. Alice's edge agent receives the proof over the private connection it set up with Bob and verifies the proof just like the Dating Site did. If it verifies, Alice accepts the new connection. If it fails, Alice's edge agent can delete the connection immediately **without ever needing to bother Alice**. Her edge agent is effectively serving as Alice's protector to make sure that any suitor meets Alice's minimum verification requirements.

What is even more powerful is that with SSI, **Alice isn't constrained to live only by the filters supported by the dating site**. Alice can request whatever proofs she wants from her suitors—and her suitors must supply those proofs before they ever even get to “knock on Alice's door,” so to speak. This could significantly change the online dating ecosystem in a way that increases trust for everyone involved.

Scenario #5: Alice applies for a new bank account

The next scenario is such a clear business case for the value of verifiable credentials that in October 2018, four companies—IBM, Workday, Alberta Trust Bank (ATB), and Evernym—collaborated to demonstrate how it works in a video called *Job-Creds*.⁴ The video shows how Alice first obtains a government ID credential from a driving licensing agency, then an employment credential from her employer, IBM, in order to finally apply for a bank account at ATB. Figure 3.11 shows the complete set of connections and interactions (note that for simplicity we left out the QR codes used to request connections or proofs).

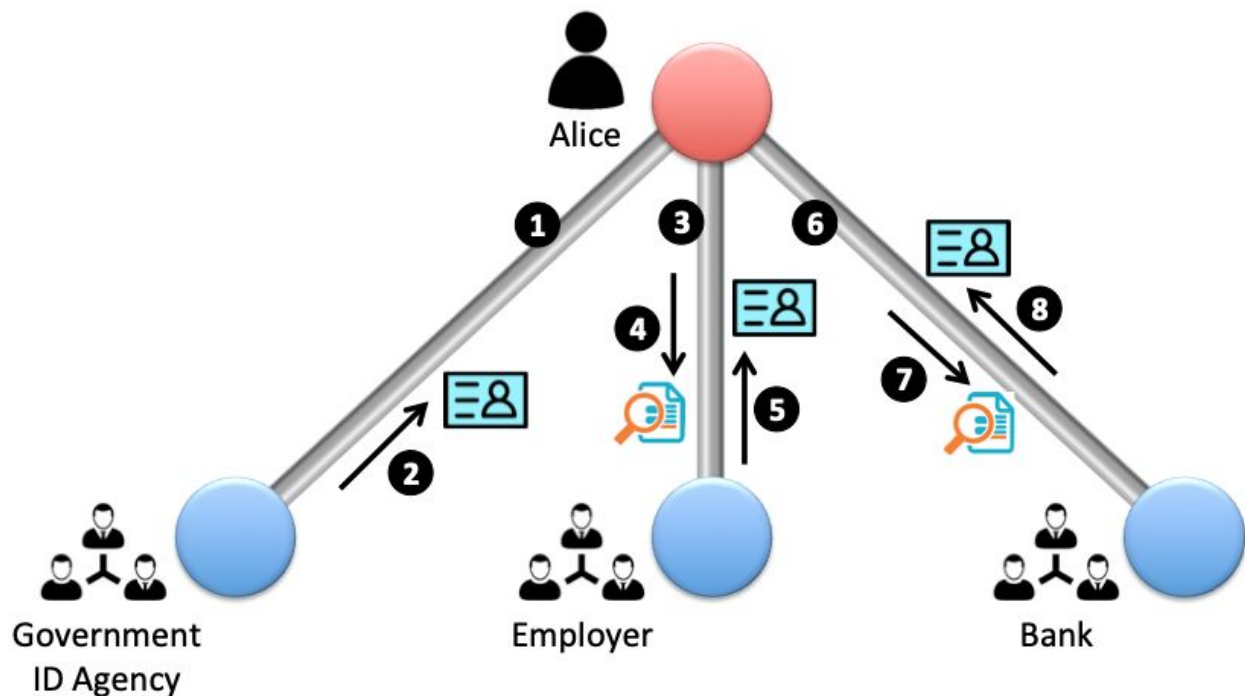


Figure 3.11: The connections and credentials Alice needs from a Government ID Agency and her Employer in order to apply for a bank account.

1. **Alice forms a connection with a Government ID Agency.** This is the same step 1 that Bob took in scenario #4.
2. **Alice receives her Government ID credential.** Once Alice satisfies the policies of the Government ID Agency, it issues the credential to her edge agent.
3. **Alice forms a connection with her Employer.** This would typically be set up as part of onboarding Alice as a new employee.
4. **Alice sends a proof of her Government ID credential to her Employer.** Again,

4

<https://www.ibm.com/blogs/blockchain/2018/10/decentralized-identity-an-alternative-to-password-based-authentication/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

this step would be typical of new employee onboarding—and in fact is required by law in some jurisdictions—so employers can save money by doing it digitally (see *Business Efficiencies* in chapter 4).

5. **Alice receives her Employment credential.** Once the Employer’s onboarding policies are satisfied, it issues Alice her new Employment credential.
6. **Alice forms a connection with the Bank where she wants to open a new account.** Note that Alice could do this by scanning a QR code anywhere she might encounter an invitation from the Bank: on a bus, in a subway, in a newspaper advertisement, in an email, on the Bank’s website, or others.
7. **Alice sends a proof of her Government ID credential and her Employment credential to the Bank.** These are the two proofs the Bank requires to open a new account. But for Alice it is just a single step: when her edge agent prompts Alice for approval, she clicks Send and it takes care of the rest.
8. **Alice receives her Bank Account credential.** The Bank’s cloud agent verifies both of the proofs sent by Alice’s edge agent. If they verify, it completes the all-digital process of provisioning a new account and sends Alice her new Bank Account credential.

Now Alice has a new bank account and the Bank has a new customer. From the point Alice has her Government ID and Employment credentials, this process of onboarding a new bank customer, which normally requires an in-person visit and several person-hours of time, can be performed entirely digitally in under a minute. And the result is actually more secure than the equivalent offline process—because the credentials are immediately verified using strong cryptography instead of having bank employees trying to verify paper or plastic credentials that are orders of magnitude easier to fake.

Scenario #6: Alice buys a car

Armed with her new credentials, Alice can now simplify and automate many other kinds of transactions—to the benefit of both herself and the businesses and agencies she interacts with. Figure 3.12 shows the process of Alice buying a new car. In this scenario, we’ll assume the following:

- Alice already has connections with her Bank, Auto Dealer, and Licensing Agency.
- She has picked out the car and negotiated the price with the Auto Dealer.
- She has qualified for a car loan from her Bank.

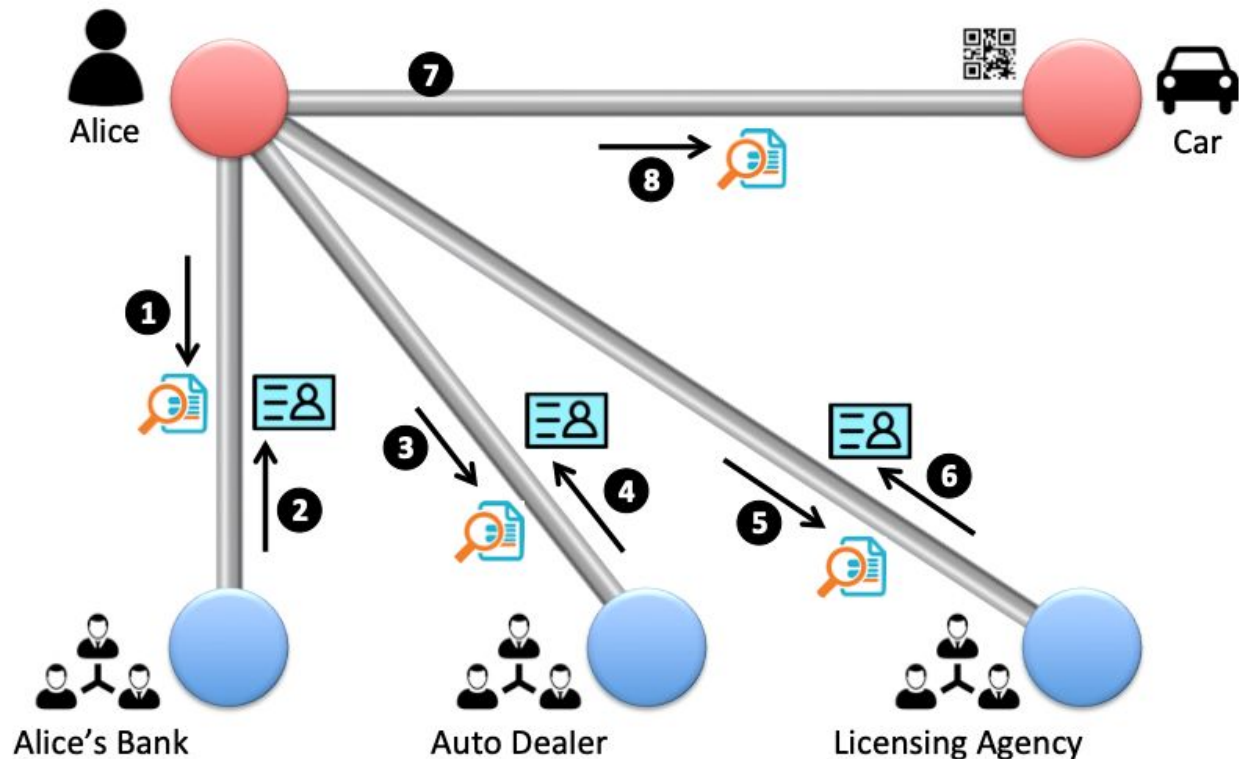


Figure 3.12: Alice buys and registers a car entirely using digital connections and credentials. In the final step, Alice proves her ownership to the car in order to unlock and drive it away.

Here are the steps Alice takes:

1. **Alice proves her identity to her Bank to request a loan.** Because this is a major purchase, the bank requests one or more other credentials (such as her Government ID) to make sure it is really Alice asking for the loan.
2. **Alice receives a Loan credential and Payment Authorization credential from her Bank.** The first credential, Alice can use to prove she has the loan; the second she can use to pay the Auto Dealer.
3. **Alice completes the purchase of the car from the Auto Dealer.** Alice sends proof of her Payment Authorization credential to the Auto Dealer, who arranges for payment (although not shown in figure 3.12, this process could also use an SSI connection between the Auto Dealer and Alice's Bank—see the Digital Banking chapter in Part 2).
4. **Alice receives her Purchase Receipt credential from the Auto Dealer.** This is an example of digital receipt delivered in the form of a verifiable credential—a powerful new tool for both consumers and merchants everywhere.
5. **Alice applies to the Licensing Agency to register her new car.** Alice presents a proof of her Purchase Receipt credential for the car and of her Loan credential from her Bank. Between them they contain all the information—including the vehicle identification number (VIN)—that the Licensing Agency needs to issue Alice a Vehicle

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Registration credential for the car that includes a lien to Alice's Bank.

6. **Alice receives her Vehicle Registration credential.** This goes straight into her digital wallet along with all her other credentials.
7. **Alice forms a connection with her new car.** This is an example of Alice connecting with a thing (as in the Internet of Things) instead of a person or an organization. However it works just like our earlier examples: Alice scans a QR code on the car (for example, on a sticker on the window, or shown on the car's digital display).
8. **Alice proves her ownership to the car.** The QR code requests proof of a Vehicle Registration credential—not just for any car, but for THAT car, with that specific VIN. That is exactly what Alice was issued by the Licensing Authority, so her edge agent sends it to the edge agent for the car (operating as part of the car's onboard computer). Once the car's edge agent verifies the proof, the car unlocks and Alice can drive it away.

Though that final step might seem like a scene from a sci-fi movie, it is very real. In March 2018, one of the authors of this book gave a demonstration to the R&D division of the U.S. Department of Homeland Security of unlocking a car using a verifiable credential. As we move towards all-electric cars, onboard computers, and autonomous vehicles, the key to your car of the future could well be a verifiable credential on your smartphone.⁵

Scenario #7: Alice sells the car to Bob

In our final example, we'll pull together all our previous scenarios and show one involving person-to-person, person-to-business, and person-to-thing relationships: Alice selling the car she bought in scenario #6 to Bob (whom she met either through scenarios #1, #2, or #4). There are enough steps that we'll break it into two diagrams to make it easier to follow. Part A—the financial arrangements—is shown in figure 3.13. This scenario assumes the following:

- Bob and Alice have already formed a connection and negotiated a price for the car.
- Bob has been approved for a car loan by his Credit Union.
- Bob and Alice both live in a jurisdiction served by the same Licensing Agency.

⁵ This would also make it easy to "delegate" car keys to family members, friends, etc. without having to make physical copies. And these digital "keys" can even be time- or usage-restricted, i.e., a child could be given a key to take the car for a trip to the movies but not to drive outside of town.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

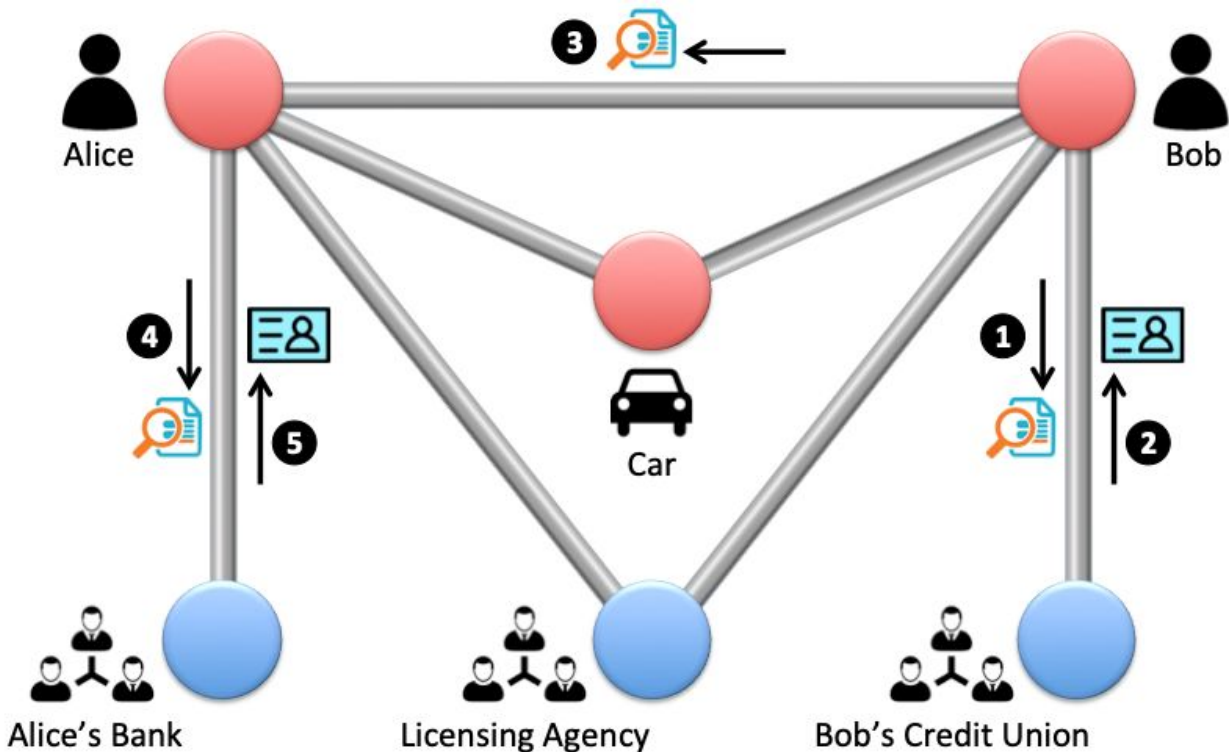


Figure 3.13: Part A of the scenario shows the steps Bob and Alice take to make the financial arrangements for Bob to buy the car.

The steps in Part A are:

1. **Bob proves his identity to his Credit Union to request a loan.** This the same first step Alice took in scenario #6.
2. **Bob receives a Loan credential and a Payment Authorization credential from his Credit Union.** Again, this is the mirror of what Alice did in scenario #6.
3. **Bob sends proof of his Payment Authorization to Alice.** Now Alice just needs to complete her side of the deal.
4. **Alice forwards her proof of the Payment Authorization credential to her Bank.** She also sends a proof of the Loan Credential she wants the payment applied to. Alice's Bank uses these proofs to arrange for payment directly from Bob's Bank.
5. **Alice receives an update to her Loan Credential showing that the loan is paid off.** Alice can use this revised credential not just to transfer the car registration in Part B, but to prove her credit history to anyone else in the future. Note that *Alice can now do this without needing to rely on a centralized credit rating agency.*

Part B (figure 3.14) shows the remaining steps to transfer the car title and registration.

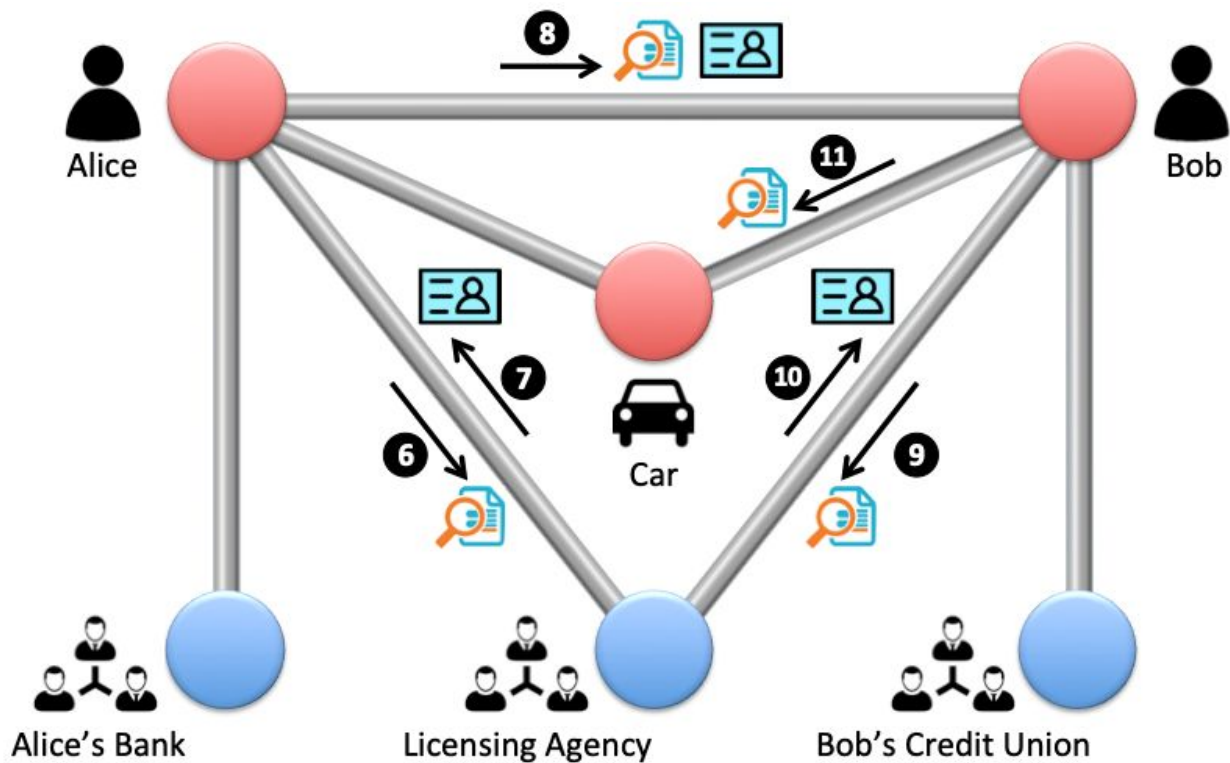


Figure 3.14: Part B of the scenario shows the steps Alice and Bob need to take to transfer the title and registration for the car.

6. **Alice sends proof of her updated Loan Credential to the Licensing Agency.** This proves that Alice's Bank has released its lien and Alice is free to sell the car.
7. **Alice receives an updated Vehicle Registration credential from the Licensing Agency.** Now Alice is ready to complete the sale to Bob.
8. **Alice sends Bob a Purchase Receipt and a proof of her updated Vehicle Registration credential.** These are the two digital artifacts Bob needs from Alice to apply for his own new vehicle registration.
9. **Bob applies to the Licensing Agency to register his new car.** Like Alice in scenario #6, Bob sends a proof of his Purchase Receipt and his Loan credential. He also forwards Alice's proof of her Vehicle Registration credential. The Licensing Agency verifies all proofs to confirm that Alice owns the car free and clear, that she has sold it to Bob, and that Bob has a loan from his Credit Union to buy it.
10. **Bob receives his Vehicle Registration credential.** At the same time, the Licensing Agency revokes Alice's Vehicle Registration credential, so her virtual "key" to the car stops working (for more on credential revocation, see the SSI Architecture chapter in the appendix).
11. **Bob connects to and unlocks his new car.** This is the same step Alice took at the end of scenario #6, only now Bob is the proud new owner.

Summary

This chapter took the SSI building blocks in chapter 2 and put them together to illustrate a variety of common scenarios requiring identity and trust.

- To understand white-collar crime, a detective's mantra is "follow the money." To understand how SSI works, the mantra is "follow the credentials and proofs".
- All that's needed to understand the basic technical flows is to map out the actors who need to establish trust (people, organizations, things), assign them agents (edge and/or cloud), establish the connections between them, and then issue credentials and present proofs in order to meet all the trust requirements.
- These agent-to-agent exchanges can take place face-to-face in the physical world, via a website representing one party, or via websites designed to introduce multiple parties (social networks, dating sites, online communities).
- Once established, an SSI connection can also provide passwordless login to any SSI-enable website or app, and can send and receive credentials and proofs in either direction.
- Entire multi-party workflows, such as applying for a job, buying a car, or selling a car from one owner to another, can all be accomplished by putting together the same basic SSI building blocks.

What may not yet be clear is just how much value can be produced with this new paradigm for achieving digital trust. That is the subject of Chapter 4, where we introduce the SSI Scorecard.

SSI Scorecard:

Major features and benefits of SSI

by Drummond Reed and Alex Preukschat

By now it should be clear that SSI is not just a point technology, like Web shopping carts or mapping apps. It is a fundamental technology shift—akin to the Internet or the Web itself. As such it doesn't have just one primary feature or benefit—or even just a small set of them. Rather it offers an entire spectrum of features and benefits that vary in their impact depending on the use cases of a particular industry or application.

We have developed a simple tool for analyzing this impact: the SSI Scorecard. It classifies 25 major features and benefits of SSI into five categories as shown in table 4.1. In this chapter we'll define each of these categories and the major features/benefits within them. In Part 4, we will use the SSI Scorecard to analyze the impact of SSI across the use cases presented for each of eight industry verticals.

1. **Bottom Line:** dsafds
2. **Business Efficiencies:** the digital transformation of business is the largest impact of SSI. This category will highlight how deper re-engineering of business processes or business process automation (BPA) will be be accelerated with SSI.
3. **User Experience & Convenience:** This category will look at the same five features and benefits as the business efficiencies category, but through the lens of how they benefit the end-user.
4. **Relationship Management:** SSI will increase the trust, productivity, and value of relationships and this category explores at which levels we foresee those changes to happen.
5. **Regulatory Compliance:** Cybersecurity and cyberprivacy infrastructure will be

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

enhanced with SSI and allow companies and people to comply with regulations.

Table 4.1: SSI Scorecard is a tool for analyzing the impact of SSI for any use case, application, industry, or vertical market.

SSI Scorecard	
Category	Feature/Benefit
Bottom Line	Fraud reduction
	Reduced customer onboarding costs
	Improved ecommerce sales
	Reduced customer service costs
	New credential issuer revenue
Business Efficiencies	Auto-authentication
	Auto-authorization
	Workflow automation
	Delegation & guardianship
	Payment and value exchange
User Experience & Convenience	Auto-authentication
	Auto-authorization
	Workflow automation
	Delegation & guardianship
	Payment and value exchange
Relationship Management	Mutual authentication
	Permanent connections
	Premium private channels
	Reputation management
	Loyalty & rewards programs
Regulatory Compliance	Data security
	Data privacy
	Data protection
	Data portability
	RegTech (Regulation Technology)

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

Feature/benefit category #1: bottom line

This category represents the easiest sale in business: features and benefits that deliver **directly to a company's bottom line**, that is, they either make a company more money or save them money—quickly. Following are five ways SSI can do that.

Fraud reduction

The first and fastest way SSI can help the bottom line is reducing fraud. Javelin Strategy reported that in 2016, **15.4 million consumers were victims of identity theft or fraud**, costing a total of \$16 billion dollars in losses.¹ Javelin also reported that new account fraud—criminals opening up new accounts under victims' names—increased from \$3 billion in 2017 to \$3.4 billion in 2018.²

Although the potential savings from fraud reduction varies by industry segment, for some industries it is one of the largest potential sources of savings. For example, the National Health Care Anti-Fraud Association estimates that in 2017 health care fraud costs the United States about **\$68 billion annually** — about 3 percent of the nation's \$2.26 trillion in health care spending.

Bottom line: **even if fraud reduction was the only benefit of SSI**, it would warrant a massive investment by businesses and governments around the world. Indeed, fraud reduction is one of the primary reasons the global credit union industry is embracing SSI as its first major use of blockchain technology. See the Digital Banking chapter in Part 4.

Reduced customer onboarding costs

The cost of customer onboarding varies by industry, but in financial services in particular, the cost of Know Your Customer (KYC) compliance has gone through the roof. According to Thomson Reuters, out of 92% of the firms they surveyed, **KYC onboarding processes cost an average of \$28.5 million**.³ Ten percent of the world's top financial institutions spend at least \$100 million annually on it.⁴ And onboarding a new financial services customer takes anywhere from one to three months on average.⁵

¹ <https://www.cnn.com/2017/02/01/consumers-lost-more-than-16b-to-fraud-and-identity-theft-last-year.html>

²

<https://www.javelinstrategy.com/coverage-area/2019-identity-fraud-report-fraudsters-look-for-new-targets-and-victims-bear-brunt>

³ <https://www.bankingtech.com/2018/09/the-future-of-client-onboarding/>

⁴

<https://www.forbes.com/sites/forbestechcouncil/2018/07/10/know-your-customer-kyc-will-be-a-great-thing-when-it-works/>

⁵ <https://www.opus.com/future-of-kyc/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

There is also a steep cost for not being compliant with these regulations. In 2018, Fenengo reported that a staggering **\$26 billion in fines** had been imposed on financial institutions worldwide for non-compliance with KYC, Anti-Money Laundering (AML), and sanctions regulations in the last decade.⁶

Although SSI is not a silver bullet for all the complexity of automating customer onboarding and ensuring KYC and AML compliance, it is in fact a major new weapon in this arms race—a weapon that benefits all three sides: customers, financial institutions, and regulators. By securely and privately digitizing the information required by these regulations—and enabling it to be cryptographically verified in real time with a full audit trail—SSI has the potential to save all three groups many billions of dollars annually. And it can reduce customer onboarding time from months to days or even hours.

Improved ecommerce sales

Statista forecasts that the total value of **global retail ecommerce will reach \$3.45 trillion in 2019**—up from \$1.34 trillion in 2014 and \$2.84 trillion in 2018.⁷ Nasdaq predicts that by 2040, **around 95% of all purchases** are expected to be via ecommerce.⁸

More than a third of online Black Friday 2018 sales were completed on smartphones.⁹ But on average, only 2.86% of ecommerce website visits convert into a purchase.¹⁰ **In fact, the global cart abandonment rate for ecommerce is close to 70%.** The Baymard Institute averaged out rates from 40 different studies, which give rates from as low as 55% to as high as 81%, to arrive at a global average of 69.89%.¹¹

When you add the fact that **80% of online shoppers stop doing business with a company because of poor customer experience**,¹² the improved convenience, privacy, and safety of shopping with an SSI digital wallet means the impact of SSI on improving ecommerce sales is something that no online merchant can afford to ignore.

Reduced customer service costs

Customer service has become one of the primary battlegrounds of modern business. Gartner predicts that **89% of businesses are expected to compete mainly on customer experience**.¹³

But it is an expensive proposition. Forbes reports that **in 2018 business were losing \$75 billion per year through poor customer service**—up \$13 billion since 2016.¹⁴ According

⁶ [https://www.fenengo.com/press-releases/global-financial-institutions-fined-\\$26-billion-for-aml-kyc.html](https://www.fenengo.com/press-releases/global-financial-institutions-fined-$26-billion-for-aml-kyc.html)

⁷ <https://www.statista.com/statistics/251666/number-of-digital-buyers-worldwide/>

⁸ <https://www.nasdaq.com/article/uk-online-shopping-and-e-commerce-statistics-for-2017-cm761063>

⁹ <http://exploreadobe.com/retail-shopping-insights/http://exploreadobe.com/retail-shopping-insights/>

¹⁰ <https://www.invespcro.com/blog/mobile-commerce/>

¹¹ <https://baymard.com/lists/cart-abandonment-rate>

¹² <https://research.hubspot.com/customer-acquisition-study>

¹³ <https://blogs.gartner.com/jake-sorofman/gartner-surveys-confirm-customer-experience-new-battlefield/>

¹⁴ <https://www.forbes.com/sites/shephyken/2018/05/17/businesses-lose-75-billion-due-to-poor-customer-service/>
©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

to Infosecurity Magazine, just one persistent customer service issue—**lost passwords—costs businesses an average of over \$60 per incident.**¹⁵

SSI can have a massive impact on improving customer experience (CX) and reducing customer-service costs. Passwordless authentication is only the start—the rest of this chapter is filled with examples such as permanent connections (no more losing track of customers), premium private channels, workflow automation, and integrated loyalty management. All of this goes straight to the bottom line—the Temkin Group reports that **even a moderate improvement in CX will boost the revenue of a typical \$1 billion company an average of \$775 million over three years.**¹⁶

New credential issuer revenue

All of the preceding apply to a company's existing lines of business. SSI also opens up new revenue opportunities for a surprisingly wide variety of companies. Any business whose interaction with its customers produces a measure of knowledge about their attributes and interests—or a measure of trust in their behavior—is now in a position to monetize that data in a permissioned and privacy-respecting way: by issuing their customers (suppliers, partners, contractors, and others) verifiable credentials that help them leverage this knowledge. Even better, customers themselves can be the distribution channel for this knowledge to verifiers who need it.

And verifiers will pay for that valuable knowledge for the same reason they pay for customer profile data (from data brokers), credit history (from credit rating agencies), background checks (from background verification companies), and other customer data sources today. SSI can transform this current market much the same way the Web transformed the newspaper classifieds market, the auction market, or the retail market. For example, SSI can provide the following:

- Broader, richer, and more diverse profiles of the customer than those available from third-party sources today.
- Fully permissioned and GDPR-compliant data because the customer is the vehicle for sharing the information for their own benefit.
- Fresher, richer, and more contextual data about preferences, interests, and relationships.
- Selective disclosure of attributes in a way that is all but impossible for direct behind-the-customer's-back data sharing agreements.

Feature/benefit category #2: business efficiencies

As important as the immediate bottom line is, SSI's larger impact will be in re-engineering business processes—a field known as **business process automation** (BPA)¹⁷ or more

¹⁵ <https://www.infosecurity-magazine.com/opinions/how-much-passwords-cost/>

¹⁶ <https://experiencematters.blog/2018/08/21/report-roi-of-customer-experience-2018/>

¹⁷ https://en.wikipedia.org/wiki/Business_process_automation

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

broadly as **digital transformation**.¹⁸ This kind of paradigm shift does not happen very often; it is analogous to the transition businesses underwent from snail mail to email, from phones to fax machines, and from paper to the Web.

As we illustrated in chapter 3, these efficiencies are not limited to just one area of facet of business, but accumulate across entire workflows and even across entire industries. In this section we will look at five areas where SSI can directly impact business efficiencies.

Auto-authentication

Perhaps no area of Web experience is more despised by individuals and companies alike than login. The 2015 TeleSign Consumer Account Security Report said the following:¹⁹

- 54% of people use five or fewer passwords across their entire online life
- 47% of people use passwords that are at least 5 years old
- 7 in 10 people no longer trust passwords to protect their online accounts

In 2019 Auth0 reported that:²⁰

- The average American email address has 130 accounts registered to it.²¹
- The number of accounts per user is doubling every five years.²²
- 58% of users admit to forgetting their password frequently.²³
- The average internet user receives roughly 37 “forgot password” emails a year.²⁴

But besides the sheer hassle, the real impact of username/password-based login is the friction.

- The average person has between 7 and 25 accounts that they log into every day.²⁵
- Around 82% of people have forgotten a password used on a Web site.²⁶
- Password recovery is the number one request to help desks for intranets that don’t have single sign-on portal capabilities.²⁷

In short, by moving from conventional login to SSI auto-authentication—using an SSI digital wallet instead of a username and password—we can finally “kill the password.” It will be like replacing frequent, error-prone toll booths with a wide-open, well-paved highway. Everyone can go about their business faster, more easily, and more safely.

¹⁸ https://en.wikipedia.org/wiki/Digital_transformation

¹⁹ <https://www.entrepreneur.com/article/246902>

²⁰ <https://auth0.com/learn/password-reset/>

²¹ http://blog.dashlane.com/wp-content/uploads/2015/07/MailboxSecurity_infographic_EN_final1.jpg

²² <http://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought/>

²³

<http://www.marketwired.com/press-release/lunabee-survey-finds-that-17-percent-internet-users-often-forget-their-online-passwords-1850682.htm>

²⁴ http://blog.dashlane.com/wp-content/uploads/2015/07/MailboxSecurity_infographic_EN_final1.jpg

²⁵ <http://usablyauthentic.blogspot.com/2011/09/random-factoids-ive-encountered-in.html>

²⁶ <http://passwordresearch.com/stats/statistic97.html>

²⁷ <http://www.nngroup.com/reports/intranet/portals/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Auto-authorization

Authentication (login) is just the first step in most trusted business processes. It proves that you are the rightful owner of an account. But it does not answer the next question: what are you authorized to do? What privileges should you be granted? What actions can you take?

In the world of Identity and Access Management (IAM), this is called **authorization**. It is an even harder problem than authentication. But this is where verifiable credentials truly shine. To use the analogy illustrated in figure 4.1, if verifiable credentials are a hammer, then authentication is only a thumb-tack. Authorization is a full 16-penny nail.



Figure 4.1: While authentication is important, authorization is actually a much bigger nail for the verifiable credentials hammer to hit.

The reason verifiable credentials are such a powerful tool for authorization is that they can solve three hard problems in one stroke:

1. **They can provide exactly the right claims needed for an authorization decision.** These decisions are made by applying the verifier’s access control policies. **Attribute-based access control**²⁸ is based on specific attributes of the identity owner: age, gender, zip code, browser type, and so on. **Role-based access control**²⁹ is based on the role or roles of the identity owner: employee, contractor, customer, regulator, and so on. Either way, verifiable credentials represent the fastest and easiest way for the verifier to request and the holder to supply the precise claims needed.
2. **They can be cryptographically verified in real-time.** To be confident in an authorization decision, a verifier must trust the claims being presented. As explained in chapter 2, the whole point of SSI architecture is so a verifier’s agent can verify the

²⁸ <https://www.axiomatics.com/attribute-based-access-control/>

²⁹ https://en.wikipedia.org/wiki/Role-based_access_control

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

issuer's signature on the holder's proof in seconds.

3. **They can be bound to the holder of the credential as the authorized party.** One of the greatest sources of fraud is stolen usernames/passwords—it has grown into a \$6 billion annual market precisely because they are not verifiable.³⁰ With verifiable credentials, there are several techniques for proving that the claims they contain were issued to the holder of the credential. These include sharing proof of a biometric for the holder and cryptographically linking credentials to a holder using zero-knowledge proofs (ZKPs) .

So using a verifiable credentials model, a verifier's job can be simplified to three steps:

1. **Determining the set of claims** needed for any particular authorization decision.
2. **Determining the issuers—or the governance framework**—that the verifier trusts for those claims.
3. **Making it easy for users to acquire those credentials** so the user experience is as simple and seamless as possible.

But the SSI model can go one step further in business efficiency. Once a user has established a connection with a verifier and approved sharing the claims necessary meet the verifier's policies, **the user can apply his/her own policies** to this process. For example, the user can instruct his/her agent to automatically share the same claims with the verifier when the user needs to repeat a business process in the future (order a supply, approve a budget, publish a web page).

Now the entire authentication and authorization process for a user—even if quite sophisticated—can be automated to the point it is carried out entirely by the user's and verifier's respective agents **including the audit trail needed for accountability**. Obviously this is a big win for users (see *User Experience & Convenience* below), but for verifiers the benefits of auto-authorization can be on the same order as the benefits of credit cards for merchants: customers can perform an essential exchange of information far more easily and painlessly, accelerating business for everyone.

Workflow automation

Every business process has a workflow—the series of steps that must be performed to carry it out end-to-end. Each step that crosses a trust boundary—branch to branch, customer to merchant, supplier to vendor, company to government—typically requires the authentication and authorization processes described above. So SSI agents can already wring out major inefficiencies just by performing auto-authentication and auto-authorization.

But those same agents can also **apply the business logic necessary to orchestrate the steps in the process** no matter how many trust boundaries it crosses.

This is the heart of business process automation (BPA): designing a business process so humans are doing only the steps that require their expertise, awareness, judgement, and

³⁰ <https://qz.com/1329961/hackers-account-for-90-of-login-attempts-at-online-retailers/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

empathy. The rest can be assigned to digital agents (who in some cases may further reassign it to robots). Besides enacting the trust infrastructure necessary to do this work safely, SSI agents are ideal for BPA because they can literally “follow a script”—in this case Javascript or a similar programming language that instructs them to apply each step in the flow of a business process. Agents can be pre-programmed with such scripts, or they can download them dynamically via SSI connections with **orchestration agents** whose job it is to maintain a library of the current scripts required for a specific business process.

SSI is a major leap forward in BPA because process improvements no longer need be limited to a single company or a single supply chain. Like the Internet and the Web, SSI enables BPA workflows to be carried out across any set of trust boundaries, according to any set of policies (governance framework) agreed to be the participants. It is truly “world wide BPA”.

Delegation and guardianship

Digital agents can be given instructions and assigned responsibilities via program code. But most business processes require specific human workers to perform specific functions or make specific decisions as part of the process. How are those humans assigned those responsibilities?

This is the job of a subclass of verifiable credentials called **delegation credentials**. They are how a holder can prove that they have the authority to carry out a specific task or make a specific decision as part of a business process. Common examples in a corporate context:

- **Employees** can be given a delegation credential for the right to send tweets from the company’s Twitter account or post new articles to the company’s blog.
- **Drivers** can be given a delegation credential to pick up and deliver goods on behalf of the company.
- **Officers** can be given specific delegation credentials authorizing them to execute specific types of contracts on behalf of the company, e.g., HR officers to sign an employment contract, procurement officers to sign a purchase order, CFOs to execute a bank order, and so on.
- **Board members** can be given delegation credentials to execute electronic board votes that do not require them to be physically present for a meeting.

There are countless more examples from every context that needs to carry out business processes—governments, schools, non-profits, churches, even households. For example, parents could use delegation credentials to specify how much screen time their children are allowed, or what kinds of foods they are allowed to buy for lunch at school.

The parental example brings up another case: taking responsibility for an individual who is not in a position to wield SSI technology on their own. There are many examples besides babies or young children: the elderly, individuals with disabilities, refugees and displaced peoples, individuals without mobile phones or Internet access.

To enjoy the same rights as everyone else to self-sovereign identity, these individuals need **digital guardians**—individuals or organizations who can operate SSI agents and wallets on their behalf. This form of “complete delegation” is carried out using a **guardianship credential**. It acts much like the digital equivalent of a guardianship order from a court (and in fact may be authorized by such an order). It enables the guardian to set up and operate an SSI agent and wallet on behalf of the dependent and, when necessary, prove that they are acting in the capacity of a guardian. This extends the benefits of SSI to literally any person regardless of physical, mental, or financial capacity.

Payment and value exchange

Mention the phrase “digital wallet” and the first thought that will come to many people’s minds is payments—because that’s the primary task for which we use our physical wallets today. If digital wallets are the core metaphor for SSI, people are going to expect them to be used for payment in addition to identity.



Figure 4.2: Digital wallets are the core metaphor for SSI—so it feels natural that they would be applied to payments.

Indeed, since SSI digital wallets incorporate everything necessary for the trusted exchange of digital information—DIDs, private connections, private keys, agent endpoints—then extending them to the safe exchange of digital payments is very natural. The good news is:

1. From the standpoint of SSI agents, payments are just another type of workflow.
2. SSI wallets can be designed to work with any type of currency (including cryptocurrencies), payment system, or payment network (including credit/debit card networks).
3. With digital wallets and verifiable credentials, payment can be integrated directly into workflows that require KYC and AML as discussed above.

Even better news: **payments are just one type of value exchange that can be automated using SSI**. The term “payment” usually means a specific type of currency—fiat

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

currencies like dollars, pounds, euros, yen, or cryptocurrencies like bitcoin and ether. However there are many other means of value storage and exchange beyond those: points, airline miles, coupons, and many other different types of loyalty programs. And SSI digital wallets and agents can be used for all of them as fast as these value exchange systems can be translated to verifiable credentials and agent-to-agent protocols. (See the Scorecard entry for *Loyalty and reward programs* later in this chapter.)

This in turn means payments can be integrated into almost any business process workflow at almost any level of assurance and regulatory compliance. Payment automation is the frosting on the SSI-enabled BPA cake.

Feature/benefit category #3: user experience & convenience

This category will look at the same five features and benefits as the last category (business efficiencies), but this time through the lens of how they benefit the end-user.

Auto-authentication

How much do users hate passwords? A July 2019 study by MobileIon reported in Security InfoCenter³¹ said that when users encounter password troubles:

- 68% feel disrupted
- 63% feel irritated and frustrated
- 62% feel they have wasted time.

The same study found that:

- IT security leaders felt they could reduce their risk of breach by almost half (43%) by eliminating passwords.
- 86% of those security leaders would do away with passwords if they could.
- 88% of these leaders believed that in the near future, mobile devices will serve as your digital ID to access enterprise services and data.

In February 2019, user-centric biometric authentication leader Veridium published a study of more than 1000 U.S. adults who have experience with biometrics (such as Apple's TouchID or FaceID) found that 70 percent wanted to expand their use into everyday login.³² Speed (35 percent), security (31 percent), and not having to remember passwords (33 percent) were cited as the primary incentives.

On May 1, 2018, Microsoft announced in a blog post that it was "Building a world without

³¹ <https://www.securitymagazine.com/articles/90530-in-10-it-leaders-want-to-eliminate-passwords>

³² <https://www.businesswire.com/news/home/20190213005176/en/Veridium-Survey-Reveals-Strong-Consumer-Sentiment-Biometric>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

passwords”.³³ To quote:

Nobody likes passwords. They are inconvenient, insecure, and expensive. In fact, we dislike them so much that we’ve been busy at work trying to create a world without them – a world without passwords.

This is why Microsoft has been a major supporter of DIDs—the decentralized identifiers at the core of SSI—and are building DID-based passwordless authentication into multiple products.³⁴

In short, the age of the password is about to pass, and for users everywhere it could not come fast enough. There will be rejoicing in the streets.

Auto-authorization

If passwordless auto-authentication will replace the login screen, **auto-authorization will replace many (but not all) Web forms**. Can you hear more rejoicing?

Here are some realities about online forms:

- 81% of people have abandoned a form after beginning to fill it out.³⁵
- 29% of people cite security reasons as one of their main concerns when it comes to completing online forms.³⁶
- More than 67% of site visitors will abandon your form forever if they encounter any complications; only 20% will follow up with the company in some way.³⁷
- 23% of people will not fill out your checkout form if you require them to create a user account.³⁸
- Better checkout design can reduce form abandonment by as much as 35%, which translates into nearly \$260 billion in recovered orders.³⁹

³³ <https://cloudblogs.microsoft.com/microsoftsecure/2018/05/01/building-a-world-without-passwords/>

³⁴ <https://www.microsoft.com/en-us/security/technology/own-your-identity>

³⁵ <https://themanifest.com/web-design/6-steps-avoiding-online-form-abandonment>

³⁶ <https://wpforms.com/online-form-statistics-facts/>

³⁷ <https://wpforms.com/online-form-statistics-facts/>

³⁸ <https://wpforms.com/online-form-statistics-facts/>

³⁹ <https://wpforms.com/online-form-statistics-facts/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

When you stop to think about it, SSI auto-authorization solves almost every typical complaint about online forms:

- **There is no typing.** All the information being requested by the verifier is being transferred from claims your digital wallet. And even if new self-attested data is requested, your agent can capture it for you so you'll never have to type it again.
- **Your connection is your account.** The whole idea of "click here to automatically create an account with this form data" goes away. You automatically have an "account" anywhere you have a connection.
- **Data verification is built-in.** The main point of verifiable credentials is that the claims data has already been vetted by the issuer.
- **Security is built-in.** All proofs and data sent by your SSI agent automatically use your encrypted private connection with the verifier.
- **Privacy and selective disclosure are built-in.** First, verifiers can now ask only for the minimum information they need—reducing their potential liability. Secondly, the proof your agent sends can be read only by the verifier. If the verifier needs a copy of the underlying data (e.g., asks you to share your actual birthdate instead of just proving you are over 18), your agent should be able to automatically warn you if that data will not covered by a satisfactory privacy policy or governance framework.
- **Auditing is built-in.** Your agent can automatically track all the information you share—without requiring you to share that history with anyone else.

In addition, verifiable credentials let you prove many more things about yourself—as an individual, a student, an employee, a volunteer, or in any other role you play—than you could prove via any Web form today. With auto-authorization, **your ability to perform tasks online comes much closer to your ability to perform those same tasks in the real world**—i.e., using your physical wallet, paper credentials, and face-to-face verification—but orders of magnitude faster.

Workflow automation

From the standpoint of the end-user, SSI has the potential to reduce workflow steps that currently can take hours or days to as little as a few buttons on a smartphone. One such scenario—the selling of a car and the transfer of the title and registration from one owner to another—was described in detail in the final scenario in chapter 3.

Many more of these scenarios are covered in detail in Part 4, where we examine the impact of SSI across eight different industries and market verticals. You will see the same patterns repeated over and over: a business process carried out, step by step, by employees, contractors, suppliers, regulators, and other participants exchanging verifiable credentials (or digitally signed messages authorized by these credentials) between their agents and wallets. For every step, all of the following are performed automatically for the user:

1. **Authentication** that the user is the correct party.
2. **Authorization** that the user has the authority.
3. **Verification** that the step is being performed in the right sequence of the business

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

- process (and that its preconditions have been met).
4. **Validation** that the claims or messages meet the requirements of the business process.
 5. **Routing** of the credential or message produced to the next agent or agents needed in the process.
 6. **Logging** of the action taken to provide a full digitally-signed audit trail (or even automated reporting to regulators—see the very last section of this chapter).

Perhaps the ultimate example of consumers experiencing the convenience of SSI-enabled workflow automation is the perennial **change-of-address** problem. An individual is moving house and needs to inform dozens if not hundreds of agencies, suppliers, and contacts about the new address. Despite the advent of the Internet and the Web, this is still an excruciatingly labor intensive process for the individual. The main reason? **Account takeover**.⁴⁰ Fraudulent change-of-address is the first step in hijacking a bank account, credit card account, company account, or other valuable account in order to steal from it—or use that account to steal from others. So companies need to add extra hoops to ensure it's really you requesting a change-of-address.

With SSI and verifiable credentials, change-of-address can be performed in three easy steps:

1. **Obtain a verifiable credential of your new address** from a widely trusted issuer.
2. **Send a proof of that credential over all your connections** who need to know your new address.
3. **Each of their back-end systems can verify the proof** and update their systems with your new address with high confidence that it is valid.

Voila. Tens of hours of human labor and hundreds of dollars in business savings for every single change-of-address notification. Given that in America alone an average of 35 million people move house every year,⁴¹ this alone adds up to **hundreds of millions of man-hours and hundreds of billions of dollars in savings every year**.

Delegation and guardianship

As we described above, delegation credentials are what enable much of this workflow automation magic. Thankfully, the process of obtaining (or assigning) delegation credentials is just another workflow. The delegator first establishes a connection with the delegate (or vice versa) and then issues a credential granting the necessary authorizations.

Delegation credentials can be revised or revoked as conditions and positions change, with both the requirements and the current status being maintained by orchestration agents. All of this can be set forth in one or more governance frameworks that define the legal and business rules applying to the entire business process, whether it is taking place entirely inside one company, across a supply chain, across an entire industry, or in a wide-open

⁴⁰ https://en.wikipedia.org/wiki/Credit_card_fraud#Account_takeover

⁴¹ <https://www.move.org/moving-stats-facts/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

process such as international shipping that crosses multiple industries and government jurisdictions. As a general rule, if humans can define the rules of the process, including who can make what decisions when, then SSI agents, wallets, and verifiable credentials can be used to automate the necessary information exchanges.

The end result is that so many of today's most difficult user experience challenges—especially entering or interpreting data presented by machines—can be simplified by being able to focus just on the analysis and decisions that humans really need to make.

Payment and value exchange

For decades the challenge of moving money safely has the focus of entire industries—banking, credit unions, credit cards, and now cryptocurrencies. It is the very heart of human economic activity. And, like the human heart, it is the most vulnerable to attackers. As Willie Sutton famously answered when asked why he robbed banks, “Because that’s where the money is.”⁴²

So there has always been a tension between making it easier to move money and making it safe to do so. Every means of value transfer from Pony Express to Paypal has spawned a new legion of criminals to exploit it. Cryptocurrencies—arguably the most friction-free way to move money ever invented—are no different. Coindesk reported that in the first nine months of 2018, nearly \$1 billion had been stolen from cryptocurrency exchanges and other crypto holders.⁴³

While SSI is not a panacea, it does provide a complete infrastructure for trusted information exchange. This includes payments and other forms of value exchange as discussed above. With all of the protections that SSI digital agents, wallets, connections, and verifiable credentials provide, SSI could be the infrastructure that finally—from the perspective of end-user experience—makes digital payments one-click easy AND secure at the same time.

The effect on digital commerce can be profound. Amazon’s one-click purchasing capability famously helped vault it to the forefront of ecommerce. With the expiration of Amazon’s one-click patent⁴⁴ and the arrival of SSI payments infrastructure, a feature that once exclusively belonged to Amazon could now become “**one click everywhere**”.

Feature/benefit category #4: relationship management

While saving time and money is important, there is another category of features and benefits that is not purely monetary: those that increase the trust, productivity, and value of relationships.

Customer relationship management (CRM) is already a dominant industry in its own right.

⁴² https://en.wikipedia.org/wiki/Willie_Sutton

⁴³ <https://www.coindesk.com/nearly-1-billion-stolen-in-crypto-hacks-so-far-this-year-research>

⁴⁴ <https://digiday.com/marketing/end-era-amazons-one-click-buying-patent-finally-expires/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

In January 2019 Forbes reported that:⁴⁵

- **CRM now makes up nearly 25% of the entire enterprise software revenue market.**
- Worldwide spending on CRM software grew 15.6% to reach \$48.2B in 2018.
- Salesforce is the leader, with 19.5% of the CRM market, followed by SAP at 8.3%).

The impact of SSI on CRM has long been anticipated by a movement known as **vendor relationship management (VRM)**. Its leader, Doc Searls, pithily summarizes it as “the inverse of CRM”, i.e., VRM how people can control their relationships with companies. For details see _____, the chapter Doc has contributed to this book.

Without stealing Doc’s thunder, this section will examine five key ways that SSI will enable better relationship management in both directions.

Mutual authentication

The first place SSI can improve relationships is right at the very start. This is when the parties are most vulnerable—when they are meeting each other for the first time, especially digitally.

On the web today, this is a struggle from both sides. First, imagine how hard it is for a website to prove that is authentic when **phishing sites** and **phishing emails** have become so good that even trained professionals can have a hard time spotting them.⁴⁶ Verizon’s 2018 Data Breach Incident Report said phishing accounted for 93% of all data breaches.⁴⁷ Between October 2013 and May 2018, the U.S. Federal Bureau of Investigation (FBI) reported \$12.5 billion in losses to companies due to phishing.⁴⁸

Now turn the tables and think about how hard it is for **you**, the end-user, to prove to a website that *anything* about yourself is authentic. Most of us have a hard time even proving we are **human**. How many of us have tripped up over one of these?

⁴⁵ <https://www.forbes.com/sites/louiscolumnbus/2019/06/22/salesforce-now-has-over-19-of-the-crm-market/>

⁴⁶ During the writing of this book, one of the co-authors received an email from his bank (a household brand) that was so realistic, it took three phone calls to determine it was a phishing attempt.

⁴⁷ https://enterprise.verizon.com/resources/reports/DBIR_2018_Report.pdf

⁴⁸ <https://www.ic3.gov/media/2018/180712.aspx>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.



Figure 4.3: A CAPTCHA (completely automated public Turing test to tell computers and humans apart) can often be hard for even a real human to pass.

If it's that hard to prove you are even human, how are you supposed to prove:

- You are over or under a certain age
- You live in a certain place
- You have a certain degree
- You have a certain job
- You have a certain income

Generally these tasks are impossible over the Internet without: a) proving your real-world identity to the website (somehow), and then b) having the website independently verify information about you from some authoritative source (like a credit rating agency)—or worse, some data broker who may or may not have accurate information about you (let alone permission from you to share it).

This problem has become acute in both directions—and **SSI can solve it in both directions**. The beauty of SSI auto-authentication (see the earlier sections) is that your agent can not only share verifiable credentials with a website, but the website's agent can share verifiable credentials with you. Both agents can automatically verify the credentials on behalf of their owners to make sure they meet their respective policies. If so, the relationship can proceed without a hitch. If either agent spots a problem, it can immediately flag its owner—or simply deny the connection, so its owner is never bothered.

This is how all our digital relationships should work—mutual auto-authentication on both sides that will put phishers permanently out to sea.

Permanent connections

The second major benefit SSI brings to relationship management is a feature no network has ever offered before: **permanent connections**. By permanent, we mean a connection

that can last literally *forever* if both parties want it to.⁴⁹

How can SSI make such a promise? No other digital connection—a phone number, an email address, a Twitter handle, a Facebook friend, a LinkedIn connection—can make that promise. Why? **They all depend on some form of intermediary service provider for the connection to keep working.** And no intermediary service provider can promise to always be in business and always maintain your connections no matter what you do or what your connections do.

In fact, most of them promise you *the exact opposite*: they can terminate your service at any time, for any reason. Just read your terms of service.

With SSI—and the decentralized networks that make it work—there is no intermediary service provider. **Your connections belong to you.** You—or the other party—are the only ones who can terminate a connection because one or both of you want to end the relationship.

How valuable is this to you—the individual who wants to stay in touch with the people, organizations, and businesses in your life when you move, change jobs, graduate from school, or change service providers? And how valuable is it to all of your contacts and vendors who are trying to keep track of you? Earlier in this chapter we quantified that simply automating change-of-address could save millions of hours and billions of dollars annually. Multiply that by all the other information we’d like to keep “in sync” with each other over permanent connections (see below), and the total savings could be an order of magnitude greater.

Premium private channels

Permanence is not the only benefit of SSI connections. Because they are based on DIDs, DID documents (with public keys) and the DIDComm protocol, all of them natively support **end-to-end secure encrypted communications.**

From a marketing perspective, we call these **premium private channels.** “Premium” because they are exclusive to you and your connection—they are not shared by anyone else. “Private” because all your communications are automatically encrypted and decrypted by your respective agents without any effort on your part. “Channels” because you can use them to send and receive any messages or content your respective agents can “speak”.

So your connections can be used—without any permission from anyone else—with any SSI-enabled application: messaging apps, voice and video apps, data sharing apps, social networking apps, productivity apps, payment apps, games, and so on. Every one of these apps can have access to every SSI feature described in this chapter.

Messaging apps have been moving in this direction for some time. Apple iMessage,

⁴⁹ The mathematical techniques used to generate DIDs produce numbers so large that, even after thousands of years, the chances of generating the same ones are infinitesimally small.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

WhatsApp, Signal, and Telegram all support end-to-end encryption in one form or another. Others, like WeChat and Alipay in China, have also integrated messaging with secure payment and so many other plug-in functions such that many Chinese spend their entire day living and working in these apps.⁵⁰

SSI makes premium private channels a universal capability that can be integrated with any app and that can work across any trust boundary—just like the Internet and the Web. You will find multiple examples in Part 2 of this book. For instance, CULedger, a global consortium of credit unions developing SSI infrastructure for the credit union industry, plans to use premium private channels to request secure, digitally-signed authorizations and consents from credit union members for actions that would otherwise require members to send a fax or make an in-person visit to a credit union office.

Reputation management

Reputation systems have become an essential feature of doing business on the Web. For example, a Spiegel Research Center study showed nearly 95% of shoppers read online reviews before making a purchase.⁵¹ A 2016 Harvard Business School study said a one star increase on Yelp can lead to a 5-9% increase in revenue for a merchant.⁵²

However it is precisely because they are so valuable that attacking reputation systems has become big business. A February 2019 study from Fakespot, which analyzes customer reviews, revealed that 30% of reviews on Amazon are fake or unreliable, and a whopping 52% of reviews posted on Walmart.com are inauthentic.⁵³ Worse, the rise in fake reviews is undermining consumer confidence in reputation systems. A Bright Local 2018 study said 33% of all consumers reported spotting “lots” of fake reviews, and that the number went up to 89% for 18-to-34-year-olds who are savvier at detecting the signs of a fake review.

⁵⁰ <https://medium.com/@wechatminiprogrammer/alipay-vs-wechat-pay-an-unbiased-comparison>

⁵¹ <http://spiegel.medill.northwestern.edu/online-reviews/>

⁵² <https://www.hbs.edu/faculty/Pages/item.aspx?num=41233>

⁵³

<https://www.cbsnews.com/news/buyer-beware-a-scurge-of-fake-online-reviews-is-hitting-amazon-walmart-and-other-major-retailers/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Amazon has long tried to fight this gaming with various protection measures, including its **Amazon Verifier Purchase** program that is supposed to ensure the reviewer actually bought the product.



Figure 4.4: An Amazon Verified Purchase marking is supposed to confirm that the reviewer bought the reviewed product—but they are not difficult to work around.

However this only works if a reviewer actually purchased the product at Amazon, and even then, it is relatively easy for a savvy marketing company to subsidize these purchases or find other ways around Amazon's rules. And if Amazon has these issues, imagine the scope of the problem for smaller sites that have only a tiny fraction of Amazon's security budget.

At this point in the book it should be obvious how SSI can help with this problem. First, reputation systems can require verifiable credentials for reviewers, weeding out the bots. Second, they can require a verifiable credential for a product purchase—a **verifiable receipt**—so programs like Amazon Verifier Purchase can work independently of any particular retailer. Thirdly, reviewers can actually start to build reliable reputation independent of not just any product vendor but of any retailer—so we can start developing an ecosystem of widely trusted independent reviewers that can become the Web equivalent of, say, Walter Mossberg of the Wall Street Journal or Jon Udell in his days at Byte Magazine.

In short, **reputation management can become an integral part of relationship management**. Any two parties that develop a connection and engage in interactions with each other—purchases, contracts, consulting, or just community engagement—should be able to provide verifiable reputational feedback to each other and to the larger community.

The implications extend to any online survey, poll, or vote where it matters that: a) real human beings and not bots are participating, b) each unique person has only one vote, and c) people need to be accountable for voting honestly and not trying to game the system. Gaming these types of systems with fake votes has become such a common attack that the security community gave it the name **Sybil attack** after the famed case of multiple-personality disorder that was the subject of a 1973 book and a 1976 movie.⁵⁴

As fake reviews, fake sites, and fake news multiply like rabbits online, the ability for SSI to counter Sybil attacks and anchor the trustworthiness of reputations systems may become

⁵⁴ https://en.wikipedia.org/wiki/Sybil_attack

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

one of its most valuable contributions to the future health of the Web.

Loyalty and rewards programs

Every relationship involves an exchange of value of some kind between the two parties—even if it's just an exchange of pleasantries among neighbors. If that exchange is monetary, our earlier discussions of how SSI enables new forms of payment apply. But the stronger a relationship grows, the higher the probability it involves some form of non-monetary value exchange.

Rewards programs are a perfect example. Whether they involve miles, points, stamps, or some other measure of value, they are an informal, direct, relationship-based way of thanking a customer for past loyalty and incenting future loyalty. And they work:

- 69% of consumers say choice of retailer is influenced by where they can earn customer loyalty/rewards program points.⁵⁵
- A 5% increase in customer loyalty will increase the average profit per customer by 25-100%.⁵⁶
- 76% of consumers think that loyalty programs are part of their relationship with brands.⁵⁷
- The Loyalty Management market is expected to grow from \$1.4 billion in 2015 to \$4.0 billion by 2020.⁵⁸

However for consumers today, managing loyalty programs is anywhere from mildly inconvenient to downright irritating. Imagine if every retailer you dealt with required you to not only use a different type of money, but **a different wallet**—one dedicated to their specific store. That would be ridiculous—yet that is how loyalty programs work today.

SSI-based relationship management (also called vendor relationship management or VRM as noted above) can turn that on its head. Now every loyalty program can be designed to use its own premium private channel **to the consumer's own SSI digital wallet**. No matter what kind of loyalty currency is involved, they can all be managed securely and privately in one place. Consumers gain dramatically greater convenience and control; retailers gain simpler and more effective loyalty programs that can also take advantage of all the other features of SSI.

Feature/benefit category #5: Regulatory compliance

Our final category may be the least “sexy” but in some ways the most significant because it

⁵⁵ <http://www.prweb.com/releases/2013/11/prweb11372040.htm>

⁵⁶ http://en.wikipedia.org/wiki/Loyalty_Effect

⁵⁷ <https://www.invespcro.com/blog/customer-loyalty-programs/>

⁵⁸ <http://www.prnewswire.com/news-releases/loyalty-management-market-worth-usd-40-billion-by-2020-545897032.html>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

covers how SSI can contribute to the strength of our global cybersecurity and cyberprivacy infrastructure. In this section we'll cover five major ways SSI can help all actors in the global economy comply with regulations designed to keep us safe while at the same time encouraging greater economic activity through open and fair competition.

Data security

We could begin this section with a shower of statistics about the state of security on the Internet, but they are all summed up by this 2015 quote in Forbes from Gina Rommety, CEO of IBM, when she addressed the CISOs (Chief Information Security Officers), CIOs, and CEOs of 123 companies in 24 industries:⁵⁹

"We believe that data is the phenomenon of our time. It is the world's new natural resource. It is the new basis of competitive advantage, and it is transforming every profession and industry. If all of this is true – even inevitable – then cyber crime, by definition, is the greatest threat to every profession, every industry, every company in the world."

That same Forbes article said market estimates of the size of the worldwide cybersecurity industry range from \$77 billion in 2015 to \$170 billion by 2020. It is one of the fastest growing of all enterprise software segments.

By attacking the very root of the problem—digital identity—SSI represents a sea-change in cybersecurity. SSI agents will help users generate and manage private keys, automatically negotiate pairwise pseudonymous DIDs, form secure connections, and communicate over premium private channels that provide the data security required by regulations such as HIPAA (Health Insurance Portability and Accountability Act) in the United States and GDPR (General Data Protection Regulation) in Europe. This alone will lock down acres of current vulnerabilities—one reason the U.S. Department of Homeland Security funded much of the research into the decentralized identifier⁶⁰ and decentralized key management⁶¹ standards that are the foundation of SSI.

SSI can also "detoxify" personal data so it can no longer be used for identity theft and related cybercrimes. Today this personal data is valuable because if a thief has enough of it, he/she can impersonate you to either break into current accounts or open new accounts in your name. But with verifiable credentials, **your personal data alone can no longer be used to steal your identity**. If the thief does not have your private keys, he/she cannot produce proofs of your verifiable credentials.

This means **breaches of huge corporate databases containing personal data will**

⁵⁹

<https://www.forbes.com/sites/stevemorgan/2015/11/24/ibms-ceo-on-hackers-cyber-crime-is-the-greatest-threat-to-every-company-in-the-world/>

⁶⁰ <https://www.sbir.gov/sbirsearch/detail/867797>

⁶¹

<https://www.dhs.gov/science-and-technology/news/2017/07/20/news-release-dhs-st-awards-749k-evernym-decentralized-key>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

become a thing of the past. Unlike usernames, passwords, and other personal data today, your private keys will never be stored in some centralized corporate database that serves as a giant honeypot⁶² for criminals. They are always stored in your local devices, with an encrypted backup copy in the cloud (or wherever you direct it to go). This means that to steal an identity, a thief has to break into your personal SSI wallet(s), one at a time.

This is like forcing criminals to eat by catching tiny minnows one at a time instead of spearing a whale. Give criminals that choice and they will find another way to eat.

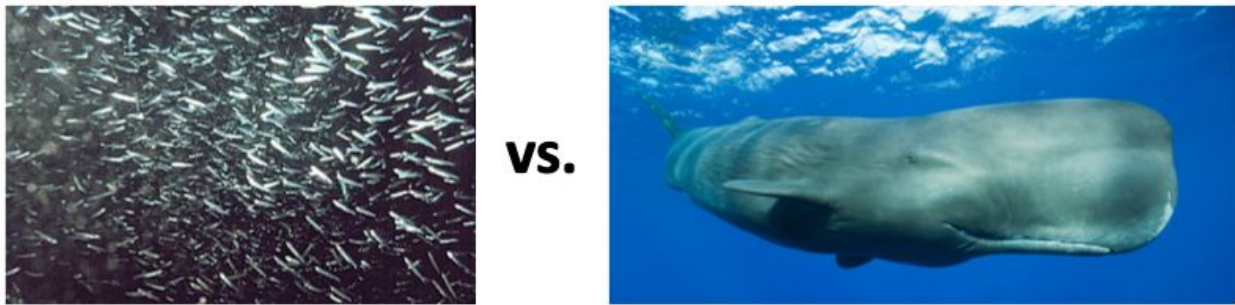


Figure 4.5: SSI will force identity thieves to try to steal private keys one wallet at a time (left) vs. break into giant corporate honeypots of personal data (right).

Data privacy

Security and privacy go hand in hand, and privacy is every bit as much of a concern on the Internet today. In June 2018 Entrepreneur Magazine reported 90% of Internet users were “very concerned” about Internet privacy.⁶³ The same article reported that the Cambridge Analytic data scandal⁶⁴ so tarnished Facebook’s reputation that only 3% of users trust how Facebook is handling their personal data (and only 4% trust Google).

Part of what has pushed Internet privacy to this crisis level is the sheer economic value of personal data in today’s digital economy. In an April 2019 review of Shoshana Zuboff’s book *The Age of Surveillance Capitalism*,⁶⁵ The Nation magazine said:⁶⁶

Zuboff shows that these increasingly frequent invasions of our privacy are neither accidental nor optional; instead, they’re a key source of profit for many of the 21st century’s most successful companies. Thus, these companies have a direct financial stake in the broadening, deepening, and perfecting of the surveillance they already profit from—and in making sure that it remains legal.

Shifting the balance of power in privacy and personal data control will not be easy.

⁶² [https://en.wikipedia.org/wiki/Honeypot_\(computing\)](https://en.wikipedia.org/wiki/Honeypot_(computing))

⁶³ <https://www.entrepreneur.com/article/314524>

⁶⁴ https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal

⁶⁵ <https://www.shoshanazuboff.com/new/the-age-of-surveillance-capitalism-comments-and-reviews/>

⁶⁶ <https://www.thenation.com/article/shoshana-zuboff-age-of-surveillance-capitalism-book-review/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

However, SSI can help companies comply with applicable privacy legislation in three specific ways:

1. **Selective disclosure.** SSI verifiable credential exchange technology—specifically for credentials that use zero-knowledge proof cryptography—enables companies to request proofs of exactly the personal data they need and no more. For example, a company can request proof you are over a specific age, rather than your actual birthdate.
2. **Verifiable consent.** Many consumers have no idea where companies get their personal data. Did it come from online forms they filled out? From marketing partners? From third party data brokers? With verifiable credentials, there is a clear, verifiable chain-of-consent to share data that starts with the individual and can easily be traced and audited by the responsible SSI agents.
3. **Governance frameworks.** Today, most privacy policies are highly custom documents written by lawyers to protect specific companies—not your privacy. Which is why ten years ago a study by privacy researchers Lorrie Faith Cranor and Aleecia McDonald estimated **the average person would require 76 work days** to read the privacy policies on the websites they visit, which for the United States alone would add up to **53.8 billion hours** or **\$781 billion in labor costs**.⁶⁷ With SSI, company-specific privacy policies could begin to be replaced by governance frameworks that: a) are uniform across all the sites that adopt them, b) can be developed in open public forums to represent the best interests of all stakeholders, c) can be pre-approved by regulators to comply with their requirements, and d) can be designed to incorporate the other protections and advantages of SSI.

These steps could become the first Internet-scale implementation of the principles of *Privacy by Design* as developed and advocated by former Ontario Information and Privacy Commissioner Ann Cavoukian.⁶⁸

Data protection

Although closely related to data privacy, data protection goes beyond privacy controls to enumerate a larger set of specific principles for the protection of an individual's personal data. Although the European Union's General Data Protection Regulation (GDPR)⁶⁹ is the best known data protection legislation, it is by no means the only one. The California Consumer Privacy Act (CCPA)⁷⁰ is setting a new standard for data protection regulation in the U.S., and many other countries have or are enacting data protection laws along the same lines.

In addition to the data privacy compliance mechanisms listed the previous section, there are

⁶⁷

<https://www.theatlantic.com/technology/archive/2012/03/reading-the-privacy-policies-you-encounter-in-a-year-would-take-76-work-days/253851/>

⁶⁸ https://en.wikipedia.org/wiki/Privacy_by_design

⁶⁹ https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

⁷⁰ https://en.wikipedia.org/wiki/California_Consumer_Privacy_Act

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

other very specific ways that SSI can enable both individuals to exercise their rights and companies to comply with their responsibilities under these data protection acts. To wit:

1. **Pseudonymous identifiers.** GDPR encourages the use of pseudonyms to minimize correlation. SSI connections use pairwise pseudonymous peer DIDs by default.
2. **Data minimization.** GDPR requires collecting no more personal data that is necessary for the purpose for which it is being processed. SSI selective disclosure and ZKP credentials are ideal for meeting this requirement.
3. **Data accuracy.** GDPR requires that personal data must be accurate and kept up to date. SSI enables data controllers to request personal data supplied by verifiable credentials from reputable issuers—and which can be automatically updated by those issuers when the data changes.
4. **Right of erasure** (also known as the “right to be forgotten”). Enshrined by Article 17 of GDPR, this can be one of the most challenging requirements because it means the data controller (the company) must give the data subject (the individual) a means of confirming what personal data a company holds while at the same time not opening a security hole for attackers. Thankfully this is precisely the job that SSI connections and premium private channels were designed for. The data subject can use auto-authentication and auto-authorization to request access to the data, and if desired send a digitally-signed request for erasure over the connection. All of these actions can be securely audited for later verification of compliance.

Data portability

GDPR enforces one more data protection right that is deserving of its own discussion: data portability. This is the right that allows data subjects to obtain data that a data controller holds on them and reuse it for their own purposes. In the words of the first paragraph of Article 20 of the GDPR:

The data subject shall have the right to receive the personal data concerning him or her, which he or she has provided to a controller, in a structured, commonly used and machine-readable format and have the right to transmit those data to another controller without hindrance from the controller to which the personal data have been provided...

GDPR is only one of many new regulations requiring partial or complete data portability across data controllers. Another is the second Payment Services Directive (PSD2) which is driving open banking in the EU.⁷¹ Both of these were preceded by the requirement for local telephone number portability (LNP) in the U.S. Telecommunications Act of 1996, which in the U.S. also applies to mobile number portability (MNP). MNP is also required to varying degrees by legislation in Africa, Asia, Australia, Latin America, and Canada.

SSI is ideal for data portability because it solves so many of the deep security and privacy issues as described in the previous sections. The secret is that SSI connections make it easy

⁷¹ https://en.wikipedia.org/wiki/Payment_Services_Directive

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

for the data **to flow in and out of connections with the individual data subject's own agent**, where the individual can always exert complete control over the data and the terms and conditions under which he/she is willing to share it with other parties.

Under this architecture—and especially under governance frameworks designed specifically to work with this architecture—personal data should be able to flow freely between systems while meeting the security, privacy, and control requirements of GDPR and other data protection regulations.

RegTech (Regulation Technology)

As substantial as all these breakthroughs are, the one regulators may get most excited about is the ability to connect directly to SSI ecosystems *themselves*. In other words, by deploying their own SSI agents and connecting directly to the companies being regulated, **regulators can be directly “in the loop” of transactions with specific regulatory requirements**—such as KYC requirements for opening a bank account, AML requirements for money transmission, or ATF requirements for purchases of certain types of goods.

For example, if a money transmission between two SSI-enabled parties exceeds the threshold over which a financial institution must apply additional AML compliance measures, this can be communicated in real time between the bank's SSI agent and the regulator's SSI agent. This has the potential to change the very nature of regulation enforcement from **an after-the-fact spot-check auditing activity to a real-time rules-driven monitoring activity**—decreasing enforcement costs, speeding up enforcement actions, and improving the quality of enforcement data all at the same time—a rare triple win for government.

This is a highly desirable development given the skyrocketing costs of compliance cited earlier in this chapter. It is also consistent with the rapid growth of the global RegTech market, which Research and Markets expects to grow from USD 4.3 billion in 2018 to USD 12.3 billion by 2023, at a Compound Annual Growth Rate (CAGR) of 23.5%.⁷² With the ability of SSI technology to connect secure, private, permissioned SSI agents to any business process requiring regulation, that growth figure could be much higher.

Using the SSI Scorecard in Part Four

In this chapter we have:

- Introduced a scorecard of 25 major features and benefits of SSI divided into five major categories.
- Defined each category and described the five most important features and benefits within it.
- Provided evidence of the market impact of each of these key features and benefits

⁷² https://www.researchandmarkets.com/research/r8ktnm/global_12_3?w=5

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

and shown how many of them are interrelated.

As its name indicates, the SSI Scorecard is a tool for analyzing the impact of SSI on any particular use case, application, industry, or vertical market. We will do just that in Part Four of this book, where we will examine in-depth use cases for SSI across eight vertical markets. Each chapter will wrap up with a scorecard for how much impact SSI is likely to have in that market along and why. This should help tie together the building blocks, example scenarios, and features/benefits we have covered in Part One with the real-world scenarios we are going to be looking at in Part Four.

SSI architecture: the big picture

by Daniel Hardman

The purpose of this chapter is to take the basic building blocks, usage scenarios, and benefits we introduced in Part 1 and place them into an overarching picture of SSI architecture. As we will keep repeating, SSI is young, and as this chapter will explore, many facets are still in formation. Still, the basic layering has emerged, along with several of the key underpinning standards, so the main questions are how they will be implemented and how interoperable they might be depending on the design choices. In this chapter, Daniel Hardman, VP Architecture of Evernym and a major contributor to most of the core standards and protocols discussed here, will walk you through four levels of SSI architecture, the key components and technologies at each level, and the critical design decisions facing architects and implementers throughout the stack. This chapter sets the stage for the rest of the chapters in Part 2 that dive deeper into specific SSI technologies.

In chapter 2 we discussed the basic building blocks of self-sovereign identity (SSI). Those building blocks represent important commonality—all approaches to the problem agree on them. However, like automobile design in the early 1900s, the young market is producing much innovation and divergence in the details. Some are inventing with two wheels, some with four or three. Some favor steam, while others favor internal combustion engines powered by gasoline or diesel.

In this chapter, we will identify important decision points in decentralized digital identity architectures, notable choices for each, and the benefits and challenges they entail.

5.1 The SSI stack

Some types of divergence are irrelevant; others are fundamental. Separating the two is an

important topic among identity architects, because it affects interoperability. A first attempt to describe all key choices in SSI stacks was made at Internet Identity Workshop (IIW) in October 2018.¹ Conference participants listed eleven layers of technology that are likely to appear in SSI solutions, and described the dependencies between them.

¹ See Terbu, Oliver. "The Self-sovereign Identity Stack". 27 Jan 2019.
<https://medium.com/decentralized-identity/the-self-sovereign-identity-stack-8a2cc95f2d45>.
Retrieved 23 Dec 2019.

However, subsequent experience has shown that eleven layers was more than necessary to describe the fundamental architectural dependencies in SSI. Seeking to distill just the essence, identity architects from various backgrounds collaborated under the umbrella of the Hyperledger Aries project in 2019 to produce the four-layer paradigm that we'll use throughout this chapter. The bottom layers, while important, are essentially invisible plumbing; the top layers embody concepts visible to ordinary users and tie directly into business processes, regulatory policies, and legal jurisdictions. The model looks like this:

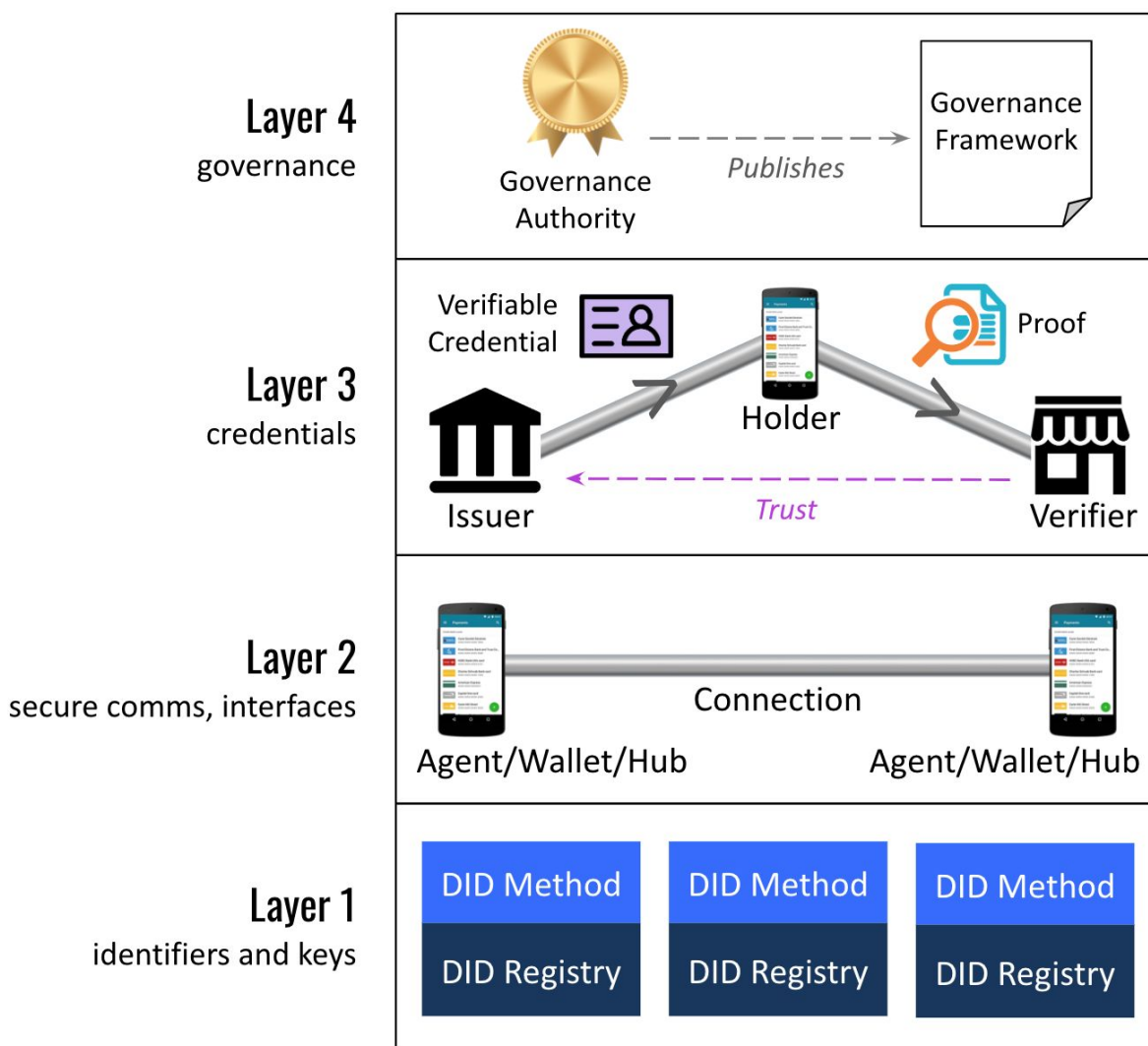


Figure 5.1: The SSI stack as a four layer model, where the bottom two layers are primarily about achieving technical trust and the top two layers are about achieving human trust

Each of these layers embodies key architectural decisions and each has important consequences for interoperability. In the balance of this chapter we will discuss the details

of each layer, building from the bottom up.

5.2 Layer 1: Identifiers and public keys

Layer 1 is at the bottom of the stack, where identifiers and public keys are defined and managed. These are often called **trust roots**, and like real trees, the stronger the roots, the stronger the tree. This layer needs to guarantee that all stakeholders agree to the same truth about what an identifier references, and how control of this identifier may be proved using cryptographic keys. It must also allow every party in the ecosystem to read and write data without reliance on or interference from central authorities—a property widely referred to in the blockchain community as “censorship resistance”.²

There is broad consensus among the SSI community that the best way to provide these features is to expose a **DID registry** (a decentralized source of truth for Decentralized Identifiers—DIDs—described in Chapter 2) through a **DID method** (conventions for how a DID registry must be used). However, this approach still allows much to vary in the particulars.

For example:

1. What’s the best way to implement a decentralized system? Should it be permissioned or permissionless? How should it be governed? How well does it scale?
2. How, exactly, should DIDs be registered, looked up, and verified on this decentralized system to ensure they are secure while still protecting privacy?
3. What other data needs be stored on the decentralized system to support the higher layers of the SSI stack?

As the chapter on Decentralized Identifiers will discuss, there are currently over 40 DID methods defined in the informal DID method registry maintained by the W3C Credentials Community Group.³ That reflects the great diversity of DID methods developed so far. In this section we’ll discuss the major architectural categories into which they fall.

5.2.1 Blockchains as DID registries

As noted at the beginning of this book, SSI was born because blockchain technology introduced an exciting new option for how to do the decentralized PKI (DPKI) that could unlock the power of verifiable credentials (VCs). So naturally blockchains in their many forms have been the first option for Layer 1. Indeed, as of the writing of this book, more than three-quarters of the DID methods in the DID method registry are blockchain- or DLT-based.

Besides the well-known public blockchains that support cryptocurrencies, this family of

² https://en.wikipedia.org/wiki/Internet_censorship_circumvention

³ <https://w3c-ccg.github.io/did-method-registry/>

technologies includes distributed ledgers (e.g., Hyperledger), distributed directed acyclic graphs (e.g., IOTA), and distributed hashtables (e.g., IPFS). “Blockchain” has become a general umbrella term for the entire family, although experts argue about exactly how to apply these definitions.⁴ In principle all of these technologies should be able to achieve the SSI design goals of strong security, censorship resistance, and self-service without central administration.

However, these blockchains also differ in important ways, including:

- How much, if at all, they integrate with cryptocurrencies and payment workflows as an identity concern.
- Whether identity operations are a first-class feature of the blockchain, or a generic feature (for example, smart contracts) that intersects with identity only in certain cases.
- How expensive they are to operate and scale.
- What latency and throughput they deliver.
- How they are permissioned and governed.
- How they address regulatory compliance and censorship resistance.

In summary, it is best to look at blockchain as one option for meeting the requirements of Layer 1, but not the only option—and not necessarily a good option depending on: a) the design and implementation of the blockchain, and b) the design and implementation of a specific DID method for using that blockchain. For example:

- A blockchain that’s prohibitively expensive to use may put SSI out of reach for many of the world’s disadvantaged;
- A blockchain that’s too slow may never be adopted.
- A blockchain that is permissioned may not adequately address concerns about censorship-resistance.
- A blockchain whose codebase is tightly controlled by a small group may not be considered trustworthy enough for broad adoption.

Since this category is so broad, let’s next look specifically at two major subdivisions within it: general-purpose public blockchains and special-purpose SSI blockchains.

5.2.2 Adapting general purpose public blockchains to SSI

Although blockchain technology is less than a decade old, it already has two “granddaddies”: Bitcoin and Ethereum. Collectively their market capitalization at the time of writing this chapter—measured in terms of the total value of their respective cryptocurrencies—is more than double that of all other cryptocurrencies combined.⁵ So they are obviously strong candidates for the stability and broad developer support needed to

⁴ See <https://blockchainhub.net/blockchains-and-distributed-ledger-technologies-in-general/>

⁵ <https://coinmarketcap.com/> as of 15 January 2020.

engender trust in them as public infrastructure. This explains why some of the first SSI implementations—by Learning Machine (Bitcoin) and uPort (Ethereum)—targetted these blockchains. It also explains why over 25% of the DID methods registered in the W3C CCG DID Method Registry are designed to work with either Bitcoin (3 methods) or Ethereum (8 methods).⁶

The common theme among these methods is that they use the cryptographic address of a transaction on the ledger—a *payment address*—as the DID. Payment addresses are already opaque strings, world-unique, and governed by cryptographic keys. Although it predates the W3C standards for verifiable credentials and DIDs, the Blockcerts standard for educational credentials, championed by Learning Machine, is built on this approach, and credential issuance and verification (Layer 3) work nicely atop it.⁷

However, payment addresses don't have rich metadata. This means there is no obvious way to contact the holder of an address, which limits interactions to a "don't call us, we'll call you" model. An identity holder can approach another party and ask to interact, but the opposite flow is problematic, because discoverability is limited. The public cannot start from a payment address and reach out to its owner at will.

Payment addresses are also global correlators, which raises privacy concerns. Law enforcement identified the operators of Silk Road because payment addresses are traceable; although that case may have had a good outcome, similar techniques could be used to harass, find dissidents, or surveil. Some interesting technical solutions have been proposed to mitigate this effect with payment addresses, but implementations are still pending, and more technical and legal measures may be required.

Some blockchains are designed so that payment addresses are retired as funds move or keys rotate. Without workarounds, this can complicate the stability of reference required by a good user experience (UX) for identity.

5.2.3 Special-purpose blockchains designed for SSI

By 2016, developers were creating the first blockchains designed explicitly to support SSI. The first came from Evernym, who developed an open source code base for a public permissioned ledger, where all the nodes would be operated by trusted institutions. Evernym helped organize the Sovrin Foundation to become the non-profit governance authority for the blockchain and then contributed the open source code to the Foundation.⁸

⁶ <https://w3c-ccg.github.io/did-method-registry/> as of 19 January 2020.

⁷ <https://www.blockcerts.org/>

⁸ <https://sovrin.org/>

The Sovrin Foundation subsequently contributed the open source code to the Hyperledger project hosted the Linux Foundation, where it became Hyperledger Indy.⁹ It joined other Hyperledger business-oriented blockchain operating systems including Fabric, Sawtooth, and Iroha. As the only Hyperledger project designed expressly for SSI, Indy is now being run by other public permissioned SSI networks including those implemented by Kiva, the non-profit operator of the world's largest microlending platform.¹⁰

The next entrant in purpose-built SSI blockchains was Veres One created by Digital Bazaar.¹¹ It is a public permissionless blockchain optimized to store DIDs and DID documents in JSON-LD, a rich semantic graph format based on Resource Description Framework (RDF). The Veres One blockchain is being used for multiple SSI pilots involving verifiable credentials for supply chains and provenance.¹²

Blockchains that are purpose-built for identity, such as Veres One and Hyperledger Indy, have transaction and record types that make DID management easy. But there is more to Layer 1 than just DIDs. In particular, with the zero-knowledge proof (ZKP) credential format supported by the W3C Verifiable Credential Data Model 1.0 standard and the Hyperledger Aries open source code base (discussed in the sections on Layers 2 and 3), several other cryptographic primitives must be supported at Layer 1.¹³

1. **Schemas.** This how issuers define the claims (attributes) that they wish to include on a verifiable credential. By putting schema definitions on a public blockchain, they are available for all verifiers to examine to determine semantic interoperability (a very big deal when it comes to data sharing across silos).
2. **Credential definitions.** The difference between a credential definition and a schema is that the credential definition is published on the ledger to declare the specific claims, public key, and other metadata that will be used by a specific issuer for a specific version of a verifiable credential.
3. **Revocation registries.** This is a special data structure, called a cryptographic accumulator, that is used for privacy-respecting revocation of verifiable credentials. See our chapter on Cryptography for more details.
4. **Agent authorisation registries.** This is a different type of cryptographic accumulator used to add additional security to SSI infrastructure by SSI users to authorize and de-authorize specific digital wallets on specific devices—for example if they are lost, stolen, or hacked.

Equally important is what does *not* go on the ledger: **any private data**. Although early experiments in blockchain-based identity had the notion of putting an individual's

⁹ <https://wiki.hyperledger.org/display/indy>

¹⁰ <https://www.slideshare.net/SSIMeetup/kiva-protocol-building-the-credit-bureau-of-the-future-using-ssi>

¹¹ <https://veres.one/>

¹² <https://blockchain.enterprisesecuritymag.com/cxinsight/blockchain-a-us-customs-and-border-protection-perspective-nid-1055-cid-56.html>

¹³ <https://sovrin.org/wp-content/uploads/2017/04/What-Goes-On-The-Ledger.pdf>

credentials and other personal data as encrypted object directly on the blockchain, subsequent research and analysis has shown that is a terrible idea. First, all encryption has a limited lifespan, so writing private data to an immutable public ledger will mean it eventually will be cracked. Secondly, even encrypted data has privacy implications just by watching who writes and reads it. Thirdly, it presents massive issues with the EU General Data Protection Regulation (GDPR) and other data protection regulations around the world.¹⁴

5.2.4 Conventional databases as DID registries

Although it may seem antithetical to a decentralized approach, the user databases of Internet giants have shown that modern Web-ready database technologies can achieve the robustness, global scale, and geographical dispersion needed by a DID registry. Some have even proposed that a massive social network database like Facebook's, or a comprehensive government identity database like India's Aadhaar, could be the basis for rapid adoption of DIDs—citing broad coverage, proven ease-of-use, and existing adoption as a rationale.

However, such databases are neither self-service or censorship-resistant. Trust in these databases is rooted in centralized administrators whose interests may not align with those of the individuals they identify. Privacy is questionable when a third party mediates every login or interaction. This is true whether the organization that runs them is a government, a private enterprise, or even a charity. Such centralization undermines one of the most important design goals of SSI (and also of the Internet itself): eliminating single points of control and failure. For this reason, most practitioners of SSI discount traditional databases as a viable implementation path for Layer 1.

5.2.5 Peer-to-peer protocols as DID registries

As DID methods have matured, SSI architects have realized that there is an entire category of DIDs that do not need to be registered in a back-end blockchain or database at all. Rather these DIDs and DID documents can be generated and exchanged **directly between the peers that need them** to identify and authenticate each other.

The "DID registry" in this case is the digital wallet of each of the peers—each is the "root of trust" for the other—along with trust in the protocol used to exchange these **peer DIDs**. Not only does this approach have massively better scalability and performance than any blockchain- or database-based DID method, but it also means the DIDs, public keys, and service endpoints are completely private—they never need to be shared with any external party at all, let alone on a public blockchain.

¹⁴ See the Sovrin Foundation's 35-page white paper on GDPR and SSI for a detailed analysis: <https://sovrin.org/data-protection/>

The primary peer-to-peer DID method, called `did:peer:`, was developed under the auspices of the Hyperledger Aries project. It is defined in the *Peer DID Method Specification*.¹⁵ Peer DIDs are generated directly in the digital wallets of the two peers involved and exchanged using their digital agents, so in reality Peer DIDs are a Layer 1 solution that's implemented entirely at Layer 2 of the SSI stack. However the Peer DID method is developing “fall-back” solutions for situations where one or both of the peers move to new service endpoints on the network and lose touch with one another; some of these require clever triangulation against a public blockchain at Layer 1.

*Triple-signed receipts*¹⁶ is an example of a protocol that also solves this problem without any blockchain at all. It was originally described as an evolution to standard double-entry accounting, but can be used to solve double-spend problems in identity as well (for example, claiming to one party that a key is authorized, while claiming to another party that a key is not). Each party in the protocol signs a transaction description that includes not just the inputs, but also the outputs (resulting balance). An external auditor signs as well. Once all three signatures have accumulated, there is no question about the truth of a transaction—and because the signed data *includes the resulting balance* in addition to the inputs, no previous transactions need be consulted to know the effect of the transaction.

Given the rich possibilities, we anticipate additional peer-based protocol solutions to Layer 1 functionality will be developed over the coming years, all of which will just add to the overall strength of this foundational layer of the SSI stack.

5.3 Layer 2: Secure communication and interfaces

If Layer 1 is about establishing decentralized trust roots—either publicly verifiable or peer-to-peer—then Layer 2 is about establishing trusted communications between the peers relying on those trust roots. This is the layer where the digital agents, wallets, and hubs we introduced in Chapter 2 live. And where their secure DID-to-DID connections are formed.

Even though actors of all kinds—people, organizations, and things—are represented by these agents, wallets, and hubs at Layer 2, the trust established between those actors at this layer is still only *cryptographic trust*, in other words, trust that:

- A DID is controlled by another peer.
- A DID-to-DID connection is secure.
- A message sent over a connection is authentic and has not been tampered with.

These are all conditions that are *necessary but not sufficient* to establish human trust because they don't yet establish anything about the person, organization, or thing identified by the DID. For example, a DID-to-DID connection doesn't care whether a remote party is ethical or honest or qualified—only whether talking back and forth can be done in a way

¹⁵ <https://openssi.github.io/peer-did-method-spec/>

¹⁶ http://opentransactions.org/wiki/index.php/Triple-Signed_Receipts

that's tamper-proof and confidential. For that we will have to move up to Layers 3 and 4.

The architectural issues at this layer fall into two main categories: protocol design and interface design. In this section we first discuss the two main approaches to Layer 2 protocol design, and then the three main approaches to interface design.

5.3.1 Web-based protocol design using TLS

The first approach is rooted in a simple observation: we already have a ubiquitous, robust mechanism for secure Web communication in the form of Transport Layer Security (TLS). Why reinvent the wheel?

Proponents of this view are building systems where parties talk to one another by making RESTful web service calls. The tooling and the libraries for such mechanisms are well understood, and millions of developers are comfortable with them. Progress is relatively easy.

However, challenges exist:

- Although TLS can be applied to other protocols, its primary adoption success has been with HTTP. This means that TLS is only an immediate answer when at least one of the parties is running a web server.
- It is inherently two-party (client and server).
- It requires a relatively direct request-response interaction, with both parties being online at the same time.
- The server is passive; it reacts when called, and can trigger webhooks or callback URLs—but it can't reach out to the other party on its own, unless the other party is also running a web server.
- The security model for TLS is asymmetric: servers use X.500 digital certificates (introduced in Chapter 2 and discussed in more detail in Chapter 8 on DIDs); clients use passwords or API keys or OAuth tokens. This tends to perpetuate a *power imbalance*, where organizations having high-reputation server certificates dictate the behavior of low-reputation clients. It is antithetical to the peer-to-peer philosophy of decentralization in SSI. It also makes the control of DIDs and their cryptographic keys a secondary (or even redundant) concern.
- Its privacy and security guarantees are imperfect. *SSL visibility appliances* are well-known hacking tools that insert a man-in-the-middle for each TLS session that runs through an institutional LAN. X.500 certificate authorities are operated by humans that can be gamed. And TLS has no story for securing communications data at rest or outside the secured channel.

For all these reasons, although there is a great deal of SSI code development following the conventional Web architecture path of using HTTPS, at least as much if not more development effort is going into a more inclusive approach.

5.3.2 Message-based protocol design using DIDComm

This second approach is called **DID communication** (**DIDComm** for short). It conceives of Layer 2 as message-oriented, transport-agnostic, and rooted in interactions among peers. In this paradigm, communications between agents is similar in concept to email: it is inherently asynchronous; may involve broadcasting to multiple parties at the same time; delivery is best-effort; and replies may arrive over different channels or not at all.

However DIDComm differs from SMTP email in that:

- Recipients are identified by a DID rather than an email address.
- All communication is secured (either encrypted, signed, or both) by the keys associated with DIDs. This is true even of data at rest.
- Messages may be delivered over any transport: HTTP, Bluetooth, ZMQ, the file system / sneakernet, AMQP, mobile push notifications, QR codes, sockets, FTP, SMTP, etc.
- Security and privacy guarantees are the same regardless of transport. A single route may use more than one of these transports.
- Routing is adapted for privacy; it is designed so no intermediary knows a message's ultimate origin or its final destination—only the next hop. This allows for (but does not require) the use of mix networks and similar privacy tools.

DIDComm can easily use HTTP or HTTPS in a request-response paradigm, so this approach to protocol design is actually a superset of the Web-based approach. However, DIDComm provides flexibility for other situations, including those where a party is connected only occasionally, or where channels contain many untrusted intermediaries.

DIDComm's most important technical weakness is its novelty. Although tools used by the world's web developers (CURL, Wireshark, Chrome's Developer Tools, Swagger, etc) are relevant and somewhat helpful with DIDComm, tools that understand DIDComm natively are immature. This makes DIDComm a more expensive choice in the early days of the SSI ecosystem.

Nevertheless, DIDComm is gaining momentum rapidly. Although it was originally incubated within the Hyperledger developer community (Indy and Aries),¹⁷ in December 2019 the growing interest beyond Hyperledger led to the formation of the DIDComm Working Group at the Decentralized Identity Foundation (DIF).¹⁸ Already there are roughly a dozen implementations of various parts of DIDComm for different programming languages.¹⁹

¹⁷ <https://github.com/hyperledger/aries-rfcs/tree/master/concepts/0005-didcomm>

¹⁸ <https://medium.com/decentralized-identity/dif-starts-didcomm-working-group-9c114d9308dc>

¹⁹ <https://github.com/hyperledger/aries>

5.3.3 Interface design options

From an architectural standpoint, interface design is about how SSI infrastructure becomes available for programmatic use by developers who want to solve real-world problems for individuals and institutions. The approaches here depend to some degree on protocol design. For Web-based client/server protocols, Swagger-style API interfaces are a natural complement; with DIDComm, peer-to-peer protocols are a more natural model. However, both underlying protocols can be paired with either style of interface, so the question is actually somewhat orthogonal.

SSI solutions tend to emphasize one of three answers to the interface question. However these answers are not mutually exclusive; all approaches overlap. What differs is the focus.

20

5.3.4 API-oriented interface design using wallet dapps

The API-based approach stems from the philosophy that SSI features are best exposed through decentralized web and mobile apps (dapps) and associated Web 2.0 or Web 3.0 APIs on complementary server-side components. Here, the prototypical SSI dapp for an individual is a wallet that holds all their cryptographic material and provides a simple UX for requesting and providing credential-based proof about identity. This wallet also interfaces directly or indirectly with a blockchain to verify data.

The Blockcerts app and the uPort app are examples of this model. These apps interface with Learning Machine's server-side Issuing System or with uPort's Ethereum-based back end, respectively. Programmers are encouraged to write their own apps or automation to call these backend APIs and leverage the identity ecosystem. Because the approach is conceptually straightforward, the learning curve is not steep. This should aid adoption and popularity.

This paradigm works well for SSI use cases where individuals want to prove things to institutions, because individuals carry mobile devices and institutions operate servers with APIs. However, when individuals are the receivers rather than the givers of proof, and when identity owners are IoT devices or institutions instead of people, there is some impedance with the overall model. How can an IoT device hold a wallet and control a DApp, for example? Perhaps we will see clever extensions of this model in the future.

²⁰ <https://www.hyperledger.org/blog/2019/07/23/rhythm-and-melody-how-hubs-and-agents-rock-together>.

5.3.5 Data-oriented interface design using hubs

Digital Bazaar, Microsoft, and various other players in the DIF and W3C CCG communities have championed a view of identity that is data-centric. In this paradigm, the major task of actors using SSI infrastructure is to discover, share, and manage access to identity data.

The focus of management in this view is an **identity hub** (or, in W3C parlance, an **encrypted data vault**)—a cloud-based point of presence through which identity data is accessed by a web API. Hubs are services configured and controlled by an identity controller. In the case of individuals, they are usually imagined to be sold as an SaaS subscription, providing a sort of *personal API*. Hubs can also be used by institutions or devices. Importantly, hubs are not necessarily conceived of as directly representing a particular identity controller; they can independent services that manage data on behalf of any number of identity controllers, as directed by each individual controller.

Figure 5.2 illustrates the role an identity hub plays in the SSI ecosystem envisioned by DIF.

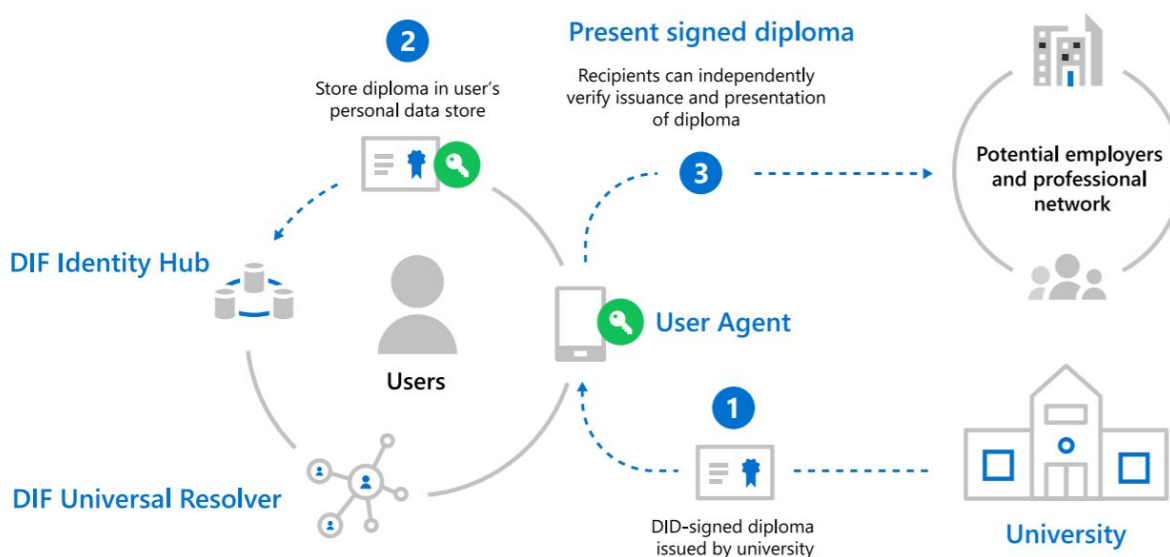


Figure 5.2: An identity hub shown as part of the SSI ecosystem envisioned by DIF

Although hubs in theory can execute code on a schedule or trigger, most design documents focus on the hub as a passive responder to external clients. Most examples assume they are constantly online to provide a stable portal for interaction, and that they behave like a specialized web server. Thus, they were originally imagined as HTTP-oriented and RESTful, with service endpoints defined in a DID document.²¹ However as DIDComm has evolved,

²¹ <https://github.com/decentralized-identity/identity-hub/blob/master/explainer.md>

some hub APIs have been recast as message-based and capable of using Bluetooth, NFC, message queues, or other transport mechanisms besides HTTP. However, the architectural center of hub design remains fundamentally rooted in data sharing.

Security for hubs is provided by encryption of messages, using keys declared in a DID document. Early discussions of encryption approaches centered on the JSON Web Token (JWT) format,²² part of the Javascript Object Signing and Encryption (JOSE) stack,²³ but the particulars are still in development.

The data hub model is attractive for companies that want to sell hosting services to consumers and enterprises. It is also convenient for institutions that want to contract with consumers to generate value from their data. On the plus side, programming against hubs is likely to be very easy for developers. On the minus side, it is not clear whether consumers want to manage their identity as a service in the cloud. There are also privacy, security, and regulatory issues with hosting providers that need to be explored. This is one area where market validation is particularly needed.

²² https://en.wikipedia.org/wiki/JSON_Web_Token

²³

https://en.wikipedia.org/wiki/Web_Cryptography_API#JavaScript_Object_Signing_and_Encryption_%28JOSE%29

5.3.6 Message-oriented interface design using agents

The Hyperledger Aries project and the Sovrin community have championed an SSI interface paradigm that focuses most heavily around active *agents* and the messages and interactions they share. Agents are direct representatives of an identity, rather than being indirect or external representatives like hubs. When you interact with an agent, you are interacting directly with the entity whose identity the agent represents.

As described in chapter 2, an SSI digital agent can be hosted anywhere: on mobile devices, IoT devices, laptops, servers, or anywhere in the cloud. Architectural diagrams for agent ecosystems don't put a blockchain at the center, and words like "client" and "API" are generally absent. The mental model might be best visualized as shown in figure 5.3.

Here, a loose collection of agents—the circular nodes—is interconnected in various ways. The dotted lines between the agent nodes are connections: Alice to Acme as employee, Faber College to Bob as school, Alice to Bob as acquaintance. Some agents have many of these connections; others have few. Some agents may even connect to the same peer more than once (Alice to Carol as coworker, Alice to Carol as ham radio buddy).

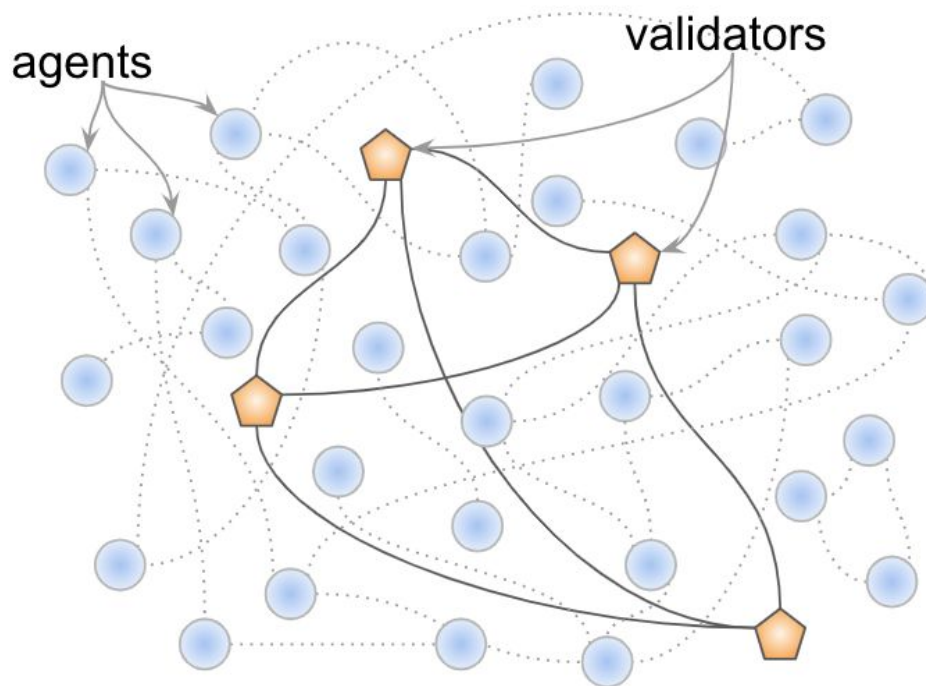


Figure 5.3: An example of a Layer 2 Hyperledger Aries ecosystem—a loose mesh of peer agents interspersed with some acting as validator nodes for DIDs at Layer 1

Connections are constantly forming, and new agent nodes are constantly appearing. Each connection requires a private, pairwise peer DID for each party. As explained earlier in this chapter, these DIDs and their DID documents are not written to any blockchain, but are stored in the wallets for each agent. The entire mesh is fluid and fully decentralized; connections are direct between peers, not filtered through any interposed authority.

The mesh of nodes is horizontally scalable, and its performance is a function of the collective capacity of the nodes to interact and store data. Thus it scales and performs in the same way that the Internet does. The overwhelming majority of protocol exchanges and meaningful interactions that produce business or social value take place directly between the agent nodes. Once a peer DID connection is formed through an exchange of DIDs and DID documents, the resulting channel (the dotted and solid lines in figure 5.3) can be used for anything the respective agents need it for: issuing credentials, presenting/proving credentials, exchanging data, securely messaging between humans, and so on.

Mingled with the circular agent nodes in figure 5.3 are a few 5-sided shapes. These are validator nodes that provide public blockchain services as a utility to the rest of the ecosystem. They maintain connections with one another to facilitate consensus as the blockchain is updated. Any agent node can reach out to the blockchain to test a community truth (such as check the current value of a public key in a DID document). They can also write to the blockchain (for example, create a schema against which credentials can be issued). Because most DIDs and DID documents are not public in this ecosystem, the bulk of traffic between agent nodes doesn't need to involve the blockchain at all, thus avoiding attendant scalability, privacy, and cost issues. However, blockchains are still very useful for establishing public trust roots—for example the DIDs and public keys for institutions that issue (and revoke) credentials that can be broadly trusted.

The interface to this world is *decentralized n-party protocols*. These are recipes for sequences of stateful interactions representing the business problems solved by applications. Because agents are not assumed to be steadily connected like servers, and because data sharing is only one of an agent's concerns, protocols unfold as encrypted JSON messages that are exchanged, not as APIs that are called (although underlying APIs may be called to trigger specific messages).

Mostly, a developer who wants to support these protocols has to produce and consume JSON over whichever set of transports the target population of agents support. Specific protocols are in various stages of standardization and implementation for every popular interaction pattern:

- issuing credentials
- creating proofs with credentials
- chatting
- scheduling
- buying and selling

- negotiating
- sharing data
- setting terms and conditions
- digitally signing documents
- creating legally binding contracts

The list is endless—and that is the point. Agent-oriented architecture (AOA) is designed to model the variety and flexibility of interactions in the real world, while still providing the security, privacy, and trust guarantees necessary to perform transactions that otherwise require direct human intervention at an order of magnitude higher cost and slower speed.

On the downside, since the AOA model focuses on loosely coordinated actors with behavior guided by convention, it is more complex than alternative approaches. It is more sensitive to network effects. It is harder to build and debug. It is also more emergent and therefore harder to characterize with confidence. So only time will tell whether AOA will be successful in the market.

5.4 Layer 3: Credentials

If Layers 1 and 2 are where *cryptographic trust*—trust between *machines*—is established, Layers 3 and 4 are where *human trust* enters the picture. Specifically, Layer 3 is the home of the verifiable credential trust triangle introduced in Chapter 2. This is so relevant to this section that we reproduce it here as figure 5.4.

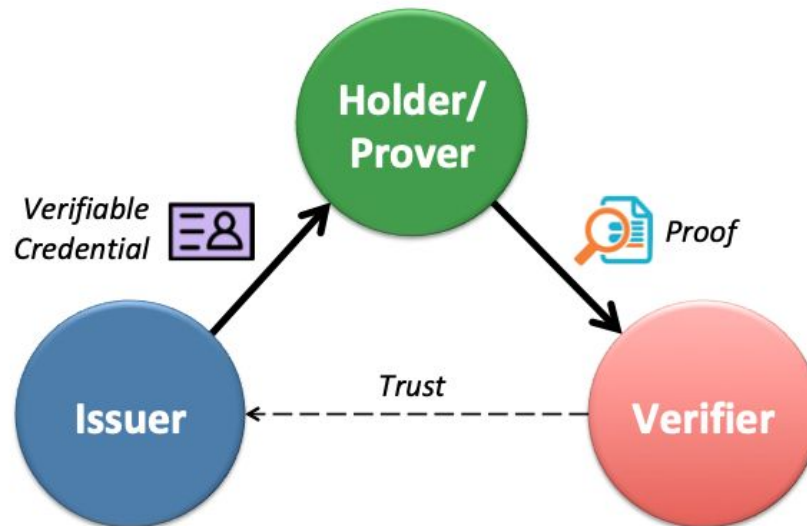


Figure 5.4: The verifiable credential trust triangle that is at the heart of all credential exchange (physical or digital)

This trust triangle is where humans are able to answer questions like:

- Is the party asking for my phone number really my bank?
- Is the person digitally signing this contract really an employee of X company?
- Does the person applying for this job really have Y degree?
- Is the seller of the product I want to buy really based in Z country?
- Does the person who wants to buy my car hold a valid passport?
- Is the device I am plugging into my wall socket really approved by Underwriter Laboratories?
- Is the coffee in this bag really sustainably grown in Nicaragua?

Of course, this list of questions is endless—they encompass everything we as humans need to know to make trust decisions, whether in our personal capacity, our business capacity, our governmental/citizen capacity, or our social capacity. The goal of Layer 3 is to support interoperable verifiable credentials that can be used in all of these capacities—from any issuer, for any holder, to any verifier.

Given all the capabilities we have described at Layers 1 and 2, interoperability at Layer 3 comes down to two straightforward questions:

- What format of verifiable credential will the parties exchange?
- What protocol will the parties use to exchange it?

Unfortunately the answers to these questions are anything but straightforward. In fact the SSI community currently has significant divergence on the answers—meaning Layer 3 is at significant risk of not being interoperable.

In this section we will dive into the details of the different answers. First with regard to credential formats—what credentials look like on the wire and on disk—and then with regard to the protocols used to exchange them.

5.4.1 JSON Web Token format

One approach to credential format is to use the well-established JWT specification (RFC 7519).²⁴ JWTs (often pronounced “jots”) were designed as a carrier of authentication and authorization grants; they are used extensively in OAuth, Open ID Connect and other modern web login technology. JWTs have good support in various programming languages (search for “JOSE JWT library”). For example, the uPort SSI ecosystem is JWT-centric.

One challenge of JWT tooling is that it provides no help for interpreting the rich metadata that is desirable in a credential. A JWT library can confirm that a college transcript is signed—but it has no idea what a college transcript is or how to interpret it. Thus, a JWT solution to credentials must add additional layers of semantic processing, or it must defer all

²⁴ <https://tools.ietf.org/html/rfc7519>

such work to proprietary, non-interoperable software or to human judgment.

Another drawback to using JWT for verifiable credentials is that it reveals everything in the signed document. There is no option for selective disclosure—the ability for the credential holder to only reveal certain claims on the credential, or to prove facts about those claims without revealing the claim value at all (such as proving “I am over 21” without revealing my birthdate).

JWTs were originally conceived as short-lived tokens used to authenticate or authorize moments after creation. Using them as long-lived credentials is possible, but without any predefined mechanism for credential revocation, it is not obvious how to test the validity of a JWT driver’s license months after issuance. It is possible to build such revocation mechanisms—perhaps with revocation lists, for example—but currently there is no standard for this, and thus no tooling to support it.

5.4.2 Blockcerts format

Blockcerts is an open-source proposed standard for machine-friendly credentials and the mechanisms that allow those credentials to be verified. It can use multiple blockchains—notably Bitcoin and Ethereum—as an anchor for credentials. Blockcerts was designed and is championed by Learning Machine.²⁵

Blockcerts are digitally signed JSON documents that encode the attributes describing the credential holder. They are issued to a payment address that must be controlled by the holder, and the payment address is embedded in the signed JSON. The holder can then prove that the credential belongs to them by demonstrating that they control the private key for the payment address. Work to adapt blockcerts to use DIDs is underway but has no announced schedule.

Blockcerts are typically issued in batches. Each blockcert is hashed, and hashes of all the certificates in the batch are then combined in a Merkle tree, with the root hash of the batch recorded on a blockchain as shown in figure 5.5 and explained in more detail in chapter 6 on cryptography.

DEFINITION A Merkle tree is a data structure that holds hashes of data, then hashes of those child hashes, then hashes of those parent hashes, and so forth. These hashes enable efficient proofs that data has not been modified.

²⁵ <https://www.learningmachine.com/>

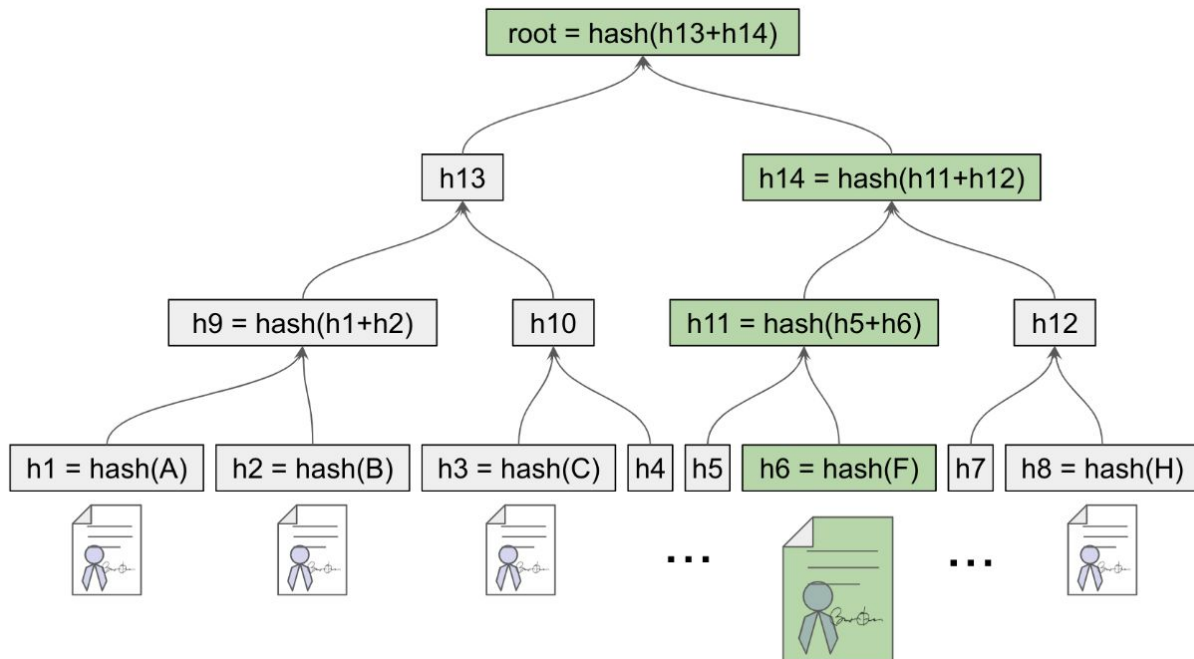


Figure 5.5: A Merkle Tree of Blockcert hashes, where the root hash is anchored to a public blockchain such as Bitcoin or Ethereum

Verifying a blockcert proves the following for a credential:

- It has not been modified since issuance.
- It is signed by the correct issuer.
- It has not been revoked.

Verification involves evaluating a Merkle proof, which means the hash of the credential is shown to map to a chain of hashes that ends with the hash stored on the ledger. The hash stored on the ledger must be a transaction recorded by the issuer (meaning digitally signed by a key that's also claimed by the issuer) at a URI that is also embedded in the credential. *Revocation* is tested by calling the issuer at another URI where revocation lists are stored. (A smart-contract-based revocation feature that improves privacy has been described but is not yet in production.)²⁶

²⁶ Santos, João and Kim Hamilton Duffy. Rebooting Web of Trust 5. "A Decentralized Approach to Blockcerts Credential Revocation". <http://bit.ly/2Ce9Bhq4>.

5.4.3 W3C verifiable credential formats

The need to establish a worldwide standard for interoperability of verifiable credentials was one of the primary reasons for the formation first of the Verifiable Claims Task Force and then of the Credentials Community Group at the W3C. The incubation work there resulted in the formation of the Verifiable Claims Working Group in 2017 and finally the publication of the Verifiable Credentials Data Model 1.0 as a full W3C Recommendation (official standard) in November 2019.²⁷

Verifiable credentials are so central to SSI infrastructure that they are covered in their own chapter (Chapter 7). For our purposes in this chapter, we can summarize that the Verifiable Credentials Data Model is an abstract data model that uses JavaScript Object Notation for Linked Data (JSON-LD—another W3C standard)²⁸ to describe credentials that can have different schemas and digital signature formats. Two general styles of W3C verifiable credential usage are contemplated: one that focuses on simple credential sharing, and one that uses specialized cryptographic signatures to facilitate zero-knowledge proofs.

In the **simple credential-sharing model**, credentials contain the DID of the holder (e.g., the person to whom they are issued). When the credential is presented, the holder reveals the whole credential, including this DID. The holder can then prove that the credential belongs to them because they possess the keys that control this DID. Revocation for such credentials uses *revocation lists*.

One challenge with the simple credential-sharing model is that it is inherently non-repudiable. Once shared, a credential can be reshared without the holder's permission to any number of parties. Perhaps legislation will clarify how consent for such resharing could be managed.

Another challenge is that revealing a credential in its entirety, including either the hash of the credential or the DID in the credential, provides an extremely easy and strong way to correlate the holder across all verifiers with whom the credential is shared. The extensive privacy risks of this approach are noted in the Verifiable Credentials Data Model specification.²⁹

The **zero-knowledge proof model** seeks to address these privacy concerns. One implementation is in Hyperledger Indy (popularized primarily by the Sovrin community). Indy credentials are not presented directly to a verifier. Instead, they are used to generate a proof of whatever data a verifier needs proved—no more, no less. A proof is essentially a derived credential that exactly matches the proof criteria. The proof is generated just in time, only when a verifier requests it. Furthermore, a unique proof is generated time proof is requested, and resharability of the proof can be controlled by the cryptography used by

²⁷ <https://www.w3.org/TR/vc-data-model/>

²⁸ <http://www.w3.org/TR/json-ld/>

²⁹ <https://www.w3.org/TR/vc-data-model/#privacy-considerations>

the holder. Because each presentation of a proof is unique, trivial correlation from using the same credential is avoided.

For example, suppose Alice lives in a city that provides free electric car-charging services to its residents. Alice could prove her residency to the charging station day after day, for years, without the station being able to track Alice's movements through the city's charging stations. Each presentation of proof is different and discloses nothing correlatable.

It is important to note that this technique does not eliminate correlation in all cases. For example, if the verifier requires Alice to disclose her real name or mobile phone number in the proof, then it would become easy to correlate her. However Alice's agent can warn her if a proof is asking for highly correlatable personal data so she can make an informed decision.

Indy credentials also take a unique approach to revocation. Instead of using revocation lists, they leverage privacy-preserving cryptographic accumulators or Merkle proofs rooted on blockchain. (See Chapter 6 on Cryptography for more details.) This allows any verifier to check in real time for the validity of a credential, while not being able to look up a specific credential hash or identifier in a way that leads to correlation.

5.4.4 Credential exchange protocols

Regardless of the format of a verifiable credential, interoperability still depends on how that credential is exchanged. This involves any number of complex protocol questions. For example:

- Should potential verifiers reach out to holders proactively—or should they wait to challenge them when holders attempt to access protected resources?
- How do verifiers ask for claims that may be on different credentials?
- Can verifiers add filters or qualifications to their proof requests such as only accepting claims from specific issuers or only if a credential was issued before or after a specific date?
- Can verifiers contact a party other than the holder to get credential data?

This is where Layer 3 has a direct dependency on Layer 2. For example, if you think of credentials primarily as inert data, and you believe the best way to distribute identity data is through hubs, then a natural way to exchange that data might be to call a web API and ask that the data be served to you. The holder of a credential could leave the credential on the hub, with policy that tells the hub what criteria must be met before serving it to anyone who asks. This is the general paradigm embodied in the Credential Handler API (CHAPI) that the W3C CCG is incubating.³⁰ It is also the approach taken by credential-serving APIs being incubated for DIF identity hubs.³¹

³⁰ <https://w3c-ccg.github.io/credential-handler-api/>

³¹ <https://identity.foundation/hub-sdk-js/>

If, on the other hand, you place a higher emphasis on disintermediation and privacy, and you want proofs of credential data to be generated dynamically in the context of a specific connection, the a more natural answer would be to a peer-to-peer credential exchange protocol using agents and DIDComm. This has been the approach of the Hyperledger Indy, Aries, and Sovrin communities. The specific protocols are specified in Aries RFC 0036: Issue Credential Protocol 1.0,³² and Aries RFC 0037: Present Protocol Proof 1.0.³³ Multiple implementations of these protocols are already in production.

Again, this is the layer where, as of the writing of this book, there is the most divergence in architectures and protocol philosophies. Standardization efforts are active, but so far consensus is elusive. From the “glass half-full” perspective, however, this is also the area where real-world adoption drives convergence most quickly, as happened with earlier “protocol wars” such as the OSI-vs-TCP/IP struggle that eventually birthed the Internet.

5.5 Layer 4: Governance frameworks

Layer 4 is not just the top of the stack; it is also where the emphasis moves almost completely from machines and technology to humans and *policy*. When we introduced governance frameworks in chapter 2, we used the diagram shown in figure 5.6 because it shows how directly governance frameworks build on verifiable credentials.

³² <https://github.com/hyperledger/aries-rfcs/tree/master/features/0036-issue-credential>

³³ <https://github.com/hyperledger/aries-rfcs/tree/master/features/0037-present-proof>

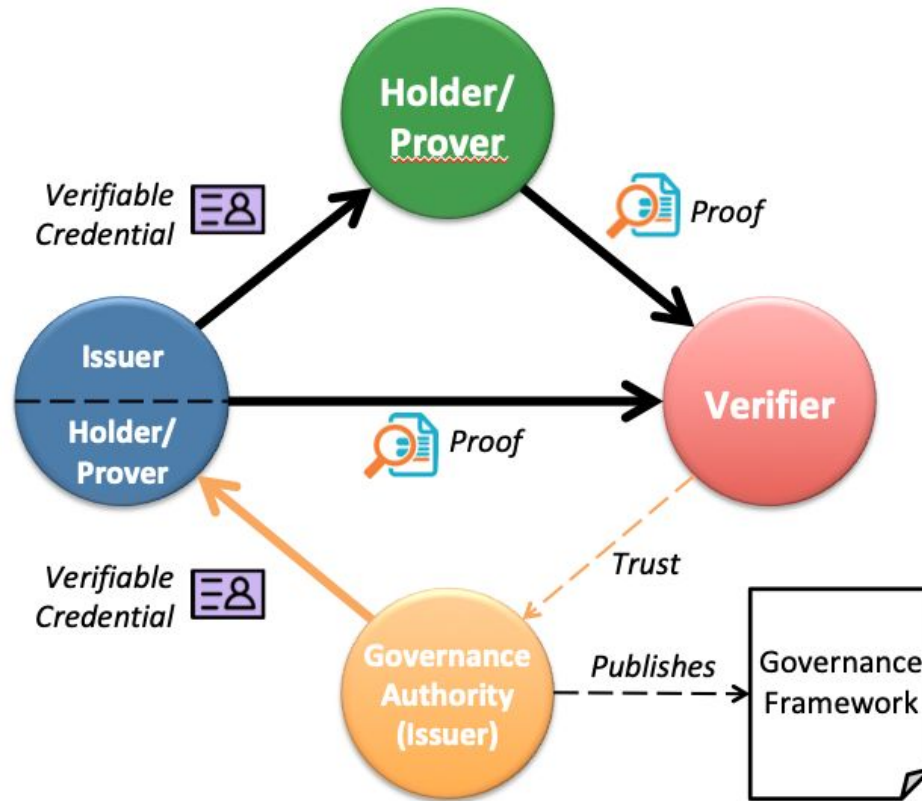


Figure 5.6: Governance frameworks are another “trust triangle” that solve specification, adoption, and scalability challenges of verifiable credentials

Governance frameworks enable verifiers to answer an entirely different—but equally relevant—set of questions about verifiable credentials. For example:

- How do I know this driving license was issued by a real government agency?
- How do I know the claims on this credit card came from a real bank or credit union?
- How do I know this diploma was issued by a Canadian university?
- How do I discover what types of credentials are issued by high schools in Austria?
- Was KYC required by the Brazilian lender that issued this mortgage credential?
- Can I rely on a proof based on the birthdate in a Japanese health insurance credential?

Governance frameworks are such a core component of many SSI solutions that we devote an entire Part 2 chapter to them (chapter 10). Because they are also one of the newer components of the SSI stack, not many SSI-specific governance frameworks have been created yet. This is in contrast to **trust frameworks** designed for federated identity

systems, of which there are quite a few around the world.³⁴

However early work on SSI governance frameworks has convinced a group of SSI architects that these frameworks will in fact be essential to broad adoption of SSI because they are where the “rubber meets the road”. In other words, they are the bridge between the technical implementations of the SSI stack and the real world business, legal, and social requirements of SSI solutions. Furthermore, these architects realized governance frameworks actually apply at all four levels of the SSI stack. John Jordan, Executive Director of Emerging Digital Initiatives at the Province of British Columbia in Canada, christened this combination of technology and governance the **Trust over IP (ToIP) stack**. The full ToIP stack, shown in figure 5.7, is a “dual stack” where the left side represents technology layers and the right side represents governance layers. See Chapter 10 for more details.

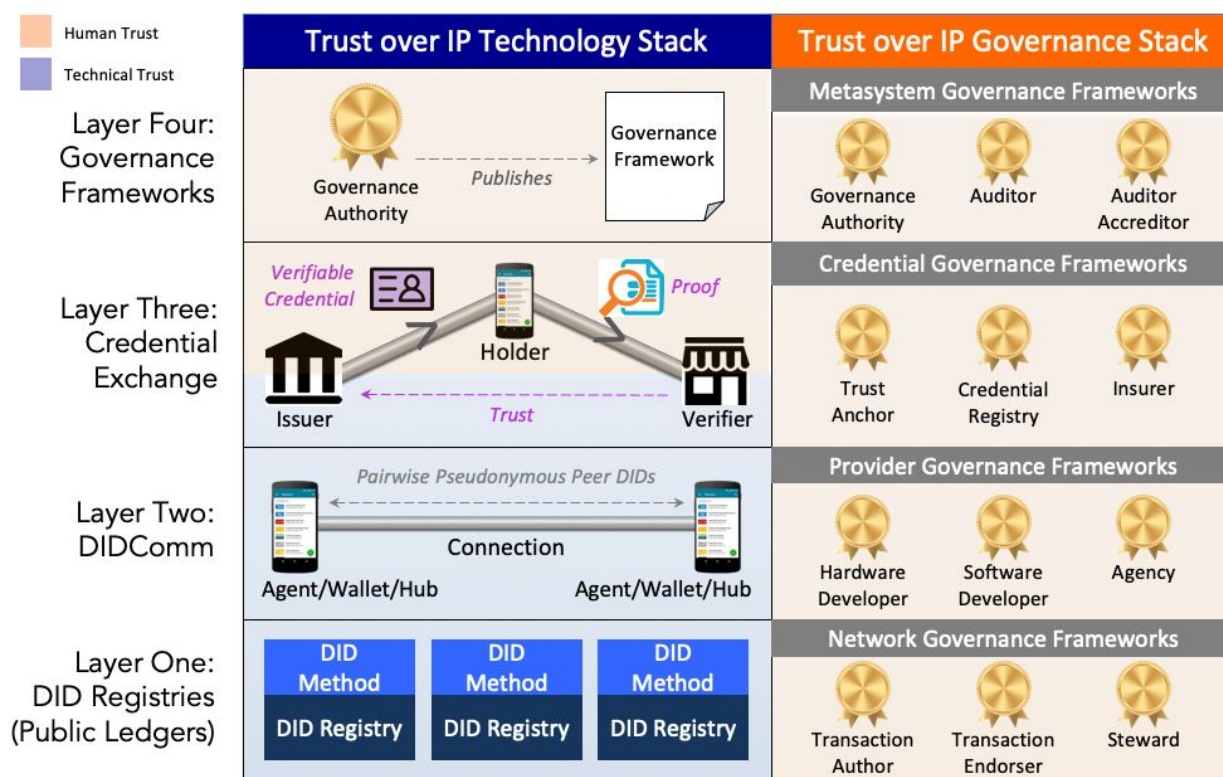


Figure 5.7: The Trust over IP stack illustrates that different types of governance frameworks apply to all four levels of the SSI stack

³⁴ See <https://openidentityexchange.org/>

Despite the fact that governance is where different trust networks and ecosystems within the SSI universe are likely to vary most significantly, the promise of the ToIP stack is that they can actually use an interoperable metamodel to define those governance frameworks. This can enable both humans and digital agents to much more easily make trust decisions across different trust communities, such as many of the questions listed at the start of this section.

As of the writing of this book, work is very actively going on by governance authorities of all kinds around the world to produce governance frameworks at all four levels. These are discussed in much greater detail in Chapter 10.

5.6 Summary

The field of SSI is young enough that the market is generating many variations in architecture. We have tried in this chapter to provide a big picture overview of where the market stands as of the writing of this book. We have been realistic that there are many areas of divergence. This is true at all four levels: different DID methods at Layer 1, different protocols and interfaces at Layer 2, different credential formats and protocols at Layer 3, and different approaches to governance at Layer 4.

However, at each level there is also the potential for convergence. The good news is that *this is what market forces want*. With automobiles in the early 1900s, market forces drove the industry to standardize on four wheels and internal combustion engines. With the Internet, they drove us to standardize on the TCP/IP stack. With the Web, they drove us to standardize on HTTP-based browsers and web servers.

We are hopeful the same market forces will drive convergence in SSI so that it, too, can rise to ubiquitous adoption. However, it also seems likely that more surprises await us. Digital identity architecture is likely to remain a fertile field for innovation for a long time to come.

6

Basic cryptography techniques for SSI

by Brent Zundel and Sajida Zouarhi

Cryptography is the fuel that powers all of SSI. The goal of this chapter is to make you conversant in the basic building blocks of cryptography: hash functions, encryption, digital signatures, verifiable data structures and proofs, as well as common patterns for how they are combined to create the cryptographic magic SSI delivers. Your guides will be two technical cryptographers with direct experience in the SSI space. Brent Zundel, Senior Cryptography Engineer at Evernym, and Sajida Zouarhi, engineer and researcher with ConsenSys. Brent also serves as co-chair of the W3C Decentralized Identifier Working Group that is producing the DID standard (the subject of our DID chapter).

To paraphrase the famous philosophical observation, “It’s turtles all the way down”,¹ some in the SSI community have said, “It’s cryptography all the way down”. Modern cryptography uses techniques from mathematics and computer science to secure and authenticate digital communications around the world. Encryption, digital signatures, and hashing are just some of the uses of cryptography that help make SSI possible—and also what make blockchains and distributed ledgers possible. Most SSI building blocks including verifiable credentials (VCs), decentralized identifiers (DIDs), decentralized key management and digital wallets rely on these cryptographic techniques.

This chapter is intended to give readers a basic understanding of the cryptographic techniques used in SSI infrastructure. While understanding these techniques is not essential to exploring SSI, it will make you more confident in some of the SSI “magic”.

Basic cryptographic building blocks

¹ https://en.wikipedia.org/wiki/Turtles_all_the_way_down

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

The cryptography underlying SSI infrastructure is constructed from five basic building blocks:

1. Hash functions
2. Encryption
3. Digital signatures
4. Verifiable data structures
5. Proofs

Hash Functions

What is a hash?

A **cryptographic hash** is like a unique digital fingerprint of a digital message or document. It is a fixed-length character sequence produced by running the input through a **hashing function**. Every input document produces a different output hash. If the same hashing function is applied to the same input data, the resulting hash will always be the same. A change of even a single bit of the input data will cause the resulting hash to be different. Table 6.1 shows some examples of hashes using the SHA-256² hash function:

Message (input)	Hash result in hexadecimal (output or digest)
"identity"	689f6a627384c7dcb2dcc1487e540223e77bdf9dcd0d8be8a32 6eda65b0ce9a4
"Self-sovereign identity"	d44aa82c3fbeb2325226755df6566851c959259d42d1259bebd cd4d59c44e201
"self-sovereign identity"	3b151979d1e61f1e390fe7533b057d13ba7b871b4ee9a2441e3 1b8da1b49b999

Table 6.1: Examples of hashes that use the SHA-256 hash function

The purpose of a hash is not to encode or hide a message, but to verify the integrity of a message. If a document has not been tampered with, its hash will stay the same. For example, hashes are used by software companies when they publish software programs with a corresponding hash. When users download the software they can verify the integrity of the files by comparing the hash of the software downloaded with the hash provided by the company. If the hashes are the same, the users know that the files have not been tampered with or corrupted—the file received by the user is an exact copy of the file published by the company.

²You can try this online using this website or other online resources: <https://www.xorbin.com/tools/sha256-hash-calculator>
©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Types of hash functions

Hash functions are an example of a **unidirectional function** (also called a **one-way function**).³ A unidirectional function is a mathematical function that provides a quick and efficient method to perform a calculation, with no known method to reverse the calculation in a reasonable amount of time.

There are many different hash functions, such as MD5 or SHA-256. Hash functions differentiate themselves by some basic characteristics:

- **Efficiency:** how fast can you generate a hash, with how little computational cost.
- **Resistance to preimage:** The input of a hash function is called the preimage. Resistance to preimage means that for a given hash, it is very difficult to discover the input. The output of a preimage-resistant hash function appears random and can not be predicted unless calculated. For example, if given the hash value `689f6a627384c7dcb2dcc1487e540223e77bdf9dcd0d8be8a326eda65b0ce9a4`, it would be computationally infeasible to determine that the input was the word "identity".
- **Resistance to second preimage or collision:** resistance to second preimage means for a given hash, there will only be one preimage. In other words, two different inputs will not produce the same hash. If two inputs produce the same hash, this is known as a collision. A hash function that has resistance to second preimage is also called "collision-resistant." For example, a collision resistant hash function that produces `689f6a627384c7dcb2dcc1487e540223e77bdf9dcd0d8be8a326eda65b0ce9a` as the hash of "identity" will not produce the same output for any other input.

NOTE (.calloutHead)There are many types of unidirectional functions. A well-known example is the product of two prime integers. Multiplying two large prime numbers is quick and efficient, but reversing that calculation and using the product to find the two input prime numbers is very difficult. This problem is called *integer factorization*.(.calloutStyle)

Some hash functions, such as MD5 or SHA-1, are no longer considered cryptographically secure. Cryptanalysts have found good attack vectors for these.

SHA-256 (SHA stands for Secure Hash Algorithm) belongs to the family of SHA-2 functions designed by the National Security Agency (NSA) and recognized by the National Institute of Standards and Technology (NIST) in the USA.

Hash function usage in SSI

Hash functions are used as a building block for verifiable data structures, and as part of digital signature algorithms, both of which enable necessary components for self-sovereign identity. Blockchains and distributed ledgers, verifiable credentials, and DIDs all rely on

³ https://en.wikipedia.org/wiki/One-way_function

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

cryptographically secure hash functions

Encryption

What is encryption?

Encryption is a way to hide the content of messages or documents so they can only be read by someone who knows a secret. Early examples of encryption consisted of ciphers and other methods of shuffling text. The secret messages usually relied on some secret method for their security. If an adversary knew the secret method, they could read the secret message. The secret message is also called the **ciphertext**.

Modern encryption methods no longer rely on keeping the encryption method secret. Instead, the encryption methods are public and well-studied, and the security of the encryption methods is based on the difficulty in solving some underlying problem. This difficulty is known as **computational hardness**. The secrecy of the ciphertext relies on secret keys. If the encryption methods are computationally hard, and the key is kept secret, the ciphertext will be unreadable for anyone who doesn't know the secret key.

Cryptography is divided into two families: **symmetric-key** and **asymmetric-key**.

In symmetric-key cryptography, the secret key used to encrypt messages is the same as the one that is used to decrypt ciphertexts.

In asymmetric-key cryptography, there are two keys, one for encrypting messages and the other for decrypting secret messages. The key used for encrypting messages is called the public key. The key used for decrypting ciphertexts is called the secret key or private key.

NOTE (.calloutHead)The intimate relationship of public and private keys to DIDs is explored in great detail in the DIDs chapter.(.calloutStyle)

Symmetric-key cryptography

In symmetric-key cryptography where the same key is used to encrypt and decrypt, one of the challenges is how to safely share the secret key with the recipient to enable them to decrypt the ciphertext. It follows that some of the most convenient uses of symmetric-key cryptography are when there is no need to share the secret key. For example, if you want to encrypt your hard drive you can encrypt and decrypt it with the same key.

One other advantage of symmetric-key cryptography is that it is more efficient than public-key cryptography; it provides the same levels of security, but using much smaller keys and much faster computations. One of the best-known algorithms for symmetric-key

cryptography is AES (Advanced Encryption Standard).⁴ AES uses secret keys of up to 256 bits⁵. This is 256 zeros and ones in a random sequence as in this example:

```
0111010100101011101011110100101011010010000101101010100100111010111110100010100
10010101110100100110100001101100100100101110011011111001110111001010111000
01010011011001101111101100110011101000011100000011010100011111000111101010
1000010010010011111
```

The size of secret keys is a critical element in determining the security of a cipher. The higher the number of possible keys, the more difficult it is for a computer to discover the valid key through a **brute force attack** (an attack based on trying one possible secret key at a time).

Asymmetric-key cryptography

Asymmetric-key cryptography, also known as **public-key cryptography**, uses a pair of keys, one public, and one secret. The keys are related mathematically and are always used in pairs. If one key is used to alter a message, only the other key can change the message back.

A secret key must be kept private, but the public key can be shared with the world. Anyone can use a public key to encrypt a message that only someone with the secret key will be able to decrypt.

In many public-key cryptosystems, you use the secret key to calculate the public key, but the secret key cannot be derived from the public key. The function to derive a public key is another type of unidirectional function.

The secret key may be nothing more than a large random number. This secret number is so big that it is nearly impossible to discover with a brute force attack. Some public-key cryptographic systems include algorithms such as RSA.

To encrypt a message using public-key cryptography, it is necessary to know the public key of the recipient. The public key is used to transform the message into a ciphertext. The ciphertext can only be decrypted back into the message using the associated secret key.

⁴ https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

⁵ This means 2^{256} possible combinations of zeros and ones. The number of possible 256 bit keys is 115,792,089,237,316,195,423,570,985,008,687,907,853,269,984,665,640,564,039,457,584,007,913,129,639,936

PUBLIC KEY CRYPTOGRAPHY

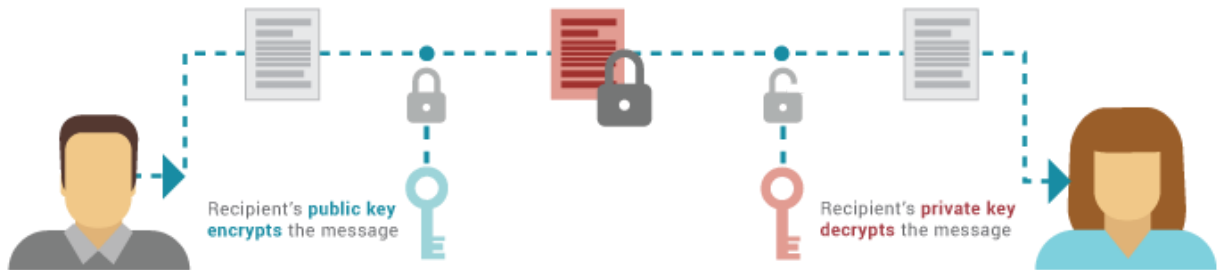


Figure xx.x: In this example Bob uses Alice's public key to encrypt a message that only Alice will be able to decrypt with his secret key.

DIDs make use of asymmetric-key cryptography. Following the same principle, a DID holder stores secret keys for DIDs in their digital wallet, while the public keys for DIDs may be publicly discoverable.

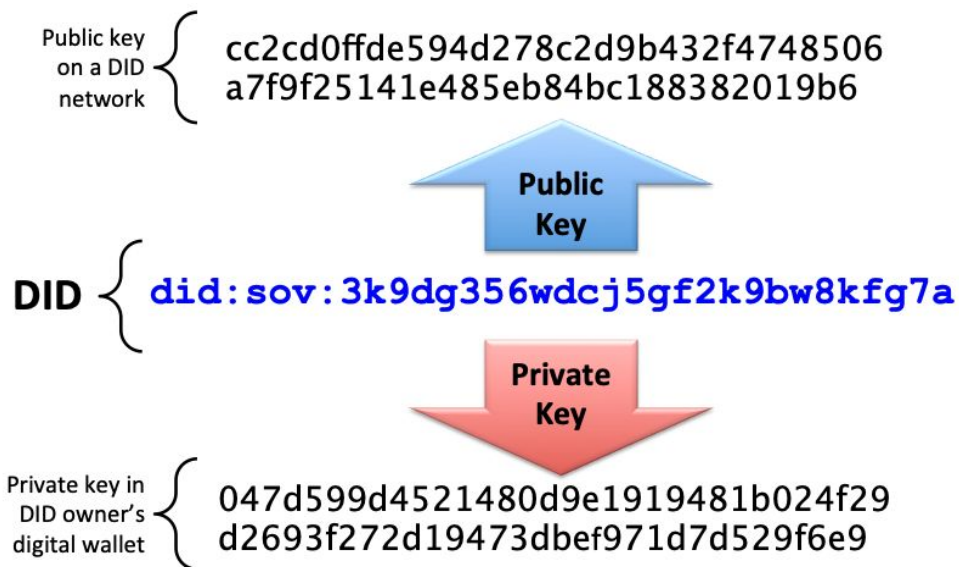


Figure 3.11: An example of a DID on the Sovrin DID network. A DID functions as the identifier of a public key on a blockchain or other decentralized network. In most cases it can also be used to locate an agent for interacting with the entity identified by the DID.

Digital signatures

Written or “wet ink” signatures are used every day to verify the authenticity of documents,

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

indicate the signer's consent, or both. Digital signatures use cryptographic functions to accomplish the same goals. Signing a message means transforming it in some verifiable way, using a secret key. The transformed message is called a **signature**. The message is then sent along with the signature to a recipient. The recipient can check the validity of a signature to verify that only the one who knew the secret key could have created the signature from the message.

Digital signatures rely on public-key cryptography as described above. A digital signature is created using the secret key of a key pair, and the signature can be verified using the associated public key. The larger the random number used to generate the secret key, the harder it is to discover with a brute force attack. Some public-key cryptographic systems include specific algorithms for digital signatures such as ECDSA (Elliptic Curve Digital Signature Algorithm).⁶

Digital signatures are used literally everywhere in SSI infrastructure—across all four layers of the SSI stack described in our architecture chapter. For example:

- In Layer 1, they are used for every transaction with a blockchain.
- In Layer 2, they are used to form DID-to-DID connections and to sign every DIDComm message.
- In Layer 3, they are used to sign every verifiable credential (some VCs actually contain digital signatures over each individual claim in the credential).
- In Layer 4, they are used to sign governance framework documents to ensure they are authentic as well as to sign VCs assigned roles within a governance framework.

Verifiable data structures

Cryptography can also be used to create data structures that have specific useful properties for data verification.

Merkle Trees

One of the most interesting cryptographic data structures was invented by Ralph Merkle in the early days of public key cryptography.⁷ Called a **Merkle tree**, it provides a very compact and computation-efficient way to verify the integrity of even very large data sets. Merkle trees are now a core component of many blockchain and decentralized computing technologies—starting with the Bitcoin protocol based on the blockchain architecture originally described by Satoshi Nakamoto in 2008.

The basic idea of a Merkle tree (also known as a *hash tree*) is that it can provide proof that a specific item of data—such as a blockchain transaction—exists somewhere within a very large amount of information (for example, the history of all bitcoin transactions) using a

⁶ https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

⁷ First described by Ralph Merkle, in his 1979 paper “A certified digital signature”, and subsequently patented by him. See https://en.wikipedia.org/wiki/Merkle_tree.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

mathematical process that results in a single hash called the **Merkle root**.

Building a Merkle tree

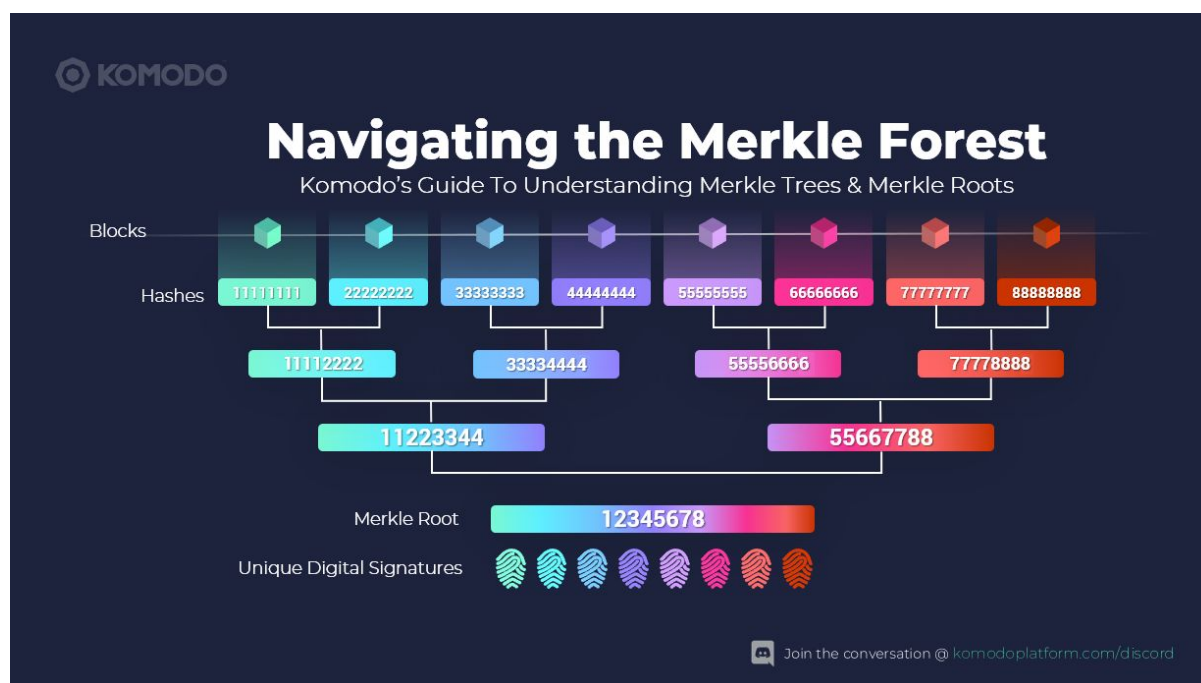
To show how a Merkle tree is built, let's start with a formal definition:

A Merkle tree is a tamper-resistant data structure that allows a large amount of data to be compressed into a single hash and can be queried for the presence of specific elements in the data with a proof constructed in logarithmic space.⁸

This means that even though a Merkle tree may contain one million pieces of data, proving that any single piece of data is in the tree only takes around 20 calculations.

To build a Merkle tree, a computer will gather the hashes of all the inputs and then group them in pairs—an operation called a *concatenation*.

For example, if you start with 20 inputs, after the first round of concatenation you will have 10 hashes. Once you repeat the operation you will have 5 hashes, then 3, then 2, then 1.⁹ This final hash is called the Merkle root.



⁸ <http://hackage.haskell.org/package/merkle-tree>

⁹ Since 5 is an odd number, the 5th element will be duplicated (we now have 6 elements) and grouped with itself, hence the resulting situation of 3 hashes in the next round. The same process repeats with the 3rd element in this round. It will be duplicated (we now have 4 elements) and grouped with itself to result in 2 hashes in the next round.

Figure xx.x: In a Bitcoin Merkle tree, different transactions are hashed until a unique hash is created for all the transactions included in the tree.

The goal of a structure like this is not to store or transfer the exhaustive set of input data, but to save a proof of their existence in a format small enough that it can be easily stored and exchanged between computers.

Searching a Merkle tree

A computer (such as a *node* on a blockchain like Bitcoin, which has a local copy of the Merkle tree of all the transactions in a block) can quickly verify that a specific piece of information (for example, a bitcoin transaction) exists within the Merkle tree. To do so, the computer only needs the following **proof information**:

1. The *leaf* hash, which is the hash of the piece of information (such as a transaction hash),
2. The *Merkle root* hash,
3. The hashes along the root *path*. The root path is the path from leaf to the root—it consists of the sibling hashes needed to compute the hashes on the path all the way from the leaf to the Merkle root.

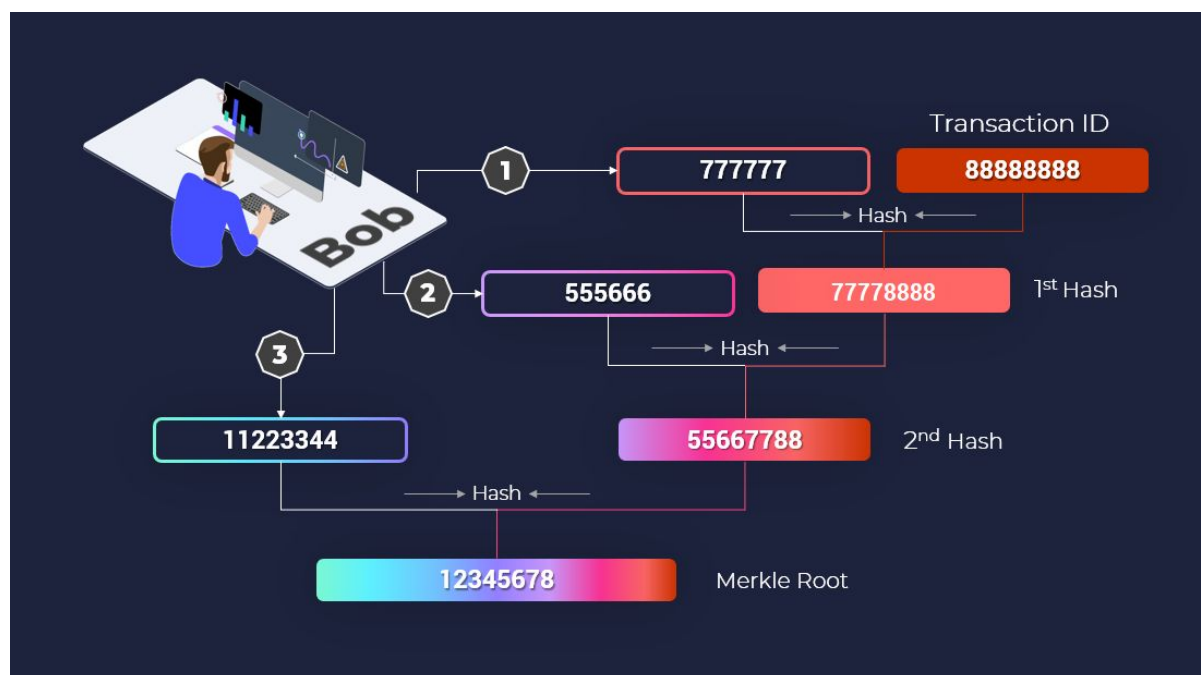


Figure xx.x: Verifying inclusion of a transaction (i.e. Merkle proof) means successfully recomputing the hashes of the branch from the top to the bottom using the *proof information* until the Merkle root. If the resulting output does not match the Merkle root, the transaction does not exist in the tree.

Instead of having to verify the whole set of transactions, a computer can quickly and efficiently verify just that a certain hash exists in the Merkle tree. This can not only help

ensure data integrity, but it may be used in consensus algorithms to reveal if a computer is trying to lie about transactions to its peers.

Merkle trees are used mainly for:

- **Storage optimization** for a large volume of information, there's no need to store the complete data set.
- **Verification speed**, only a few data points are needed for verification, rather than the whole data set.

The key concept that makes the Merkle tree tamper-proof is the hash function's resistance to a *preimage attack*, in other words, an attack that tries to find the message that has a specific hash value.¹⁰ To put it simply, we can easily verify that a value belongs in a Merkle tree by computing the resulting hash for each level in the path all the way down to the Merkle root, however we can not find the inputs that were used to generate a specific hash. Hashing is a unidirectional function, which means it is computationally infeasible to retrieve, from a hash, the original inputs.

Patricia Tries

We have seen the value of Merkle tree for protocols (e.g. the Bitcoin protocol). Building on this concept, we now focus on another interesting cryptographic data structure called a **Patricia trie**.¹¹ Instead of hashes, these tries are constituted of regular alphanumeric strings. But first what is a trie and why do we need them in SSI?

A *radix tree* (or *compact prefix tree*) is a data structure that looks like a hierarchical tree structure with a root value and subtrees of children with a parent node, represented as a set of linked nodes.

The subtlety of radix tries is that the nodes don't store any information, they are only there to indicate a location in the trie where there is a split in the string of characters. Because it knows the key, an algorithm knows how to reassemble the previous prefixes (*edge labels*) leading to that position in the trie. Figure xx.x is an example using a set of dictionary words.

¹⁰ https://en.wikipedia.org/wiki/Preimage_attack

¹¹ Tries are digital trees that get their name from the term "reTRIEval."

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

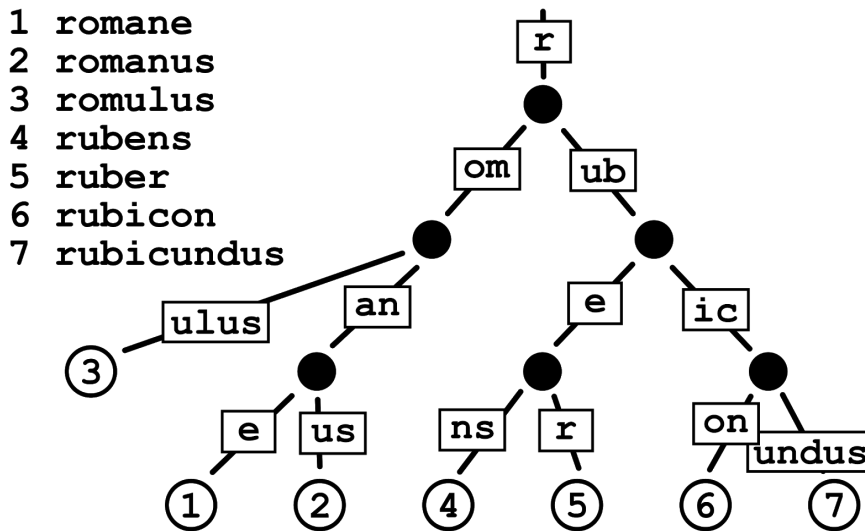


Figure xx.X In a radix tree, nodes don't store any information, they are indicators of a location in a trie. The algorithm goes through the trie and walks a path called the "key" to find the word. Here if we take the node in position 6 as the key and run through the trie as the algorithm would (from top to bottom) we reconstitute the word "rubicon".

The acronym PATRICIA stands for "Practical Algorithm To Retrieve Information Coded In Alphanumeric".¹² A PATRICIA trie is a variant of the trie we just saw, however instead of explicitly storing every bit of every key, the nodes store only the position of the first bit which differentiates two sub-trees. This makes a Patricia trie more compact than a standard binary trie and thus faster at finding common prefixes and lighter in terms of storage.

¹² Donald R. Morrison first described what he called "Patricia trees" in 1968.

Merkle-Patricia Trie - a hybrid approach

Merkle trees and Patricia tries can be used in combination to create data structures in different ways depending on the aspect a protocol needs to optimize, such as speed, memory-efficiency, or code simplicity. A particularly interesting combination example is the **MPT** (modified merkle patricia trie)¹³ found in the Ethereum protocol.¹⁴ MPT also forms the basis of the architecture for the Hyperledger Indy distributed ledger discussed in our architecture chapter—a code base designed explicitly for use in SSI infrastructure.

All MPT nodes have a hash value called a key. Key-values are paths on the MPT just like we saw with the Radix tree of figure 24.x with the “Rubicon” example.

MPT offers a **cryptographically authenticated data structure** that can be used to store key-value bindings in a fully deterministic way. This means that when you are provided with the same starting information, you will get the exact same trie with a $O(\log(n))$ efficiency.

Proofs

A **proof** is a way of using cryptography to demonstrate that a computational fact is true. For example, Patricia trees let you **prove** that a large set of data is correct without having to store the entire set. They are very useful for blockchains because instead of storing past transactions for 24 hours for example, you only need to store new “proof information” **on-chain**. The rest of the data can be securely stored **off-chain**. No one can tamper with the data without it being evident to the rest of the network.

Hybrid systems that combine on-chain and off-chain capabilities are used to overcome some blockchain limitations (e.g. scalability, privacy). This is particularly useful for applications such as SSI, where there is frequently the need to protect private information by keeping it off-chain. Hybrid systems enable software architects to have the best of both worlds.

A digital signature as discussed previously is also a form of proof. Anyone with knowledge of the public key but no knowledge of the secret key can prove that a specific signature was indeed generated by someone having prior knowledge of the corresponding secret key. The **prover**, the one who signs, is able to prove to the **verifier**, possession of knowledge on an information (the secret key) without revealing any information about the key, only the message and the signature.

¹³ <https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd>

¹⁴ <https://ethereum.github.io/yellowpaper/paper.pdf#appendix.D>

Zero-Knowledge Proofs

Now imagine if it were possible to also keep the signature and parts of the signed message secret. The only thing revealed would be the parts of the message that are disclosed and that the prover *knows* the signature. This is the goal of **zero-knowledge proof** cryptography.

A zero-knowledge proof needs to have the following three properties¹⁵:

1. **Completeness**: "If the statement is really true and both users follow the rules properly, then the verifier would be convinced without any artificial help."
2. **Soundness**: "In case of the statement being false, the verifier would not be convinced in any scenario." (The method is probabilistically checked to ensure that the probability of falsehood is equal to zero.)
3. **Zero-knowledge**: "The verifier in every case would not know any more information."

Properties (1) and (2) are needed for all interactive proof systems, such as digital signature proofs. Property (3) is what makes the proof "zero-knowledge".

As explained in our architecture chapter and in much more detail in the DID chapter, zero-knowledge digital signatures are used with some verifiable credential systems. Other zero-knowledge proofs that have emerged are the **zk-S*ARK** family of arithmetic-circuit based proof systems:

1. **zk-SNARK (optimized for privacy and consensus)**. This is a zero-knowledge succinct non-interactive argument of knowledge. It is used in blockchain protocols such as Zcash to hide the information relative to the sender and recipient of a transaction and the amount of the transaction itself, while at the same time allowing this transaction to be verified by the network and confirmed to the blockchain.
2. **zk-STARK (optimized for scalability and transparency)**. This form of ZKP, introduced by Eli Ben-Sasson, provides proofs that can be verified much faster by scaling exponentially relative to the data set they are representing.

ZKP applications for SSI

Zero-knowledge proofs can be very useful to prove information regarding someone's credentials without having to fully disclose identity information. Every situation where we need to prove that a person has the right to access a service without disclosing explicit information is a potential application for ZKP.

Zero-knowledge proving systems enable a number of beneficial properties.

Privacy and Personal Control

Concerns about privacy are driving significant changes in the way data are gathered, stored, and used. News coverage of continual data breaches has led to growing public unease about

¹⁵ S. Goldwasser, S. Micali, C. Rackoff <https://dl.acm.org/citation.cfm?id=63434>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

the lack of security around personal information, while political controversy has triggered alarm about the collection and sale of personal information without full and clear individual consent.

If personal data is less generally available, identity theft and fraud can be reduced. The best way to do this is if people—data subjects—are able to exercise personal control over what they share and how shared information is used once collected.

Signature Blinding

Zero-knowledge methods do not reveal the actual signature; instead, they only reveal a cryptographic proof of a valid signature. Only the holder of the signature has the information needed to present the credential to a verifier. This means that zero-knowledge methods automatically protect a holder from impersonation. Because the signature is not revealed, it also cannot be used as a unique identifier.

Selective Disclosure

Selective disclosure means that you don't have to reveal all of the attributes contained in a credential. If you only need to prove your name, you should not need to disclose your address or telephone number. Similarly, verifiers shouldn't have to collect more data than is necessary to complete a transaction. Selective disclosure reduces the liability of handling or holding data that they do not need.

Zero-knowledge methods allow a person to choose which attributes to reveal and which attributes to withhold on a case-by-case basis without involving the issuer. This enables two options: to reveal the attribute or to prove that you know the value of the attribute without revealing it. For example, a credential with attributes for name, birthdate, and address can be used in a presentation to reveal only your name.

Predicate Proofs

A predicate proof is a proof that answers a true-or-false question. For example, "Are you over the age of 18?" A car rental company may require proof that you are old enough to rent a car; they don't need to know your exact birthdate.

Using zero-knowledge methods, predicate proofs are generated by the user at the time of presentation without issuer involvement. For example, a credential with the attributes name, birthdate, and address can be used in a presentation to reveal your name and prove age-over-18, while withholding everything else. The same credential could then be used in a presentation to reveal your name and address, prove age-over-25, while withholding your birthdate.

Predicate proofs using zero-knowledge methods put the control of what data to reveal into the hands of the user. They also offer flexibility to users without the complexity of handling multiple credentials.

Revocation

Credentials may need to be marked as invalid. This may be because the credential data is no longer valid or the credential is being misused. The credential issuer can then revoke the credential. Verifiers must be able to determine a credential's current revocation status. Using zero-knowledge methods, such as cryptographic accumulators, the credential identifier can be checked against a set of revoked credential identifiers without revealing it. This reduces the ability of network monitors to correlate your credential presentations.

Anti-Correlation

Correlation is the ability to link data from multiple interactions to a single user. Correlation can be performed by a verifier, by issuers and verifiers working together, or a 3rd party observing the interactions on the network. Correlation is a way to collect data about a user without the user's consent or knowledge. It is also a way to deanonymize private transactions. For example, a person might use a credential to prove they are authorized to vote, then submit a secret ballot. If it is possible to correlate the person's credential with the secret ballot, thereby linking a specific vote to a specific voter, it would be detrimental to the democratic process and could enable retaliation.

One way to reduce correlation is through data minimization, by sharing only the information required to complete a transaction as discussed above in Selective Disclosure and Predicate Proofs.

A second way to reduce correlation is to make each interaction look unique. When interactions disclose unique identifiers, an observer can link multiple interactions to a single user. Zero-knowledge proofs provide unlinkability between interactions.

Zero-knowledge methods enable data minimization and allow users to have trusted interactions with verifiers without dependence on unique identifiers. Although correlation can never be eliminated completely, the goal of zero-knowledge methods is to reduce the probability of correlation and to put control over the level of correlation into the hands of the user.

Proofs and SSI

Like digital signatures (which are a form of proof), proofs are used everywhere in SSI. The most prominent usage, however, is in proving the validity of the claims in a verifiable credential. Some VC proofs are composed of many different smaller proofs. For example, if I wanted to prove the value of my first name based on a claim on a digital birth certificate signed by the government, I would also need to prove:

1. The birth certificate was issued *to* me.
2. The birth certificate was issued *about* me.
3. The government is actually the one who issued the birth certificate.
4. The birth certificate wasn't tampered with.

Even with these proofs, the verifier still needs to determine whether they trust that the government didn't make a mistake. In other words, even after I've proven the *integrity* of the credential (i.e., that it hasn't changed, is about me, and was actually issued by the government to me), the verifier still has no guarantees about the *veracity* of the credential (i.e., whether it is true), beyond the level of their trust in the issuer.¹⁶

Similarly, the information that is sent to any blockchain system, if not verified prior to its storage, could be false and at the same time used by the other elements running on that blockchain (e.g. Smart contracts) without raising any kind of error in the system. The vulnerability, for any blockchain-based systems and SSI in particular, is located at the entry points, where data comes into the system.

¹⁶ This is another place where governance frameworks come into play. See chapter 10.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

The cryptographic properties used in blockchain can't help the system know what is true and what is false, only what is stored and when. Cryptography is key to Self-sovereign Identity because of the properties it enables regarding sensitive data such as *integrity* or *traceability*. However, *veracity* isn't one of them.

It also enables *non-repudiation* which is important for DID and SSI. It means that when someone interacts with the system, for example to add new personal data, they can't afterwards claim that it wasn't them that made that operation but another entity. It also prevents anyone from adding or deleting information about someone else without her or his permission.

This illustrates some of the complexity behind proofs, as well as some of their limitations. Regardless of how cutting-edge or complicated a cryptographic proof may be, it can still only prove a computational fact about data. It cannot prove facts about the real world—only humans can do that.¹⁷

Conclusion

Blockchain technology was the dawn of SSI because it proved that cryptography could be deployed at scale in highly decentralized systems with extremely strong security properties. SSI takes the next step in harnessing that cryptography to prove digital facts about the identities and attributes of the participants in an SSI ecosystem. Every layer of SSI architecture described in our architecture chapter uses the cryptographic structures—hash functions, encryption, digital signatures, verifiable data structures, and proofs—described in this chapter.

In the next chapter we will dive into the core data structure enabled by all this cryptography—the “carrier of trust” across the trust triangle—verifiable credentials.

¹⁷ At least until we start to have very advanced artificial intelligence.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

Verifiable credentials

by David W. Chadwick and Daniel C. Burnett

Verifiable credentials (abbreviated as "VCs"), are the very heart of SSI architecture. In this chapter, you will learn how VCs evolved, what they look like as data structures, what different formats and digital signature options are supported, and how VCs have been standardized at World Wide Web Consortium (W3C). Your guides will be two of the primary authors of the W3C Verifiable Credentials Data Model 1.0 standard: David Chadwick, Professor of Information Systems Security at the University of Kent, and Dan Burnett, Blockchain Standards Architecture in the PegaSys group at ConsenSys. Daniel also served as co-chair of the W3C Verifiable Claims Working Group that created the VC standard.

We all use physical verifiable credentials (VCs) many times each day (even though you may not realize it). Examples include plastic credit cards, driving licenses, passports, membership cards, bus passes, and more. We can't live without them because they provide us with many benefits. In fact, we wouldn't possess them if they didn't provide us with some benefits. There would be no point.

Unfortunately, you can lose or misplace plastic cards and paper certificates or have them stolen. Worse still, they can be copied or cloned without you knowing about it. But most importantly, you can't use them online, not unless you type all of their details into a web form, which is both time-consuming, error-prone, and privacy-invasive. VCs, which are now a full World Wide Web Consortium (W3C) open standard, will allow issuers to convert the physical VCs they produce today into digital VCs, so that you can carry them around in your mobile phones, tablets, laptops, and other devices and use them online by simply pointing and clicking the screen.

Coupled with this advantageous ease of use, you also gain other benefits such as the

following:

- They cannot be copied or cloned.
- They are extremely hard to steal (an attacker would need to steal your electronic device as well as the method you use to authenticate to the device).
- They are more privacy protecting since they support *selective disclosure*.¹
- They are more secure because they support *least privileges*.¹
- They cost virtually nothing to issue.
- They are not bulky to carry around like plastic cards.
- You can delegate them to other people to use (providing that the issuer allows this).

Example Usages of VCs

Opening a bank account

Imagine that you want to open a bank account at BigBank. You visit the local branch of BigBank and the assistant asks you to provide two official forms of identification – the so-called "Know Your Customer or KYC" check. You have a passport and various utility bills that qualify, but unfortunately you have left them at home. Fortunately, the bank supports the use of W3C VCs, and you always carry your mobile phone around with you—on which you've stored several dozen VCs.

You select your government-issued passport VC, which confirms your nationality, name and age, and your national health VC, which confirms your current residential address. Both also contain a digital image of your face. Because the bank trusts the government and the national health service, it is willing to accept these VCs as proof of your name and address and that the person making the application is really you. This allows the bank to open your bank account and be confident of your identity.

Consequently, the bank issues you with a new VC that provides you with your bank account details and you add this to your set of VCs on your mobile phone. You can use this new VC in a variety of ways. You can add these bank account details to your online auction, payments, and shopping accounts, or give it to your employer for your monthly salary payments, or use it when opening a new high interest savings account with another bank.

Receiving a free local access pass

You're a pensioner (or a member of the FitSports sports club). You have a VC issued by the local council (or FitSports) that says you are entitled to free travel on buses in the local metropolitan area (or entry into all leisure centres operated by FitSports). Whenever you enter a local bus (or a FitSports venue) you simply place your mobile phone on the NFC

¹ When used in computer science, selective disclosure means disclosing only part of some verifiable data, whilst least privileges (also known as least authority) means only having permission to do the required tasks and nothing more.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

proximity reader, and it transfers a copy of your bus pass (or FitSports membership) VC to the operator, who then grants you free unrestricted access to the bus (or venue).

Periodically you need to renew this VC by applying online and paying the fee. The web site does not need a username or password to authenticate you. Instead, you simply present your existing VC together with your credit card VC (for payment). A replacement VC is issued with a new expiry date; it automatically replaces the old VC on your mobile phone.

Using an electronic prescription

You are ill at home in bed. The doctor visits you and diagnoses an infection, for which she prescribes penicillin. The prescription is written in the form of a VC, where you are the subject, the doctor is the issuer, and the contents are the drugs that have been prescribed to you.

Unfortunately, you can't go to the pharmacy yourself to pick up your prescription, so you ask your wife to go in your place. You transfer the prescription VC to her mobile phone, and in addition issue a relationship VC for your wife, with her as the subject, you as the issuer, and the attribute or property² says she is your wife.

Your wife goes to the pharmacy, and presents the prescription VC and relationship VC. The pharmacist checks who issued the prescription, and because he knows the doctor who signed it and has previously validated her medical practitioner VC, he's happy to give the prescribed drugs to your wife. The pharmacist records all this information in his audit log for future inspection.

These scenarios are just a few examples of how VCs can greatly simplify your life, both online and in the physical world. VCs were created for the following two purposes:

1. To provide a digital version of the credentials we carry in our digital wallets today.
2. To allow us to prove our identities "bottom-up"—with a set of claims about an identifier—rather than "top-down" by trying to define an "identity" and then attach information to it.

In the rest of this chapter, we'll explain these points and delve more into the technical details of VCs.

² Some identity management systems talk about a subject's attributes or identity attributes, others about a subject's personal information or PII. The verifiable credentials work talks about a subject's properties. We will use the term property throughout this document, on the understanding that it is synonymous with attribute or identity attribute or PII.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

The VC ecosystem

As you can see from the previous examples, there are a number of entities involved in the VC ecosystem (figure 7.1).

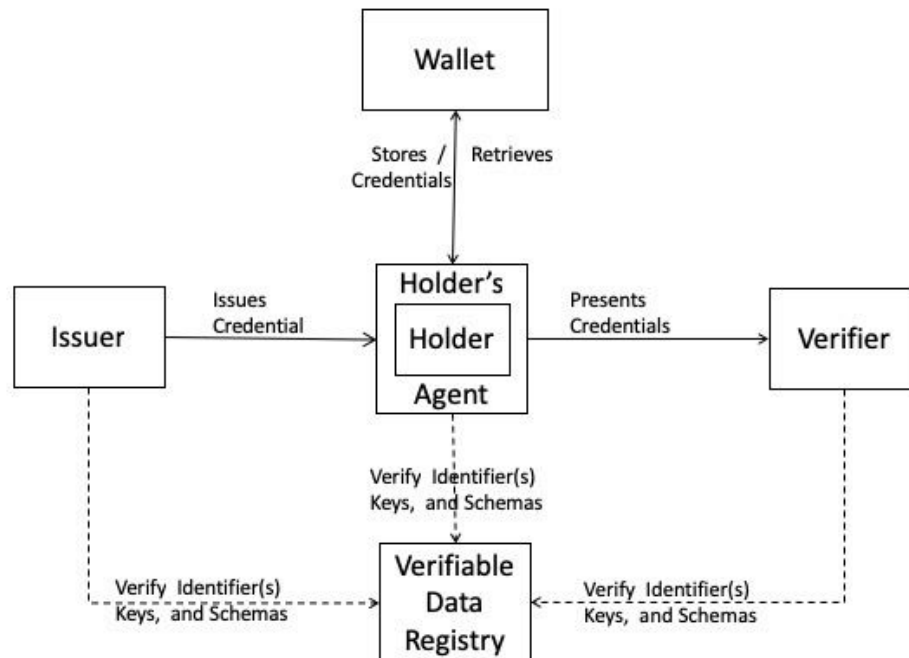


Figure 7.1: The overall architecture of VCs, in which the holder sits at the center—the essence of user-centric design

NOTE: To help familiarize the reader, in this section only the specific roles defined in the official W3C VC specification will appear in First Letter Capitals. In subsequent sections they will appear as standard generic nouns.

The components of the VC architecture are as follows:

- Issuer – the entity that issues VCs to users. In most cases the user is the Subject, but in some cases it may not be, for example, if the Subject is a pet cat, and the VC is a vaccination certificate, then the Issuer will issue the VC to the cat's owner.
- Subject – the entity whose properties are stored in the VC. A Subject can be anything with identity—a person, organization, man-made thing, natural thing, logical thing, and so on.
- Holder – the entity who is currently holding the VC and presents it to the Verifier. In most cases the Subject and the Holder are the same entity, but as you saw in the previous prescription and cat examples, this is not always the case.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

- Verifier – the entity who receives the VCs from the Holder, and provides benefits in return.
- Wallet – the entity that holds the VCs for the Holder. In many cases the Wallet will be integral to the Holder’s Agent, but the model allows for a remote wallet to exist, such as a cloud storage wallet.
- Holder’s Agent – this is the software that interacts with the VC ecosystem on behalf of the Holder. This could be an app that you load onto your mobile phone, or a program that you run on your laptop.
- Verifiable Data Registry – this is conceptually an Internet-accessible registry that holds all the essential data and metadata that enables the VC ecosystem to operate. Examples of the types of data and metadata that can be stored in this registry are the following:
 - The public keys of Issuers
 - The schema or ontology for all the properties that the VCs may contain, revocation lists of revoked VCs
 - The subject properties that Issuers say they are authoritative for (for example, a national government may list social security numbers and passport details; a university may list its degrees and transcripts)

In practice, we envisage that many different Verifiable Data Registry components will exist initially, because currently there are no standards for them. VCs have become popularized due to the potential for blockchains and distributed ledgers to serve as decentralized verifiable data registries. However blockchains are by no means the only option. Some verifiable data registries may be centralized, others widely distributed but based on different technologies, whereas yet others may be virtual and come pre-configured into holders’ software agents. This may change in the long term as standards evolve, but for now this entity is a placeholder to indicate that such a registry or registries are essential for any practical VC ecosystem to operate effectively and efficiently.

The VC ecosystem will consist of many Issuers, Verifiers, Holders and registry services working together in “ecosystems” or “trust networks”. A typical arrangement might operate as follows:

1. A Verifier defines its policies for accepting VCs from Holders for its supported services. One notable feature of the VC ecosystem is that a Verifier may offer a range of services, and each service may have a different policy. This helps to provide the *least-privileges* feature, because the Verifier needs to request only those subject properties that are necessary for the requested service. Each policy will say which Issuers the Verifier trusts to issue which VCs for this service; for example, a pizza web-site might have a policy that says to place an online order for a pizza, the user should present a credit card VC issued by Visa or Mastercard (for payment), and optionally either a preferred customer VC issued by itself or a student VC issued by the National Students Association (to claim a 10% discount on the order).
2. Either before or after or simultaneously with step 1, the various Issuers will issue their VCs to their Subjects, who will store them in their digital wallets. In certain cases the VC will be issued to a Holder who is not the Subject (e.g. pet vaccination

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

- VC), or to multiple Subjects e.g. a marriage certificate VC.
3. When allowed by Issuers, some Subjects may pass on their VCs to other Holders.
 4. The ultimate Holder will request a particular service from a Verifier.
 5. The Verifier returns its policy to the Holder's Agent, which checks if the Holder has the necessary set of VCs in its Wallet. If so, the policy can be fulfilled.
 6. The Holder presents the requested set of VCs to the Verifier, optionally inside a Verifiable Presentation (VP), which is a packaging mechanism to cryptographically prove that the Holder is sending the VCs, along with any conditions he or she may be placing on the Verifier.
 7. The Verifier verifies that:
 - a. the presented VCs and VP (if present) have authentic digital signatures.
 - b. the VCs match its policy.
 - c. the Holder is entitled to hold them.
 - d. the Verifier will conform to any conditions the Holder may have placed in the VP.
 8. If all is good, the Verifier performs the requested service for the Holder; if not, it returns an error message.

One caveat: as noted in the SSI Architecture chapter, the standards for Verifiers' policies and for the protocols between the various entities are still evolving in the marketplace. It is also worth noting that using VCs does not require the use of DIDs, just as DIDs do not require VCs in order to take advantage of using distributed ledger services for identification. However, combining VCs and DIDs together should bring many advantages to both technologies. VCs can leverage distributed ledgers for implementing components of the Verifiable Data Registry, and distributed ledgers can leverage VCs for gaining access to their services (as outlined previously).

The VC trust model

Federated identity management vs. VCs

For those of you who are familiar with today's federated identity management (FIM) systems (described in chapter 1), you will see that the VC architecture is quite different than FIM architecture.

DEFINITION FIM architecture is comprised of the user, the Identity Provider (IdP) and Service Provider (SP). At first sight this appears very similar to the holder, issuer and verifier roles in VC architecture. But the way the various parties interact is fundamentally different. In the FIM architecture the user first contacts the SP, is then redirected to the IdP where he/she logs in, and is then redirected back to the SP providing the latter with the user's identity attributes that the IdP is willing to release. In VC architecture there is none of this web-based redirection within a defined "federation"—the user as holder obtains VCs from issuers and uses them independently at any verifier that will accept them.

So you can see that the FIM architecture places the **IdP at the center of the ecosystem**, whereas the VC architecture places the **holder at the center of the ecosystem**. This is fundamental to understanding the VC philosophy, namely, *users are paramount, and they decide who to give their VCs to*. This compares drastically with the FIM philosophy, namely, *IdPs are paramount and they decide who can receive the user's identity attributes*.

One of the largest FIM infrastructures in the world is eduGain, comprising thousands of university IdPs from all over the world, along with numerous academic service providers. This community directly recognizes this key deficiency of FIM.

"Insufficient attribute release by IdPs is considered by user communities as the major problem today in the eduGAIN space."³

FIM architecture has this requirement because IdPs must trust SPs to maintain the privacy and confidentiality of the user's attributes that they send to the SP. In the VC ecosystem, this connection is broken. Issuers send the user's identity attributes to the user as properties in VCs, and the user decides what to do with them, just like they would do with a physical credential issued to the user's physical wallet. The issuer doesn't know which verifiers the user is giving his or her VCs to (unless the user or the verifier tell the issuer). So the issuer no longer needs to trust the verifier. This is a fundamental difference between the trust models of conventional FIM systems and the VC system.

³ EU AARC Project Deliverable DNA2.4 "Training Material Targeted at Identity Providers" 27 July 2016. Available from <https://aarc-project.eu/wp-content/uploads/2016/07/AARC-DNA2.4.pdf> [Accessed 8 Nov 2018]

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

Specific trust relationships in the VC trust model

The differences between the FIM trust model and the VC trust model are apparent from figure 7.2. Here we see the three major actors—issuers, holders, and verifiers—and their relationships to the three technical components: the holder’s agent, the wallet, and the verifiable data registry. Trust relationships are shown by directed arrows.

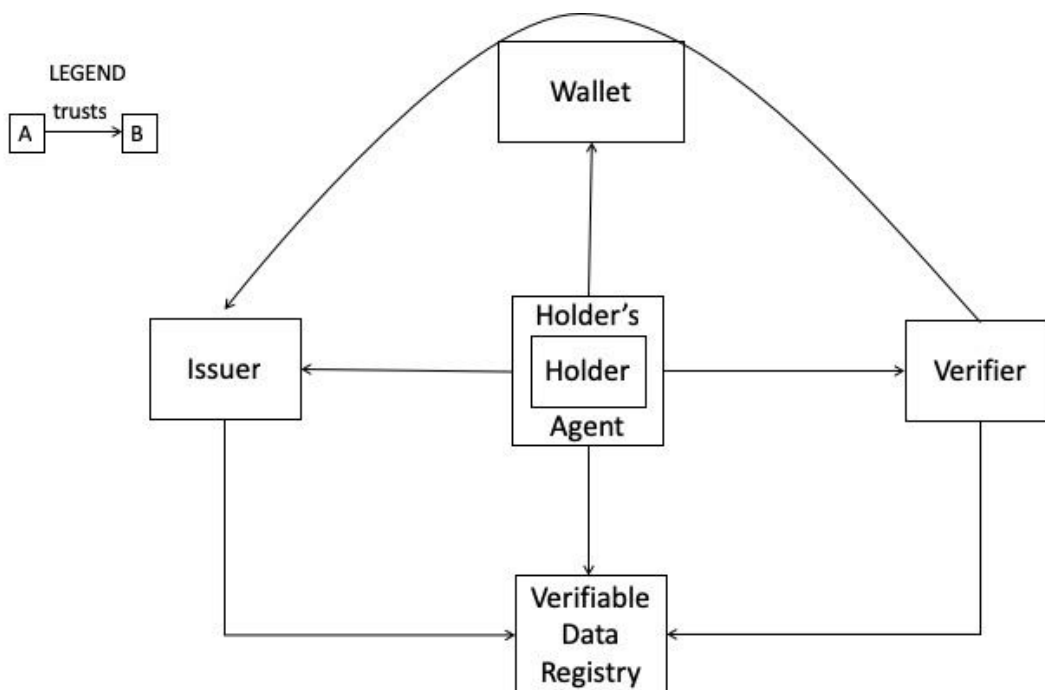


Figure 7.2: The VC trust model, where holders are at the center, and verifiers only need to trust issuers (and the verifiable data registry trusted by all parties)

Let's delve more deeply into each of the trust relationships depicted here.

- The verifier must trust the issuer to be authoritative for the subject's properties that are contained in a VC that it receives. For example, a verifier would normally trust a national government to be authoritative for a subject's nationality property, but would not necessarily trust the government to be authoritative for a university degree property. Conversely, a verifier would normally trust a university to be authoritative for a subject's degree qualification property, but not necessarily for the subject's golf club membership property. **Each verifier determines its own trust rules.** There is no compulsion on any verifier to trust any VC or any issuer. The verifier determines which VCs from which issuers to trust according to its own risk profile. However, to help VCs make their trust decisions, a Verifiable Data Registry (or a governance authority that publishes a governance framework—see the Governance Framework chapter) will often contain a list of known issuers and the

properties that they state they are authoritative for. Verifiers can then determine who and what to trust from this list.

- Consequently, all participants trust the Verifiable Data Registry to be tamper-evident and to be an accurate and up to date record of which data is controlled by which entities. Note that we say *tamper-evident* and not tamper-proof. No system on earth can stop determined attackers from tampering with it, but the Verifiable Data Registry should allow its users to detect whether it has been tampered with.
- It is obvious that both the subject/holder and the verifier must trust the issuer to issue true VCs. An issuer that lies cannot be trusted by anyone.
- Both the subject/holder and the verifier must trust the issuer to revoke VCs that have been compromised or are no longer true, in a timely manner. This policy is determined by the issuer. The best practice is for issuers to publish the revocation timeframes under which they operate, because this is fundamental to the risk-mitigation strategies of the verifiers. In certain circumstances a subject may know that an issuer holds out-of-date information and that the corresponding VC is incorrect, but the subject finds it very difficult to get the issuer to update its database and re-issue the VC. For this reason, there is a dispute procedure that is discussed further in the *Advanced features* section.
- Finally, the holder trusts his or her wallet to store VCs securely, not release them to anyone other than him or herself, and not allow them to be corrupted or lost while in its storage.

As you can see, this is a relatively simple trust model, and one which corresponds almost exactly to the trust model for the physical credentials we have been using for decades. This is one of the major strengths of the VC ecosystem for the simple reason that it *works the way trust in the real world does*—there is no artificial insertion of an Identity Provider into all of a user’s relationships.

Bottom-up trust

Credentialing systems often begin with a notion of first “securing one’s identity”, as in the PKI world. However, these top-down approaches to trust lead to centralization of information and control, dominance by small numbers of commercial entities, and loss of individual privacy. The bottom-up approach to trust taken with VCs seeks to avoid these pitfalls.

In fact one of the biggest challenges to adoption so far has been the simplicity of the VC trust model. The current dominance of the Internet giants who operate the largest FIM systems in the world (the “Login with _____” buttons you see on websites everywhere) has trained Internet users to believe they are the only sources of truth that verifiers will trust. But in the offline world, there are millions of issuers of credentials and millions of verifiers of those credentials, and the entire system is decentralized. Trust relationships are established peer-to-peer, “pairwise”, directly between holders and verifiers. This is sometimes called the *web of trust* model, and it is the bottoms-up trust model that VCs emulate. There are numerous examples of how this will work in the use-case part of this

book.

W3C and the VC standardization process

The work on what we now call Verifiable Credentials began as an offshoot of the Web Payments Interest Group at the World Wide Web Consortium (W3C). The W3C is the standards body that defines the HTML web programming language and the other key standards for the interoperability of the Web. The job of the Web Payments Interest Group was to standardize how to do payments on the web, in other words, how to pay directly from your web browser.

This required a way to authenticate individuals. So the Web Payments Interest Group started up the Verifiable Claims Task Force (VCTF)⁴ to explore whether the W3C could productively do work in this area. This new non-standards-track group consisted largely of members from both the Web Payments Interest Group and the W3C Credentials Community Group (another non-standards track group that is incubating all things related to digital credentials).

The VCTF developed an initial specification from which the group concluded that, indeed, W3C could productively help in this area. The VCTF wrapped up in May 2017 and transferred its specification to the newly-created W3C standards-track Verifiable Claims Working Group (VCWG). At first the specification was named the “Verifiable Claims Data Model and Syntaxes” specification, but after a year’s worth of work, the VCWG changed the name to “Verifiable Credentials Data Model 1.0”.

At this point you may be wondering why the group was named “Verifiable Claims” while the specification talks about “Verifiable Credentials”. The initial proponents of the working group very much believed—and still do—that the work is about creating credentials that are verifiable, where each credential can contain one or more claims from the same issuer. As the standards-track Working Group was being created, some parties at W3C objected to using the term “credentials” out of a concern that in the Web security community, the word “credentials” meant only username and password. Once the VCWG was underway, and more people joined the effort, that concern diminished—especially as it became clear that “credential” really was the appropriate word.

So, in summary, the W3C “Verifiable **Credentials** Data Model 1.0” specification⁵ (which we will refer to as the VC Data Model spec) was created in the W3C Verifiable **Claims** Working Group.

⁴ <https://w3c.github.io/vctf/>

⁵ <https://www.w3.org/TR/vc-data-model/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

The specification actually defines more than just a data model for VCs, however. It includes:

- A data model for Verifiable Credentials (VCs), the credentials that an issuer will provide to a holder
- A data model for Verifiable Presentations (VPs), the collection of credentials that a holder may present to a verifier
- A way to represent (or express) that data model using JSON Linked Data (JSON-LD) syntax
- A way to represent (or express) that data model using JSON Web Tokens (JWT) syntax.

We refer to the latter two items as *syntactic representations* of the data model. The difference between a data model and a syntactic representation (also sometimes referred to as a *serialization*) is that a data model describes relationships of one entity to another, for example: *I am a son to my father or that he is a father to me*. We could represent that with a diagram, but diagrams are not convenient for computers to work with. Alternatively, we could represent those relationships using a written format, or *syntax*, that is readable to computers, humans, or both. Both of the syntactic representations for VCs that we will cover are readable by both computers and humans.

Syntactic Representations

Although it might seem logical to describe the general data model before delving into how that data model is expressed in a concrete syntax, the data model itself is much easier to explain using concrete examples. So, in this section we will present concrete representations defined by the specification. These will support the examples we will show later on. For now, do not be concerned with the various properties or attributes you will see in these examples; definitions will come later in the data model section.

First, some background.

JSON

The two syntactic representations defined by the VC data model 1.0 specification are based on **JavaScript Object Notation (JSON)**, which is a simplification of the syntax used to represent collections of data items in the JavaScript programming language. This subsection will summarize JSON itself; the following subsections will then summarize the two syntactic representations defined in the VC data model specification that are based on JSON.

NOTE: Although these two syntactic representations, the **JSON-LD representation** and the **JWT representation**, are the only ones defined in the 1.0 specification, it is expected that others will be defined in the future.

A JavaScript object consists of an outer pair of curly braces that contain zero or more key-value pairs separated by commas, where each key value pair is a key string, a colon,

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

and a value. Between any two of those parts, white space is ignored. The key is referred to as a property of the object, and the value as the property's value. Here are some examples:

```
{height:5, width:7}

{"my height":75, "your height": 63}

{direction:"left",
  coordinates1: {up:7, down:2},
  coordinates2: {up:3, down:5},
  magnitude: -6.73986
}
```

Figure 7.3: Three examples of VC claims expressed in JSON

In the first example, the first key string is "height" and its value is "5" and the second key string is "width" and its value is "7".

Notice that by default a key is a sequence of alphanumeric characters (beginning with a letter) without spaces but that spaces can be introduced by quoting the string.

The value can be a string, an integer, a floating point value, or another object. Some unquoted strings have particular meanings when used in JavaScript. Examples include true, false, and null.

Beyond JSON - adding standardized properties

JSON itself allows for direct representation of any tree-structured data, but there is no standard set of properties for JSON objects. By standardizing a set of properties for VCs, it becomes possible to automate the creation and use of those credentials. The next two subsections describe how to represent VC data, with standardized properties, using two different syntactic representations: JSON-LD and JWT. The properties in each that are critical to the data model are mappable one-to-one between the two formats.

JSON-LD

JSON Linked Data, or JSON-LD, makes it easy to incorporate properties that are defined in structured templates called *schemas*. As an example, <https://schema.org/Person> defines a set of properties for a person. JSON-LD processors know how to automatically use these schemas to see what types of values the schema's properties expect.

In JSON-LD the desired schema locations are listed in the @context property's value. Then the type property specifies which particular schemas from those locations are being used. Those, in turn, define the allowed properties.

Here is an example where we are using properties from the VerifiableCredential schema at <https://www.w3.org/2018/credentials/v1> and the alumniOf property from the Person

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

schema at <https://schema.org>:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://schema.org",
  ],
  "id": "http://example.edu/credentials/58473",
  "type": ["VerifiableCredential", "Person"],
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": "Example University"
  },
  "proof": { ... }
}
```

Figure 7.4: Simple JSON-LD VC example showing @context attribute

This approach to naming is incredibly flexible, since anyone can create a new schema (called a “context”), post it at a stable URL on the Internet, and then reference its properties in the VC. Since the above example only cares about the `alumniOf` property, one could create a new, more specific, JSON-LD context called `AlumniCredential` that contains only one property, `alumniOf`. Assuming the new context was stored at <https://mysite.example.com/mycredentialschemas>, the example would then look like this:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://mysite.example.com/mycredentialschemas",
  ],
  "id": "http://example.edu/credentials/58473",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": "Example University"
  },
  "proof": { ... }
}
```

Figure 7.5: JSON-LD example showing custom @context schema

An important aspect of JSON-LD contexts is that the URLs and the schema definitions **MUST** be stable and unchanging. This allows for the context and schema definitions to be looked up only once (either automatically or manually) and then used from then on without any risk of them changing. It is very important to note that the JSON-LD syntax for VCs can be used *without any dynamic fetching of context information*. Essentially these contexts can be

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

thought of as defining separate name spaces so one person's definition of `alumniOf` does not have to be the same as someone else's. When the contexts are different, you know that they are not intended to be the same thing. This feature of JSON-LD is very helpful in VCs since the VC data model is purposefully an "open world" data model, where user-defined properties may be added at any time.⁶

As in the Verifiable Credentials Data Model specification, this chapter will use JSON-LD⁷ as the default syntax for most of the examples.

JWT

JSON Web Tokens, [RFC7519] pronounced either as "J W Ts" or as "Jots", are a previously standardized way to represent signable claims, or *attestations*, using JSON. The syntax is convenient for converting into binary representations that are precise in their use of white space and easily compacted to reduce storage and transmission costs. Given the breadth of the JWT-based ecosystem, it made sense to find a way to represent VCs and VPs in JWTs.

JWTs have three parts: a *header*, a *payload*, and an optional *signature* (of the payload). For our purposes, the *header* and its property values are not particularly relevant but are outlined in the VC Data Model specification. Since the JWT *payload* is essentially a restructuring of a JSON-LD VC, the easiest way to understand the JWT syntax for VCs is to first generate the JSON-LD syntax and then convert it. The most common properties are mapped as follows:

- replace the `id` property with `jti`
- replace the `issuer` property with `iss`
- replace the `issuanceDate` with `iat` and change the date format to a UNIX timestamp (NumericDate)
- replace the `expirationDate` with `exp` and change the date format to a UNIX timestamp (NumericDate)
- remove the `credentialSubject.id` property and create a `sub` property with the same value

A similar process is used to convert VPs to a JWT payload, with details explained in the specification. If a JSON Web Signature (JWS) is provided, in a VC it proves the issuer of the VC and in a VP it proves the holder of the VP. The VC `proof` property may still be provided as well if other proof information is used, for example zero-knowledge based approaches.

After the above conversions, any other remaining properties (including `proof`, if present) are moved into a new JSON object under the new JWT custom claim `vc` (or `vp` for a presentation).

⁶ https://en.wikipedia.org/wiki/Open-world_assumption

⁷ For those interested in learning more, the authors recommend <https://json-ld.org/primer/latest/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

As an example, let's look at a JSON-LD VC (minus the proof):

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://schema.org"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "Course"],
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "educationalCredentialAwarded":
      "Bachelor of Science in Mechanical Engineering"
  },
  "issuer": "did:example:abfe13f712120431c276e12ecab",
  "issuanceDate": "2019-03-09T13:25:51Z",
  "expirationDate": "2019-03-09T14:04:07Z"
}
```

Figure 7.6: Simple JSON-LD VC for an awarded educational degree

After converting to a JWT payload it would look something like this:

```
{
  "sub": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "jti": "http://example.edu/credentials/3732",
  "iss": "did:example:abfe13f712120431c276e12ecab",
  "iat": "1541493724",
  "exp": "1573029723",
  "nonce": "660!6345FSer",
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://schema.org"
    ],
    "type": ["VerifiableCredential", "Course"],
    "credentialSubject": {
      "educationalCredentialAwarded":
        "Bachelor of Science in Mechanical Engineering"
    }
  }
}
```

Figure 7.7: The same VC using JWT syntax (note that the nonce is a feature of JWTs not derived from the original JSON-LD VC)

The description above was designed to give experienced users of JWTs a rough idea of how this conversion process works and to let others know that the JWT syntax exists. The entire JWT-based syntactic framework is fairly extensive, and there are subtleties in how the pieces work together for VCs and VPs. These are best answered by reading the VC Data Model spec.

Basic VC Properties

Now that we've covered the syntactic representations, we can explain the properties defined by the data model. The most basic VC only needs to hold six pieces of information (encoded as JSON properties) as shown in Figure 7.8:

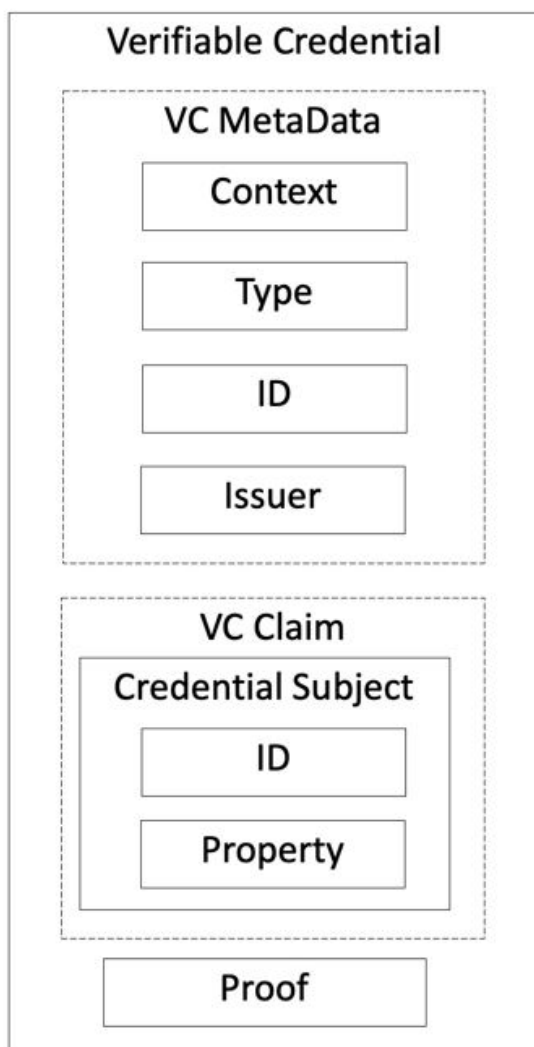


Figure 7.8: The structure of a basic VC, showing the metadata component, the claim component, and the proof component

The contents of each of these JSON properties are described below:

- **Context** – When people communicate, they need to know what language and vocabulary to use. While the default encoding language used for VCs is JSON-LD, that does not tell us what JSON properties a VC might contain. The `@context` property tells us which sets of vocabularies were used to construct this VC. Syntactically, the `@context` comprises a sequence of one or more Uniform Resource Identifiers (URIs). Ideally each URI should point to a machine-readable document containing a vocabulary that a verifier can automatically download and configure. Because many implementations may not be that sophisticated, the URI may alternately point to a human readable specification that will allow an administrator to configure the verifier software with the necessary vocabulary. Note that the `@context` on its own may provide more information (vocabulary) than a verifier wishes to use. Therefore, the VC also contains a `Type` property.
- **Type** – The type property contains a list of URIs that assert what type of VC this is. The first type must always be `"https://www.w3.org/2018/credentials/v1/VerifiableCredential"`, which can be abbreviated to `"VerifiableCredential"` through the use of the JSON-LD `@context` mechanism. Verifiers can read the list of types and quickly determine if they are able to understand and process this VC. If the VC is of a type the verifier does not recognise, it can decide to immediately reject it without further processing.
- **ID** – The id property is the unique identifier of this VC, created by the issuer. It comprises a single URI. This allows any entity to unambiguously reference this VC.
- **Issuer** – the issuer property uniquely identifies the issuer. It is a URI. This URI could point to a document that fully describes the issuer—for example a DID that point to a DID document—or it could contain the DNS name of the issuer and the Verifiable Data Registry could contain further details about the issuer, such as its X.509 public key certificate.
- **Credential Subject** – This contains the claims the issuer is making about the subject. It comprises the ID of the subject and the set of properties the issuer is asserting about the subject. In some cases the ID may be missing, for example, if this is a bearer VC (such as a concert ticket) that can belong to anyone. Since VCs are designed to preserve a subject's privacy, the ID is a pseudonym for the subject in the form of a URI. A subject could have countless pseudonyms and each VC could contain a different ID. In this way, without additional information, a verifier would not know that two VCs with different subject IDs belonged to the same individual.

Selective disclosure is another privacy feature that VCs can support. There are two recommended ways to achieve this: 1) by only including the absolute minimum number

of properties (preferably one) in a single VC (a so-called *atomic VC*), or 2) by using Zero Knowledge Proof VCs (see later). In the former case, an issuer, instead of issuing a complex VC containing multiple properties, might issue a set of atomic VCs. For example, a full driving license VC might contain the following 4 properties: name, address, date of birth and vehicle classes, where as the alternative would be four atomic VCs each containing one property and a link ID (to link them all together). The holder can then selectively disclose individual properties from his or her driving license.

- **Proof** – For a credential to be verifiable it needs a signature, referred to more generally in the VC Data Model spec as a *proof*. This cryptographically proves that the issuer issued this VC and that it has not been tampered with since issuance. Every VC must either contain the proof property or, if using JWT syntax, a JSON Web Signature. There is no single standard for the contents of the proof property, since several different types of proof are envisaged (see later in this chapter). If this property is used, the one common property that all proofs must contain is the `type` property, which asserts the type of the proof.

Other basic properties that are optional—but generally very useful:

- **Issuance Date** – This states the combined date and time, in ISO 8601 format, after which the VC is valid. Note that it is not necessarily the actual date the VC was issued, as an issuer may issue a VC before or after this date, but it is the date *before* which the VC is invalid.
- **Expiration Date** – This states the combined date and time, in ISO 8601 format, after which the VC is invalid.
- **Credential Status** – This provides the verifier with details of the current status of the VC, i.e. whether it has been revoked, suspended, superceded or otherwise changed since its issuance date. This is a particularly useful property for a long-lived VC, but less so for a short-lived VC, i.e. one where it is expected that the VC would not change its status before its expiration date.⁸ There is no standard format for the credential status property, but every status must contain an `id` and `type` property. The `id` property is the unique URL for this credential status instance—it is where the verifier can go to get the status information for this VC. The `type` property states the type of credential status, which in turn dictates what other properties the `status` property should contain.

⁸ Whether a VC is short-lived or long-lived depends upon the application using the VC and the risk profile of the verifier. For a stock market transaction long-lived might be greater than a second; for a national passport short-lived might be 24 hours.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

Figure 7.9 below shows a VC, encoded in JSON-LD, that contains all these basic properties.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://example.com/examples/v1"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": "https://example.edu/issuers/14",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "expirationDate": "2020-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science in Mechanical Engineering"
    }
  },
  "credentialStatus": {
    "id": "https://example.edu/status/24",
    "type": "CredentialStatusList2017"
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-06-18T21:19:10Z",
    "verificationMethod": "https://example.com/jdoe/keys/1",
    "nonce": "c0aelc8e-c7e7-469f-b252-86e6a0e7387e",
    "signatureValue": "BavEll0/I1zpYw8XNlbgVg/sCneO4Jugez8RwDg/+
      MCRVpjOboDoe4SxxKjkCOvKiCHGDvc4krqi6Zln0UfqzxGfmatCuFibcClwps
      PRdW+gGsutPTLzvUEMWmFhwYmfIFpbBu95t501+rSLHIEuuJM/+PXr9Cky6Ed
      +W3JT24="
  }
}
```

Figure 7.9: A JSON-LD encoded VC containing all the basic properties

Whilst the above describe the basic properties of a VC, it should also be noted that:

- 1) A VC can contain multiple claims, where each claim is about a different credential subject. For example, a marriage certificate VC could contain one claim about subject ID x, and another claim about subject ID y. The property of the first claim could be "married to subject ID y" and of the second claim "married to subject ID x".
- 2) A VC can contain multiple proofs, for example, a highly confidential or valuable VC might require two directors of the issuing company to sign it.
- 3) VCs can optionally be wrapped in a *verifiable presentation* by the holder.

Verifiable presentations

A verifiable presentation (VP) is one way that a holder may combine several VCs together to send to a verifier. It is very similar to a VC, in that it contains metadata about the presentation plus a proof signed by the holder. However, the contents are now a set of VCs rather than a set of claims.

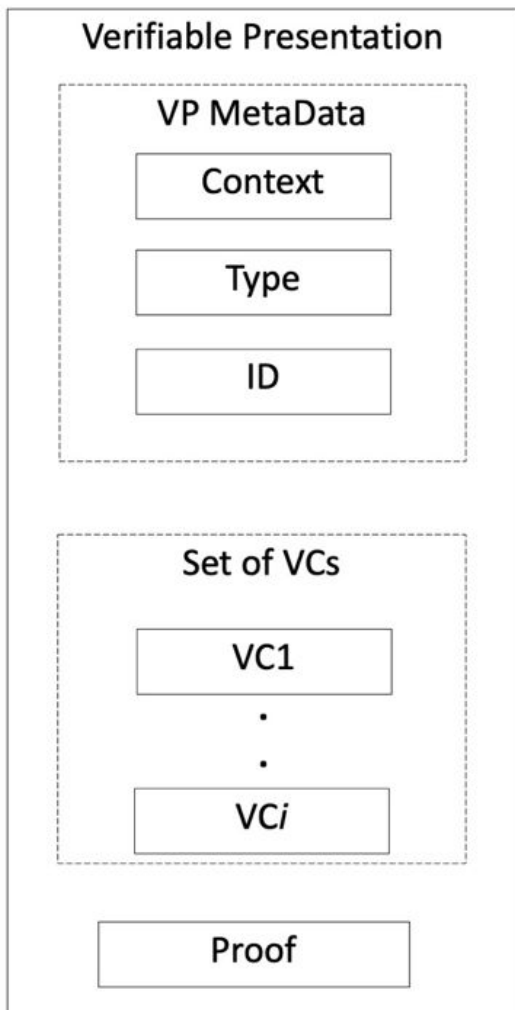


Figure 7.10: A basic VP that contains a group of VCs plus metadata about them

One notable difference between a VC and a VP is that the issuer property is missing. If present, this would have contained the ID of the holder, and indeed, a draft version of the specification did contain just that. When it was present, and if the ID of the VP issuer equaled the ID of the credential subject, then it was very easy for the verifier to determine that the holder is the subject of the encapsulated VC. However, this property was removed

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

in order for zero knowledge proof implementations not to be forced to reveal the ID of the holder (see ZKP section of this chapter for a fuller explanation). For non ZKP VCs, it is expected that the verifier will be able to determine the holder either from the protocol exchange prior to the VP being sent, or from the proof property of the VP (which can contain an identifier for the signer).

Another difference between a VP and a VC is that the `id` property is optional. It only needs to be present if the holder wants to uniquely refer to this VP on a subsequent occasion.

Similar to VCs, a VP may contain multiple proofs. For example, say a holder has several VCs each with a different credential subject ID that identifies him/her, implying a different asymmetric key pair for each VC, then the holder could send a VP to a verifier containing this set of VCs, and a set of proofs, each created by one of the key pairs.

Although usually clear from the context, it is common for users of VCs and VPs to refer to both generically as VCs unless they are specifically talking about a VP.

More Advanced VC Properties

Since VCs were developed using an open world model—meaning that anyone can add any property to a VC that is suitable for their application needs—nevertheless there are several properties that VCWG thought would be generally useful for a whole range of applications. These are described in this section. The following section will describe the rules the VCWG recommends application designers follow if they wish to extend their VCs and remain conformant to the W3C specification and interoperate with others. (Of course, anyone can extend a VC in any way they wish, but in that case they should not expect other implementations to interoperate with theirs.)

Refresh Service

VCs are designed to have a limited lifetime for several reasons. First, cryptography grows weaker over time, so enhanced proof mechanisms need to be put in place. Second, people's circumstances and credentials change with time. A good example is a VC based on age-related properties, or a student ID for a specific grade level.

The `refreshService` property allows an issuer to both control and provide details about how the current VC can be refreshed or updated. If the issuer wants the verifier to know how to refresh the VC, the issuer can insert the `refreshService` property into the VC. If the issuer only wants the holder to know how to refresh the VC, the `refreshService` property should *not* be inserted into the VC, but rather into the VP that encapsulates the VC that the issuer sends to the holder, since this VP is retained by the holder and is not forwarded to anyone else.

The `refreshService` property contains two mandatory properties:

- **ID** – This is the URL where the enquirer can obtain the refreshed VC
- **Type** – This says what type of refresh service this is, and controls what other properties the refresh service property should contain.

If a verifier wants to know if the presented VC contains the very latest information about the subject, then it could use the refresh service to find out. However, there are several caveats to this. First, this can breach the subject's privacy, because the issuer now learns that the subject's VC has been presented to a particular verifier. Second, the verifier will need to be authorized to access the issuer's refresh service (because the issuer is unlikely to issue a subject's VCs to anyone who asks for it). Yet if the verifier is already authorized, it will already know the details of the refresh service and won't need to consult the `refreshService` property. Consequently the practice of inserting the `refreshService` property into a VC is generally not recommended.

Conversely, if a subject/holder knows that a VC is about to expire, or that a property (claim) in his/her VC is out of date (and the issuer knows the new property value), then using the `refreshService` property in a VP is an easy for the holder to obtain an updated VC (and for the issuer to simultaneously revoke the old VC).

An example of a VC containing the "refreshService" property is given in Figure 7.12.

Disputes

Sometimes an issuer holds out of date information about a subject—but the subject is unable to compel the issuer to update its database and issue a revised VC. At other times a person could be the victim of identity theft and have an attacker masquerade as them by using falsely obtained VCs. In both these cases the VCs that exist are false and the rightful subject wants them to be revoked. If the issuer is slow to act in these circumstances, what should the rightful subject do? The answer is a **DisputeCredential**. This differs from a normal VC in that:

- The issuer is set to the URI of the rightful subject.
- It is signed by the rightful subject, and not the original issuer.
- The credential subject property ID contains the ID of the disputed VC.
- The credential subject property should also contain:
 - o A `currentStatus` property with the value set to `Disputed`.
 - o A `statusReason` property with the value set to the reason for the dispute. This is currently a free form string, rather than an encoded value, but standardized reason codes are anticipated in the future.
 - o If only part of the disputed VC is wrong, and some of its claims are correct

(i.e. it is not an atomic VC) then the `credentialSubject` property could also contain a reference to the disputed claim.

In the case of an out-of-date or incorrect VC that is held by the rightful subject, both the disputed VC and the `DisputeCredential` can be sent to the verifier. The verifier can then validate that the subject is the same in both VCs and determine the disputed property(ies).

In the case where a criminal is performing identity theft on a rightful subject, or if a denial-of-service attacker is trying to cause a verifier to doubt a valid VC, the subject in the disputed VC will be different from the subject who signed the `DisputeCredential`. In this case the verifier should ignore the `DisputeCredential` unless it has some out of band means of assessing its validity. Verifiers must establish their own policies for how to handle these cases.

An example of a `DisputeCredential` is shown in figure 7.11.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.com/credentials/123",
  "type": ["VerifiableCredential", "DisputeCredential"],
  "credentialSubject": {
    "id": "http://example.com/credentials/245",
    "currentStatus": "Disputed",
    "statusReason": "Address is out of date"
  },
  "issuer": "https://example.com/people#me",
  "issuanceDate": "2017-12-05T14:27:42Z",
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-06-17T10:03:48Z",
    "verificationMethod":
"did:example:ebfeb1f712ebc6f1c276e12ec21/keys/234",
    "nonce": "d61c4599-0cc2-4479-9efc-c63add3a43b2",
    "signatureValue": "pYw8XNi1..Cky6Ed="
  }
}
```

Figure 7.11: An example of a `DisputeCredential`

Terms of Use

Most physical VCs today are governed by terms of use. Some are stated on the plastic card itself, whilst others are published on a web site, with the plastic card containing the URL of

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

the web page. Examples of terms of use printed on physical cards include “Not Transferable” or “Only the authorised signatory may use this card.” Examples of referring to a web site include “Go to <URL> for full details of membership” or “Refer to <URL> for conditions of use”.

The standard way of adding terms of use to a VC (or a VP) is via the `termsOfUse` property. Terms specified by the issuer—that apply to both the holder and any verifier—will typically be inside the VC. Terms specified by the holder—which apply only to the verifier—will be inside the VP. Like all extensions to the basic VC (or VP), the `termsOfUse` property must contain its `type`, as this governs its contents. The `id` property is optional, but if it is present, it should point to a web page where the terms of use for this VC (or VP) can be obtained.

It is recommended that the terms of use should specify what actions the verifier, if it is to accept the VC (and or VP) from the holder, either:

1. Must perform (i.e. an Obligation).
2. Must not perform (i.e. a Prohibition).
3. May perform (i.e. a Permission).

More sophisticated terms of use may specify *when* these actions should take place e.g. “Notify the subject *upon accepting* this VC”, “delete this information *after 2 weeks*”, “archive this VC *for up to one year*”. and so on.

An example `termsOfUse` property is shown in Figure 7.12.

Evidence

The VC ecosystem is based on trust. However, trust is rarely binary (on or off). It is usually qualified. I may trust you with €50 but not with €5000. Similarly, a verifier may trust a VC issuer, but the level of trust that the verifier places in the issuer and in the issued VC may be qualified depending upon the procedures undertaken by the issuer, the strength of the cryptographic algorithms it used, the evidence it gathered, and what service the holder wishes to perform, and so on. The `evidence` property is designed for the issuer to help the verifier determine the level of confidence it can have in the claims inside the VC.

Authentication systems have had the concept of *Level of Assurance (LOA)*. As defined by the widely followed original NIST standard,⁹ this was a 4-level metric informing recipients about the level of confidence they can have in the strength of authentication of the remote party. This has since been superseded by more sophisticated LOA matrix from NIST¹⁰ because experience showed that a simple 1-to-4 le is insufficient to convey the inherent complexity of user authentication.

⁹ NIST Special Publication (SP) 800-63-2. “Electronic Authentication Guideline”. August 2013. Available from <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-63-2.pdf>

¹⁰ NIST Special Publication (SP) 800-63-3. “Digital Identity Guidelines”. June 2017. Available from <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-63-3.pdf>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

VCs are just as complex, if not more so, as authentication tokens. Consequently, rather than inserting a simple fixed metric into a VC, similar to an authentication LOA, the VCWG adopted an open-ended approach using the `evidence` property. This allows the issuer to insert whatever information it decides into a VC to assist the verifier in determining the level of confidence it can have in the VC's claims. It also provides future proofing: as VCs gain more traction and user experience, usage of the `evidence` property is bound to become more sophisticated.

As always, every `evidence` property must contain its `type`, since this determines what type of evidence it is and what other properties the evidence must contain. The `id` is an optional field that should point to where more information about this evidence instance can be found.

An example of the `evidence` property is given in figure 7.12.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://example.org/examples/v1"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": "https://example.edu/issuers/14",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science in Mechanical Engineering"
    }
  },
  "credentialSchema": {
    "id": "https://example.org/examples/degree.json",
    "type": "JsonSchemaValidator2018"
  },
  "termsOfUse": {
    "type": "IssuerPolicy",
    "id": "http://example.com/policies/credential/4",
    "profile": "http://example.com/profiles/credential",
    "prohibition": [{
      "assigner": "https://example.edu/issuers/14",
      "assignee": "AllVerifiers",
      "target": "http://example.edu/credentials/3732",
      "action": ["Archival"]
    }]
  },
  "evidence": [{
```

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

```

      "id":
"https://example.edu/evidence/f2aeec97-fc0d-42bf-8ca7-0548192d4231",
      "type": ["DocumentVerification"],
      "verifier": "https://example.edu/issuers/14",
      "evidenceDocument": "DriversLicense",
      "subjectPresence": "Physical",
      "documentPresence": "Physical"
    }, {
      "id":
"https://example.edu/evidence/f2aeec97-fc0d-42bf-8ca7-0548192dxyzab",
      "type": ["SupportingActivity"],
      "verifier": "https://example.edu/issuers/14",
      "evidenceDocument": "Fluid Dynamics Focus",
      "subjectPresence": "Digital",
      "documentPresence": "Digital"
    }
  ],
  "refreshService": {
    "id": "https://example.edu/refresh/3732",
    "type": "ManualRefreshService2018"
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-06-18T21:19:10Z",
    "verificationMethod": "https://example.com/jdoe/keys/1",
    "nonce": "c0a1c8e-c7e7-469f-b252-86e6a0e7387e",
    "signatureValue": "BavEll0/I1zpYw8XN1lbgVg/s...W3JT24 = "
  }
}

```

Figure 7.12: A more complex VC containing “termsOfUse”, “evidence”, “refreshService” and “credentialSchema” properties

When the holder is not the subject

In many VC uses, the subject and the holder will be the same entity. The verifier can determine this simply by ensuring that the `credentialSubject` ID inside the VC equates to the identity of the holder who signed the VP. But what about those cases where the holder and the subject are different? We have already cited several examples, such as when the subject is a pet, or an IoT device, or a relative of the holder. How is the verifier to know the difference between a rightful holder, who obtained the VC with the full permission of both the subject and the issuer,¹¹ and an attacker who stole the VC from the rightful holder? For example, say I steal your prescription VC, and take it to the pharmacy so that I can obtain your drugs. How is the pharmacist to know the difference between your friend, who you want to collect your prescription, and myself who claims to be your friend?

The VC Data Model spec contains four suggested ways to do this, however none of them were standardized in version 1.0 because it was deemed to be too early to determine which of these will become the preferred method(s).

1. The issuer issues the VC to the subject who passes it to a holder, then the **subject** issues a new, very similar VC to the holder. This new VC contains the same `credentialSubject` property value as the original VC, but now the holder is the subject, and the original subject is the issuer of the new VC. An example of this “passing on” is provided in Figure 7.13 below.
2. The issuer issues the VC to the subject who passes it to a holder, then the subject issues a **relationship VC** to the holder indicating the relationship between them. An example of a relationship VC is given in Figure 13 below.
3. The issuer directly issues the VC directly to the holder (who is not the subject) and then **the issuer also issues a relationship VC** to the holder indicating the relationship between the subject and the holder.
4. The issuer issues the VC to the subject with a **relationship claim** that asserts the relationship between the subject and a third party. Now the subject can pass the VC to the third party to become the holder, either immediately or at a later point in time. For example, a VC could be issued to a child that contains a relationship claim to the ID of the child’s parent. An example of this is given in Figure 7.13.

¹¹ For security purposes, both of these entities should give their permission for the VC to be transferred. The VCWG is standardizing a way for the issuer to mandate that the VC **must not** be transferred.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

```

{
  "id": "did:example:76e12ec21ebhyulf712ebc6f1z2",
  "type": ["VerifiablePresentation"],
  "credential": [{
    "id": "http://example.gov/credentials/3732",
    "type": ["VerifiableCredential",
"PrescriptionCredential"],
    "issuer": "https://example.edu",
    "issuanceDate": "2010-01-01",
    "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "prescription": {
        "drug1": "vall"
      }
    },
    "revocation": {
      "id": "http://example.gov/revocations/738",
      "type": "SimpleRevocationList2017"
    },
    "proof": {
      "type": "RsaSignature2018",
      "created": "2018-06-17T10:03:48Z",
      "verificationMethod":
"did:example:ebfeb1f712ebc6f1c276e12ec21/keys/234",
      "nonce": "d61c4599-0cc2-4479-9efc-c63add3a43b2",
      "signatureValue": "pky6Ed..CYw8XN1l="
    }
  },
  {
    "id": "https://example.com/VC/123456789",
    "type": ["VerifiableCredential",
"PrescriptionCredential"],
    "issuer": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "issuanceDate": "2010-01-03",
    "credentialSubject": {
      "id": "did:example:76e12ec21ebhyulf712ebc6f1z2",
      "prescription": {
        "drug1": "vall"
      }
    },
    "proof": {
      "type": "RsaSignature2018",
      "created": "2018-06-17T10:03:48Z",
      "verificationMethod":
"did:example:ebfeb1f712ebc6f1c276e12ec21/keys/234",

```

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

```

        "nonce": "d61c4599-0cc2-4479-9efc-c63add3a43b2",
        "signatureValue": "pYw8XNi1..Cky6Ed="
    }
}
],
"proof": {
    "type": "RsaSignature2018",
    "created": "2018-06-18T21:19:10Z",
    "verificationMethod":
"did:example:76e12ec21ebhyulf712ebc6f1z2/keys/2",
    "nonce": "c0a1c8e-c7e7-469f-b252-86e6a0e7387e",
    "signatureValue": "BavEl10/I1..W3JT24="
}
}

```

Figure 7.13: A Verifiable Presentation that contains (i) a VC passed to the holder by the original subject and (ii) a VC created by the original subject confirming that it has passed the VC on to the holder

```

{
    "id": "http://example.edu/credentials/3732",
    "type": ["VerifiableCredential", "RelationshipCredential"],
    "issuer": "https://example.edu/issuers/14",
    "issuanceDate": "2010-01-01T19:23:24Z",
    "credentialSubject": {
        "id": "did:example:ebfeb1c276e12ec211f712ebc6f",
        "child": {
            "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
            "type": "Child"
        }
    },
    "proof": {
        "type": "RsaSignature2018",
        "created": "2018-06-18T21:19:10Z",
        "verificationMethod": "did:example:76e12ec21...12ebc6f1z2/keys/2",
        "nonce": "c0a1c8e-c7e7-469f-b252-86ijh767387e",
        "signatureValue": "BavEl10/I1..W3JT24="
    }
}

```

Figure 7.14: A Relationship VC issued to a parent that identifies the child

```

{
    "id": "http://example.edu/credentials/3732",
    "type": ["VerifiableCredential", "AgeCredential",

```

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

```

"RelationshipCredential"],
  "issuer": "https://example.edu/issuers/14",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "ageUnder": 16,
    "parent": {
      "id": "did:example:ebfeb1c276e12ec211f712ebc6f",
      "type": "Mother"
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-06-18T21:19:10Z",
    "verificationMethod":
"did:example:76e12ec21ebhyulf712ebc6f1z2/keys/2",
    "nonce": "c0a1c8e-c7e7-469f-b252-86e6a0e7387e",
    "signatureValue": "BavEll0/I1..W3JT24="
  }
}

```

Figure 7.15: A child's VC containing identification of the mother

Extensibility and Schemas

VCs are built on an open world model, meaning that anyone is free to extend VCs in any way that they want to – “*permissionless innovation*” is the phrase used in the VC Data Model spec. Open extensibility maximizes the applicability of VCs, since all application developers are able to extend VCs to satisfy their application’s requirements. But if this extensibility is not properly controlled, it will lead to a lack of interoperability as software A will not be able to understand a VC transmitted by software B.

This is not a new problem. X.509 certificates had the same issue, and solved it by allowing the content of certificates to be extended—in any way that anyone wanted—by requiring every every extension to be labelled with a globally unique object identifier (OID).¹² The flaw in this solution was that there was no global object identifier registry, so it was impossible for software A to find out what an extension used by software B actually meant. This led to the IETF PKIX group standardizing dozens of extensions, and publishing their OIDs, so that everyone could know what they meant. But this is a cumbersome and time consuming mechanism. Something better was needed.

¹²OIDs form a hierarchical tree of numbers where the control of nodes is delegated downwards. You can think of this as the DNS in numerical form. See https://en.wikipedia.org/wiki/Object_identifier

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

VCS have a new way of solving the extensibility problem, by using the Internet itself to publish all the extensions that application developers invent, and mandating that VCs include the following in their encoding:

- The **@context** for this VC - as we saw earlier, this allows the issuer, holder and verifier to establish the right context, in terms of VC vocabularies (properties and alias names), for understanding the contents of the VC. One disadvantage of the `@context` property, for non JSON-LD users, is that contexts can be (and often are) nested, so several levels may need to be plumbed in order to discover the complete set of definitions;
- What **type** of VC this is - as we saw earlier, this allows a verifier to very quickly check if it supports this type of VC, and if not, to quickly reject it without the need for further processing;
- What **schemas** this VC uses - different types of VC will contain different properties, and both the `@context` and `credentialSchema` properties say where the definitions of these new properties can be found on the Internet. An example of the `credentialSchema` property is given in Figure 7.12 above. It is perhaps easier for non-JSON-LD users to understand and use than the `@context` property.

The preferred way of extending VCs is through the use of the JSON-LD syntax and its inbuilt `@context` extension mechanisms, since these already define how to extend objects in a way that maximises interoperability. However, the use of JSON-LD is not strictly necessary as pure JSON can be used providing that the `@context` feature of JSON-LD is supported as a JSON property. Remember that the `@context` property provides short form aliases for URIs so that they can be more easily referenced in the VC itself. It also says where the definitions of the core properties of VCs can be located on the Internet.

Diagrams of the VC ecosystem such as figure 7.1 often show a “verifiable data registry” where this extensibility data resides. In practice, there does not need to be a single place for this data; rather, it may be implemented as one or more decentralized blockchains, or as standard web pages accessible via HTTPS. However it is important that the system serving as the verifiable data registry be designed to persist at least as long as the expiration of any VC issued that references it.

Zero-knowledge proofs

As explained in our Cryptography chapter, the term *zero-knowledge proof*, or ZKP, refers to a class of cryptographic algorithms or protocols intended to allow for proof of the knowledge of a specific secret value, such as a password, without ever revealing the secret itself. Algorithms that fall under this category include *ZK-Snarks*, *ZK-Starks*, *Bullet proofs*, and *ring signatures*. There are many tutorials explaining what such proofs are and how they work,¹³ but for our purposes it is enough to understand that they can do one or more of the following:

1. Provide verification of claims in a VC without the issuer being involved or knowing who the verifier is (in other words, digital signatures)
2. Provide verification of claims in a VC while protecting the privacy of the holder
3. Allow for selective disclosure of a subset of the claims in a VC without revealing the contents, or even existence, of the other claims
4. Allow for derived claims to be presented to a verifier, e.g. over 18, rather than providing the full claim e.g. data of birth.

These properties are extremely helpful when VCs are used in strong privacy-preserving contexts or ecosystems. In particular, the approach taken in the VC Data Model spec focuses on preserving the privacy of the holder. In the many cases where the holder is also the subject, this preserves the privacy of the subject.

¹³ Most tutorials are either too high-level or too low-level, however a good starting point is https://en.wikipedia.org/wiki/Zero-knowledge_proof

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

<http://www.manning-sandbox.com/forum.jspa?forumID=861>

In the VC Data Model spec, the use of ZKPs involves three stages:

1. One or more *base* VCs are created by issuers using any proof (signature) approach. Additionally, the `credentialSchema` property in each contains a) a DID that will identify that base credential, and b) a `type` indicating the zero-knowledge proof system that will be used. One such system is the *Camenisch-Lysyanskaya Zero-Knowledge Proof* system,¹⁴ also known as *CL Signatures*.
2. For each VC the holder wishes to present to a verifier, a *derived* VC is created by the holder. In this derived VC, `credentialSchema` has the exact same contents here as the original base credential, which is how you know they are linked. The proof section, however, contains a CL proof which enables the holder to prove knowledge of a signature from the issuer over the credential *without revealing that signature*. This means only the holder can prove the signature, yet the proof does not reveal who the holder is to anyone else who might intercept the credential. Additionally, this derived credential can support *selective disclosure* by presenting only a subset of the claims in the base VC.
3. The derived VCs are placed inside a Verifiable Presentation issued by the holder to the verifier. The VP's `proof` section contains a CL proof with the following properties: the verifier can prove knowledge of a proof that all of the derived credentials were issued to that specific holder without revealing the latter proof. This means only the verifier—and no one else—can verify this proof (the verifier cannot share it with anyone else), and yet the VP's proof still does not reveal who the holder is to anyone else who might intercept the VP.

Many of the details here are glossed over in the VC Data Model spec itself, largely because of the huge variability in zero-knowledge proof systems. However, the data model and syntax of VCs and VPs should be sufficient to accommodate a variety of ZKP-based proof types.

¹⁴ <https://groups.csail.mit.edu/cis/pubs/lysyanskaya/cl02b.pdf>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

Protocols and deployments

While the VC Data Model 1.0 specification defines the format and contents of a VC, and a VP, that can be used to (i) transfer a set of VCs from the issuer to the holder and (ii) transfer a set of VCs from the holder to a verifier. However it does not define protocols for transferring and using VCs. That was deliberately put out of scope to keep the work of creating the 1.0 spec manageable. Obviously, these are needed before a VC identity ecosystem can become operational.

Several groups of researchers have been experimenting with different ways of transferring VCs, and we will highlight some of these in this section. Earlier we outlined the most likely steps involved in an operational VC ecosystem. Most of these steps still need to be standardised, for example:

- How does a verifier describe its policy (or requirements) for access to its resources, and transfer this to the holder?
- How does a holder request a VC from an issuer?
- How does an issuer inform the holder which VCs it is capable of issuing?

The very first demonstration of VCs was by Digital Bazaar¹⁵ in their Credential Handler API (CHAPI) for websites.¹⁶ It is a common problem on the web that login pages for websites have a list of potential ways to log in, for example using your Facebook account, Google account, LinkedIn account, etc. This has been referred to as the “Nascar Problem” because of the growing list of logos websites need to carry to accommodate all these identity providers. When using CHAPI, the web browser acts as mediator for the user, presenting them with choices in a standardized way to simplify the user experience. Different websites can be VC storage/holders (“wallets”), VC issuers, VC verifiers, or any combination of these.

The process begins when a user visits a credential storage site to get a new “wallet”. The site uses the CHAPI to set up handlers for both `CredentialStoreEvents` and `CredentialRequestEvents`. The browser agent saves the service address for this site and records that this site is a credential repository. The user then visits a credential issuing site.

When the site calls the credential management `store()` method, the browser presents the user with the credential to be stored and the list of wallets the user has into which the credential can be stored. When the user selects a wallet, the browser sends a `CredentialStoreEvent` to the wallet provider’s service address, at which point the credential is sent from the issuer to the wallet, where it is stored (and a “hint” for the credential is saved by the browser).

¹⁵ A video demo of this process is available at <https://www.youtube.com/watch?v=bm3XBPB4cFY>

¹⁶ <https://w3c-ccg.github.io/credential-handler-api/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

Similarly, when the user visits a credential requesting site (a verifier) and the site calls the credential management `get()` method with a requested type of credential, the browser presents the user with hints for the list of available credentials of the requested type. When the user selects the credential, the browser sends a `CredentialRequestEvent` to the wallet provider's service address, at which point the wallet sends the credential to the requester. Note that wallet and credential options are only presented to the user via the browser agent, and that credentials themselves are directly transported between wallet and issuer/verifier.

One of the earliest VC ecosystems was defined in 2015 by co-author David Chadwick. Built in 2016, it was presented at the European Identity Conference in 2017.¹⁷ It was based on enhancements to the (at that time) new FIDO specifications. In this ecosystem, the VCs are held on the user's mobile phone and are tied to the FIDO keys that are used for pairwise authentication with verifiers.

The system uses a hybrid model of centralized and decentralized identity models discussed in Chapter 1. The holders, issuers and verifiers are peers of the distributed model that communicate via connections established with FIDO keys. The blockchain component is replaced with the FIDO Alliances' centralized schema and key management services. When the user contacts an issuer, it says which identity attributes it is capable of issuing as VCs, and the user decides which one(s) he wants. This provides the issuer with consent, an important requirement of GDPR.

Each verifier is assumed to provide a range of services, and each service has its own authorization policy i.e. its VC requirements for access to the particular resource. After the user has browsed the verifiers services and decided which one(s) he wants to access, the verifier sends its specific policy to the user's device. The agent on the user's device searches the VCs that are obtainable from the various issuers in order to decide if the policy can be fulfilled. If it can, then VCs are requested from each issuer, to be bound to the FIDO public key for the verifier. These are then packaged together into a VP and signed by the user (using the private key paired to the verifier).

In this way the verifier receives a set of VCs, each bound to the same public key ID, which the user has proved ownership of by signing the VP. The system was tested with university students and the with a small sample of NHS patients, who unanimously found it easy to use and much preferable to usernames and passwords.

There are already a number of other deployments of VCs. For example, the government of the Province of British Columbia, Canada, has developed the Verifiable Organizations Network to serve as a public holder of business credentials.¹⁸ OrgBookBC¹⁹ contains a searchable registry of publicly-available licenses and permits granted to businesses in the province. Although the digital credentials will eventually be given directly to the subject organizations, publishing the credentials to the OrgBook makes the same information

¹⁷ <https://www.kuppingercole.com/speakers/405>

¹⁸ <https://vonx.io/>

¹⁹ <https://orgbook.gov.bc.ca/en/home>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

publicly available (and cryptographically-verifiable). This simplifies the process of identifying which businesses comply with relevant laws or, more critically, suggested but not mandated guidelines. It also serves as a convenient way for business owners, once they acquire a VC wallet, to obtain the relevant VCs for their business.

Security and Privacy Evaluation

The Security Considerations²⁰ and Privacy Considerations²¹ sections of the VC Data Model spec run tens of pages each. In this section we will just summarize the major points made in those sections.

- VCs are integrity protected end to end, i.e. from the issuer to the verifier through any intermediate storage wallet, because they have a proof property (or a JWT signature) inserted by the issuer that cryptographically provides tamper detection.
- VCs should be confidentially protected during transfer by only transferring them via encrypted communications links such as TLS.
- VCs have high availability if they are stored on the holder's device, since they do not require and communications with the issuer or any other third party in order to be transferred to the verifier.
- VCs facilitate the security property of *least privileges* by allowing the verifier to only request those subject properties (or attributes) that are essential for providing its service.
- VCs provide privacy protection through *selective disclosure*, by the issuer issuing either a ZKP VC or a set of atomic VCs, which allow the holder to only disclose those attributes that the Verifier needs, and nothing more.
- VCs provide privacy protection to subjects by identifying them via pseudonymous IDs rather than their names or other identities. If these IDs are pairwise between subjects and Verifiers then no globally unique correlating handles are created.
- VCs support flexible role-based²² and attribute-based²³ access controls, because the verifier only needs to specify which roles or attributes are needed to access its services, and not the identities of the users who can access its services.

Of course the ultimate security and privacy of any operational system depends upon the quality of the implementation and its correct use by the users. If a holder provides his or her unique email address and telephone number to a Verifier, then the use of pseudonymous IDs will not protect him/her from releasing globally unique correlating handles. Similarly if the user stores his/her VCs on an unprotected device, which is stolen, then the thief will be able to masquerade as the user.

²⁰ <https://www.w3.org/TR/vc-data-model/#security-considerations>

²¹ <https://www.w3.org/TR/vc-data-model/#privacy-considerations>

²² https://en.wikipedia.org/wiki/Role-based_access_control

²³ https://en.wikipedia.org/wiki/Attribute-based_access_control

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copy editors and proofreaders.

Hurdles to adoption

Physical VCs, such as plastic cards and passports, are indispensable in today's society. It is high time we moved these credentials into the electronic world so that we can carry them around on our mobile devices and have them with us all the time. But as we have seen, there are technical, security and privacy hurdles that need to be overcome before electronic plastic cards become ubiquitous. And this is before we even discuss the business models issuers will need to adapt or adopt to benefit from moving to this new mode.

As we explained in the first part of this chapter, today's federated identity management infrastructures give issuers (IdPs) great power because they are at the centre of the ecosystem. VCs turn this model on its head and place users at the centre. So issuers will need to see some financial benefit before they are willing to move. Many of the compelling benefits are discussed in chapter 4 on Major Features and Benefits of SSI, and more of these will be discussed in Part 4 of the book that covers SSI usage scenarios across major industries.

Summary

As they say in the movies, VCs are the "hero" of the SSI show. They are the most visible icon of SSI infrastructure, since they are what will sit directly in the digital wallets of individuals (and organizations) and be presented to verifiers digitally the same way we present physical credentials to verifiers today (like airport security agents to board a plane).

But in the move from physical credentials in paper and plastic (and even metal in some of the latest premium credit cards) to digital credentials, every last bit matters—literally. Since all the security and privacy properties of VCs depend on cryptography, the data structures and rules for composing and verifying VCs must be locked down to the last detail. That's what this chapter has walked through, from the supported syntaxes, to the required and optional properties, to the different signature mechanisms and extensibility options.

In our next chapter we will actually dive down one level deeper—into the special new form of cryptographically-verifiable identifier that was designed explicitly to support VCs and is now in the process of being standardized in its own W3C working group: Decentralized Identifiers.

Decentralized identifiers

by Drummond Reed and Markus Sabadello

Decentralized identifiers (abbreviated as "DIDs"), are the cryptographic counterpart to verifiable credentials (VCs) that together are the "twin pillars" of SSI standardization. In this chapter you will learn how DIDs evolved from the work started with VCs, how they are related to URLs and URNs, why a new type of cryptographically-verifiable identifier is needed for SSI, and how DIDs are being standardized at World Wide Web Consortium (W3C). Your guides will be two of the editors of the W3C Decentralized Identifier 1.0 specification: Markus Sabadello, Founder and CEO of Danube Tech, and Drummond Reed, Chief Trust Officer at Evernym.

At the most basic level, a **decentralized identifier (DID)** is simply a new type of globally unique identifier—not that different from the URLs you see in the address bar of your browser. But at a deeper level, DIDs are the atomic building block of a new layer of decentralized digital identity and public key infrastructure (PKI) for the Internet. This **decentralized public key infrastructure (DPKI)**¹ could eventually have as much impact on global cybersecurity and cyberprivacy as the development of the SSL/TLS protocol² for encrypted Web traffic (currently the largest PKI in the world).

This means you can understand DIDs at four progressively deeper levels (figure 8.1):

¹ <https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust/blob/master/final-documents/dpki.pdf>

² Secure Sockets Layer (SSL) and Transport Layer Security (TLS) For more details please access https://en.wikipedia.org/wiki/Transport_Layer_Security

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

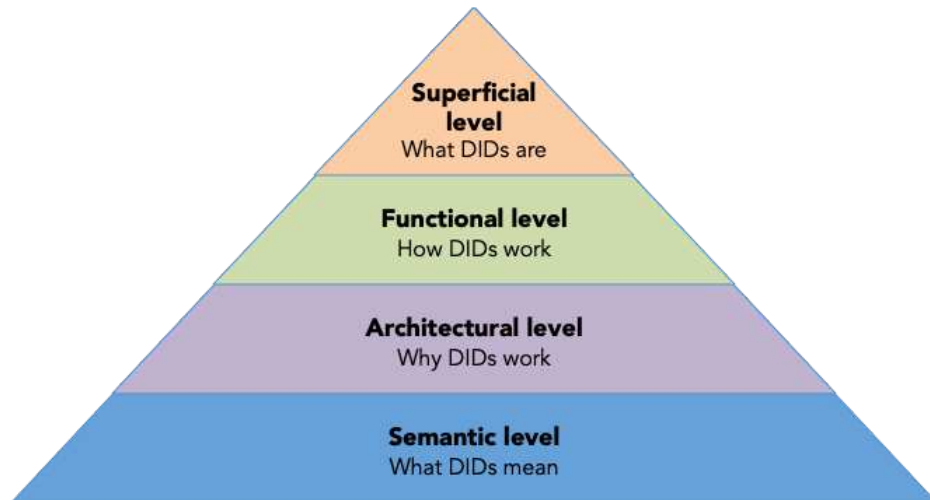


Figure 8.x: The progression of four levels at which one can understand DIDs—from a basic definition to a deep understanding how and why they work and what they mean for the future of the Internet and the Web

In this chapter we will travel all the way down through these four levels to provide a much deeper understanding of DIDs.

The superficial level: what is a DID?

To begin with, as defined by the W3C DID specification³ published by the W3C DID Working Group⁴, a DID is a new type of **identifier**: a string of characters that identifies a **resource**⁵. A resource is literally *anything that can be identified*, from a web page to a person to a planet. This string of characters looks very much like any other Web address, except it doesn't begin with "http:" or "https:", but with "did:".



Figure 8.x: The general format of a DID—the scheme name followed by a DID method name followed by the method-specific string, the syntax for which is defined by the DID method

URIs

In terms of technical standards, a DID is a type of URI (Uniform Resource Identifier). URIs are an IETF standard (RFC 3986)⁶ that was adopted by the W3C for identifying any type of resource on the World Wide Web. A URI is a string of characters in a specific format that makes the string globally unique, in other words, no other resource has that same identifier. This of course is very different than human names where many people can have exactly the same name.

URLs

A URL (Uniform Resource Locator) is a URI that can be used to **locate a representation** of that resource on the Web. A representation is anything that describes the resource. For example, for a website URL, the resource is a specific page on that site. But if the resource is a person, then he or she obviously can't be "on the Web" directly. So the representation needs to be something describing that person, e.g., a résumé, a blog, a LinkedIn profile, and so on.

Every single resource representation available on the Web has a URL—web page, file, image, video, database record, and so on. It's not "on the Web" if it doesn't have a URL.

³ <https://www.w3.org/TR/did-core/>

⁴ <https://www.w3.org/2019/did-wg/>

⁵ "Resource" is the term used by the World Wide Web Consortium (W3C) across all Web standards. The digital identity community generally uses the term "entity". For the purposes of this book, the two terms can be considered equivalent.

⁶ <https://tools.ietf.org/html/rfc3986>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

The addresses that appear in the address bar of your browser are generally URLs.

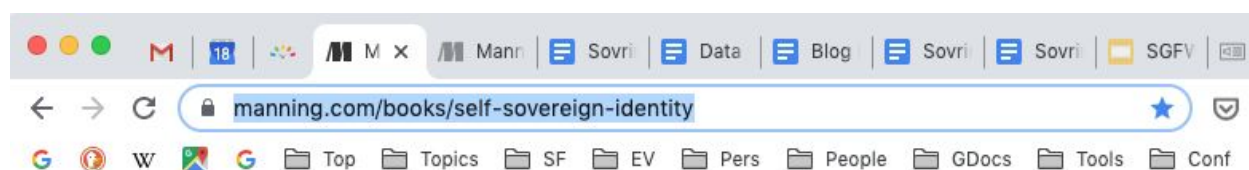


Figure 8.x: An example of a browser address bar displaying the URL for the web page about this book

Why do we need the distinction between URIs and URLs if everything on the Web has a URL? Because we also need to identify resources that are **not** on the Web. People, places, planets—all the things for which we had names long before the Internet even existed. All of these are resources that we often need to **refer to** on the Web without the resource actually being **represented on** the Web. A URI that is not a URL is often called an **abstract identifier**.

NOTE The question of whether a DID by itself serves as a URL—while sounding quite trivial—is actually quite deep. We will have to go all the way down to level four before we can answer it completely.

URNs

If URLs are the subclass of URIs that point to *the location of a representation of a resource* of the network—which can always change—then what about the subclass of URIs that *identify the abstract resource itself* and thus are designed to never change. It turns out there are many uses for this kind of **persistent identifier** (also called a **permanent identifier**). This is exactly what you want when you need to be able to refer to the resource:

- Independent of any particular representation.
- Independent of any particular location.
- Independent of any particular human language or name.
- In a way that will not change over time.

In Web architecture the subclass of URIs reserved for persistent identifiers are called *Uniform Resource Names* (URNs). They are defined by RFC 8141⁷, which goes into great detail about both the syntax and the policies necessary to manage namespaces for identifiers that are meant to never change. (Think about how complex it would get if every phone number, email address, and human name could be assigned only once and never reused for another person ever again for the rest of time.)

⁷ <https://tools.ietf.org/html/rfc8141>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Figure 8.x shows how URLs and URNs are both subclasses of URIs.

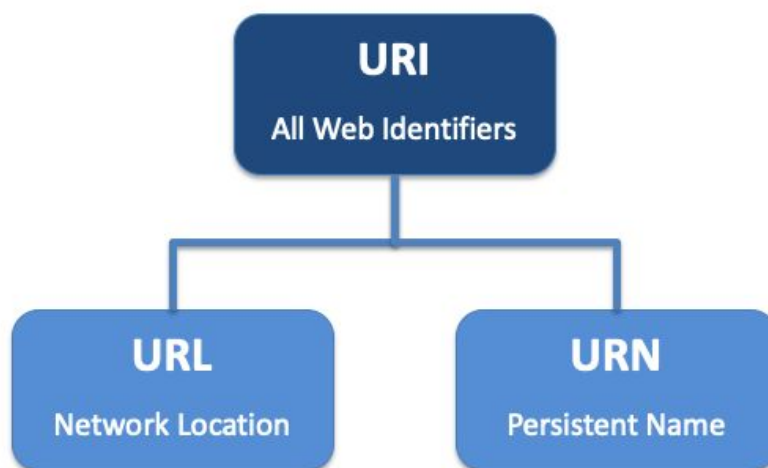


Figure 8.x: URLs and URNs are both subclasses of URIs—URLs always point to representations of resources on the Web, whereas URNs are persistent names for any resource, on or off the Web

DIDs

Having described URIs, URLs, and URNs, we can now be more precise about the definition of a DID: a DID is a URN that can be looked up (“resolved”) to get a standardized set of information (“metadata”) about the resource identified by the DID (as described in the next section of this chapter). If the identified resource has one or more representations on the Web, the metadata can include one or more of those URLs.

But that definition captures only two of the four properties of a DID—**persistence** and **resolvability**. It is the other two properties—**cryptographic verifiability** and **decentralization**—that most strongly distinguish a DID from other URIs or any other globally unique identifiers. At the first meeting of the W3C DID Working Group in September 2019 at Fukuoka, Japan, one presenter summarized the four properties this way:

The four core properties of a DID

1. A permanent (persistent) identifier

It never needs to change

2. A resolvable identifier

You can look it up to discover metadata

3. A cryptographically-verifiable identifier

You can prove control using cryptography

4. A decentralized identifier

No centralized registration authority is required

Figure 8.x: A summary of the four core properties of a DID presented at the first meeting of the W3C DID Working Group

What is special about the third and fourth properties is that they both depend on cryptography. In the first case, cryptographic verifiability, cryptography is used to generate the DID. Since the DID is now associated with exactly one public/private key pair, the controller of the private key can prove that he/she/it is also the controller of the DID. (See *The Architectural Level* section of this chapter for more details.)

In the second case, decentralized registration, cryptography is what eliminates the need for centralized registration authorities—the kind that are needed for almost every other globally unique identifier we use, from postal addresses to telephone numbers to domain names. It is the centralized registries run by these authorities that can determine if a particular identifier is unique—and allow it to be registered only if it is.

By contrast, cryptographic algorithms for public/private key pairs are based on random number generators, large prime numbers, elliptic curves, or other cryptographic techniques for producing globally unique values that do not require a central registry to effectively guarantee uniqueness. We say “effectively guarantee” because there is an infinitesimally small chance of a collision⁸ with someone else using the same algorithm. But this chance of collision is mathematically so small that for all practical purposes it can be ignored.

As a result, anyone with the proper software can generate a DID according to a particular DID method (discussed in the following sections) and begin using it immediately without requiring the authorization or involvement of any centralized registration authority. This is the same process used to create public addresses on the Bitcoin or any other popular blockchain—it is the essence of what makes a DID *decentralized*.

⁸ See https://en.wikipedia.org/wiki/Universally_unique_identifier#Collisions

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

The functional level: how DIDs work

DID documents

Identifiers can of course be useful just by themselves, as strings of text characters that can be used to refer to a resource. This string can be stored in a database or in a document, or it could be printed on a t-shirt or business card, or attached to an email. But for digital identifiers, the usefulness comes just not from the identifier itself, but from how it can be used by applications designed to consume that particular type of identifier. For example, a typical Web address that starts with "http" or "https" is not very interesting as a string by itself. It only becomes useful once you type it into a Web browser or click on a hyperlink in order to access a representation of the resource behind the identifier (such as a Web page).

This is similar with DIDs: Although (at least today) it is not possible to type a DID into a Web browser, you can give it to a specialized piece of software (or hardware) called a **DID resolver** that will use it to retrieve a standardized data structure called a **DID document**. This data structure is not like a Web page or an image file, in other words it is not designed to be viewed directly by end users in a Web browser or similar software.⁹ Instead, it is consumed by digital identity applications or services such as wallets, agents, or personal data stores, all of which use DIDs as fundamental building blocks.

Every DID has exactly one associated DID document. The DID document contains metadata about the **DID subject**, which is the term for the resource **identified by** the DID and **described by** the DID document. For example, a DID for a person (the DID subject) has an associated DID document that contains metadata about that person. The entity that controls the DID and its associated DID document is called the **DID controller**. In many cases, this is the same as the **DID subject**, but they could also be different. An example is when a parent controls a DID that identifies their child—the DID subject is the child but the DID controller (at least until the child comes of age) is the parent.

⁹ In the future, "DID navigators" may be able to let you travel across a "trust web", but most likely this will be done by associated conventional Web pages with DIDs.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

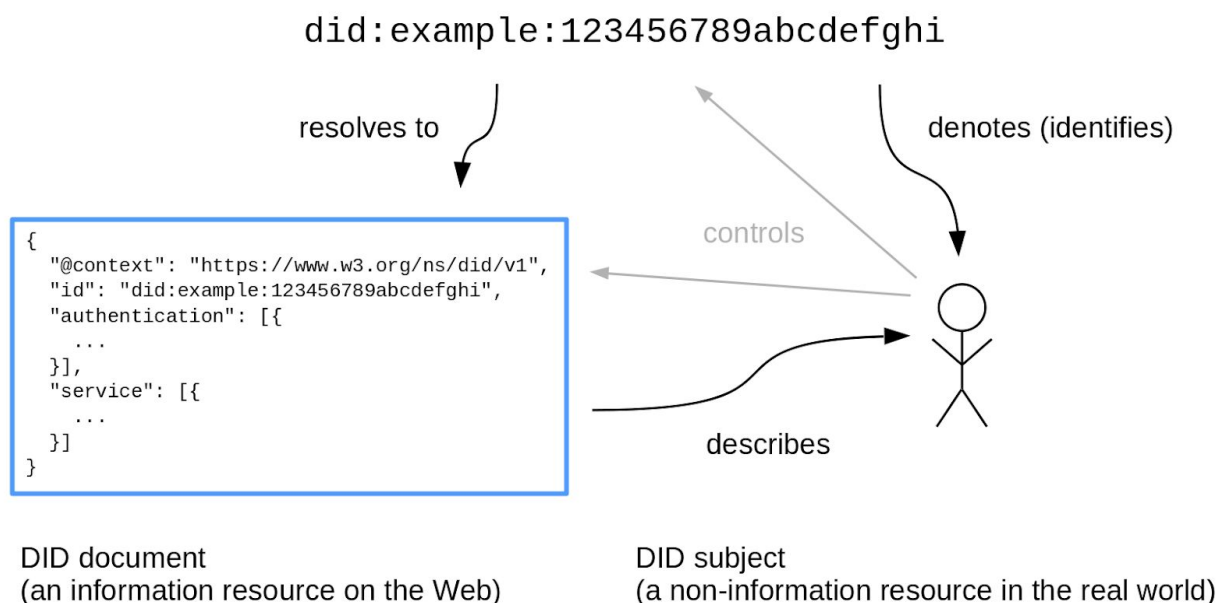


Figure 8.x: Relationships between the DID, DID document, and DID subject (in cases where the DID subject is also the DID controller)

Theoretically, a DID document can contain any arbitrary information about the DID subject, even personal attributes such as a name or an e-mail address. In practice however, this is problematic for privacy reasons; instead, the recommended best practice is for a DID document to contain only the minimum amount of machine-readable metadata required in order to enable **trustable interaction** with the DID subject. Typically, this includes the following:

- One or more **public keys** (or other verification methods) that can be used to authenticate the DID subject during an interaction. This is what makes interactions involving DIDs trustable, and this is also the essence of the DPKE enabled by DIDs.
- One or more **services** associated with the DID subject that can be used for concrete interaction via protocols supported by those services. This can include a wide range of protocols from instant messaging and social networking, to dedicated identity protocols such as OpenID Connect (OIDC), DIDComm as described in chapter 5 about SSI architecture, and others.
- Certain additional metadata such as **timestamps**, **digital signatures** and other **cryptographic proofs**, or metadata related to **delegation and authorization**.

Figure 8.x is an example of a very simple DID document using a JSON-LD encoding.

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58" : "H3C2AVvLMv6gmMnam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "service": [{
    "id": "did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

Figure 8.x: Example DID document with one public key used for authentication, and one service

This metadata that is associated with every DID, especially public keys and services, is the technical basis for all interaction that happens between different actors in an SSI ecosystem.

DID methods

As we explained in the previous sections, DIDs are not created and maintained in a single type of database or network like most other types of URIs. There is no authoritative centralized registry—or a hierarchy of federated registries like DNS—where all DIDs are written and read. In fact, many different types of DIDs exist in today's SSI community—see the *Types of DIDs* section later in this chapter. They all support the same basic functionality but they differ in how that functionality is implemented, e.g. how exactly a DID is created, or where and how a DID's associated DID document is stored and retrieved.

These types or classes of DIDs are known as **DID methods**. The second part of the DID identifier format—between the first and second colons—is called the **DID method name**. Figure 8.x shows examples of DIDs created using five different DID methods: `sov` (Sovrin), `btcr` (Bitcoin), `v1` (Veres One), `ethr` (Ethereum), and `jolo` (Jolocom).

<code>did:sov:WRfXPg8dantKVubE3HX8pw</code>
<code>did:btc:xyz35-jzv2-qqs2-9wjt</code>
<code>did:v1:test:nym:3AEJTDMSxDDQpyUftjuoeZ2Bazp4Bswj1ce7FJGybCUu</code>
<code>did:ethr:0xE6Fe788d8ca214A080b0f6aC7F48480b2AEfa9a6</code>
<code>did:jolo:1fb352353ff51248c5104b407f9c04c3666627fcf5a167d693c9fc84b75964e2</code>

Figure 8.x: Example DIDs generated using five different DID methods

As of the writing of this book, more than 40 DID method names have been registered at the informal DID Method Registry¹⁰ maintained by the W3C Credentials Community Group. Each DID method is required to have its own technical specification, which must define the following aspects of the DID method:

- The syntax of the third part of the DID identifier format (the portion after the second colon). This is called the **method-specific identifier** and is typically a long string which is generated using random numbers and cryptographic functions. It is always guaranteed to be unique within the DID method namespace.
- The four basic operations that can be executed on a DID:
 - **Create:** How can a DID and its associated DID document be created?
 - **Read:** How can the associated DID document be retrieved?
 - **Update:** How can the contents of the DID document be changed?
 - **Deactivate:** How can a DID be deactivated so it can no longer be used?
- Security and privacy considerations specific to the DID method.

It is difficult to make generic statements about the four DID operations, since DID methods can be designed in very different ways. For example, some DID methods are based on blockchains or other distributed ledgers. In this case, creating or updating a DID typically involves writing a transaction to that ledger. Other DID methods do not use a blockchain; they implement the four DID operations in other ways (see *Types of DIDs* later in this chapter).

One consequence of the technological variety of DID methods is that some may be better suited for certain use cases than others. DID methods may differ in how "decentralized" or "trusted" they are, as well as in factors of scalability, performance, or cost of the underlying technical infrastructure. The charter of the W3C DID Working Group¹¹ includes a deliverable of a "rubric" document to help evaluate how well a particular DID method will meet the needs of a particular user or community.

DID resolution

¹⁰ <https://w3c-ccg.github.io/did-method-registry/>

¹¹ <https://www.w3.org/2019/09/did-wg-charter.html>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

The process of obtaining the DID document associated with a DID is called *DID resolution*. This process allows DID-enabled applications and services to discover the machine-readable metadata about the DID subject that is expressed by the DID document. This metadata can be used for further interaction with the DID subject, for example:

- To look up a public key in order to verify a digital signature from an Issuer of a Verifiable Credential.
- To authenticate the DID controller when he/she needs to “log in” to a website or app.
- To discover and access a well-known service associated with the DID controller, such as a website, social network, or licensing authority.
- To request a DID-to-DID connection with the DID controller.

① Verifiable Credentials

```
{
  "issuer": "did:example:456",
  "credentialSubject": {
    "id": "did:example:123",
    "degree": "M.Sc."
  },
  "proof": {
    "jws": "eyJhbGciOiJIUzI1Ni...",
    ...
  }
}
```

A verifier resolves the issuer's DID, in order to look up the public key needed to verify the signature on a Verifiable Credential.

② Login



A relying party resolves a user's DID, in order to look up the public key needed to verify a proof during a challenge-response authentication protocol.

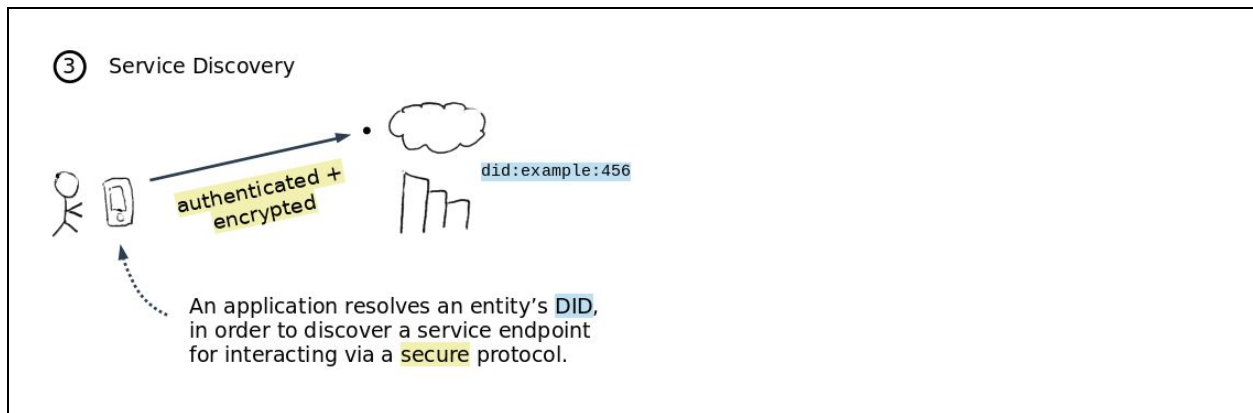


Figure 8.x: Common scenarios requiring DID resolution—the first one is using a DID to identify the issuer of a verifiable credential; the second to login to a website, and the third to discover a service associated with the DID

The DID resolution process is based directly on the Read operation defined by the applicable DID method, which as noted can vary considerably depending on how the DID method is designed. This means DID resolution is not confined to a single protocol in the same way as DNS (which makes it possible to resolve domain names to IP addresses) or HTTP (which is used to retrieve representations of resources from Web servers).

For example, no assumption should be made that a blockchain, distributed ledger, or database (centralized or decentralized) is used for resolving DIDs, or even that interaction with a remote network is required during the DID resolution process. Furthermore, a DID document is not necessarily stored in plain-text in a database or available for download from a server. Though some DID methods may work like this, others may define more complex DID resolution processes that involve on-the-fly construction of a "virtual" DID document.

Therefore, rather than thinking of DID resolution as a protocol, it should be considered an abstract function or algorithm, which takes a DID (plus optional additional parameters) as its input, and returns the DID document (plus optional additional metadata) as its result.

DID resolvers can come in several architectural forms: they can be implemented as a native library included in an application or even an operating system, in the same way as DNS resolvers are included in all modern operating systems. Alternatively, a DID resolver can be provided by a third party as a hosted service, responding to DID resolution requests via HTTP or other protocols (called *bindings*). Mixed forms are also possible. For example a local DID resolver could delegate part of or all of the DID resolution process to a pre-configured remotely hosted DID resolver. This is similar to how DNS resolvers in our local operating systems typically query a remote DNS resolver hosted by an Internet Service Provider (ISP), which performs the actual DNS resolution work.



Figure 8.x: An example of a local DID resolver querying a remote DID resolver, which then retrieves the DID document according to the applicable DID method

Of course reliance on an intermediary service during the DID resolution process introduces potential security risks and potential elements of centralization, both of which can impact the security and trust properties of other layers in the SSI stack that rely on DIDs. Therefore whenever possible DID resolution should be integrated directly into a DID-enabled application in such a way that the application is able to independently verify that the DID resolution result is correct (meaning the correct DID document has been obtained).

DID URLs

DIDs are powerful identifiers by themselves, but they can also be used as the basis for constructing more advanced URLs rooted in a DID. This is similar to how http and https URLs can consist not only of a domain name, but also have other syntactic components appended to the domain name: an optional path, an optional query string, and an optional fragment. With URLs, these enable identifying arbitrary resources under the authority of the domain name. The same is true for DIDs. This means that, although the most important function of a DID is to resolve it to a DID document, it can also serve as a root authority of a set of **DID URLs** that enable an "identifier space" for additional resources associated with the DID.



Figure 8.x: DIDs and DID URLs

DID URLs can be used for many different purposes. Some uses of DID URLs will be well-known and standardized, whereas others are completely extensible and DID method- or application-dependent. See Table 8.1 for example DID URLs and their meanings:

Table 8.1: Example DID URLs and their meanings

<pre>did:example:1234/</pre> <p>The simplest possible DID URL that is just the DID itself plus a forward slash to identify the DID document. See the very last section of this chapter for more about this.</p>

```
did:example:1234#keys-1
```

DID URL with a fragment. This DID URL identifies a specific public key inside the DID's associated DID document. This is similar to how an "http" or "https" URL with a fragment can point to a specific part of an HTML Web page.

```
did:example:1234;version-id=4
```

DID URL with a DID parameter ("version-id"). This DID URL identifies a previous version of the DID's associated DID document, as opposed to the latest version. This is useful in situations when the contents of a DID document have been updated, but a stable reference to a specific version is needed.

```
did:example:1234;version-id=4#keys-1
```

Combination of the previous two examples. This DID URL identifies a specific public key inside a specific previous version of the DID's associated DID document.

```
did:example:1234/my/path?query#fragment
```

DID URL with additional syntactic components (path, query string, fragment). The meaning and processing rules of this DID URL are not defined by the core DID standard, but are dependent on the DID method and/or the applications that use these identifiers.

```
did:example:1234;service=hub/my/path?query#fragment
```

DID URL with a DID parameter ("service"). This DID URL identifies a specific service inside the DID's associated DID document (in this case, the "hub" service). The meaning and processing rules of the remaining syntactic components (path, query string, fragment) are not defined by the core DID standard, but are specific to the "hub" service.

Once a DID document has been retrieved by a DID resolver—the process called DID *resolution*—for most DID URLs there is a second stage called DID *dereferencing*. Whereas resolution only returns the DID document, in the dereferencing stage, the DID document is further processed to accessing or retrieving the resource identified by the DID URL (which almost always is a *different resource* than the DID subject). These two terms—resolution and dereferencing—are defined by the URI standard (RFC 3986), and they apply not only to DIDs, but to all types of URIs and the resources they identify. Figure 8.10 is an example of how these two processing stages differ.

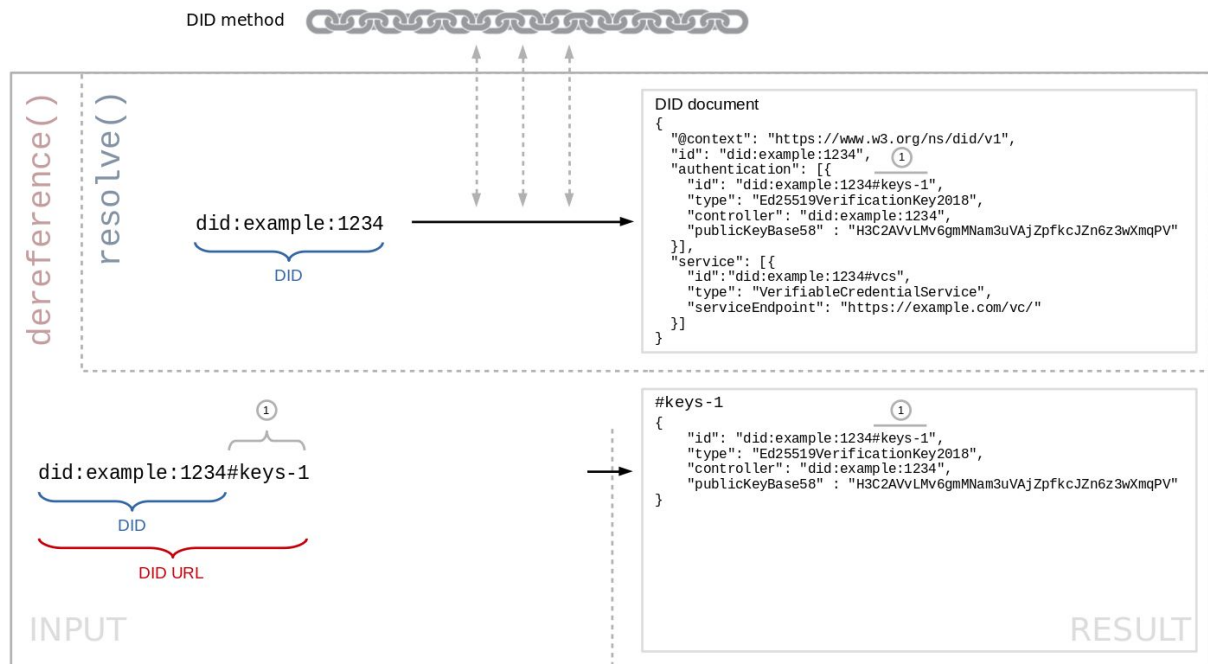


Figure 8.x: The process of first resolving a DID and then dereferencing a DID URL containing a fragment; the result is a specific public key inside the DID document

Comparison with the Domain Name System (DNS)

We have already used domain names and the domain name system (DNS) as an analogy when explaining certain aspects of DIDs and how they are different from other identifiers. The following table summarizes the similarities and differences between DIDs and domain names.

Table 8.x Comparison of DIDs with domain names

Decentralized Identifiers (DIDs)	Domain Names
Globally unique	Globally unique
Persistent	Reassignable
Machine-friendly identifiers (i.e., long character strings based on random numbers and cryptography)	Human-readable names
Resolvable using different mechanisms defined by the applicable DID method	Resolvable using the standard DNS protocol
Associated data is expressed in DID	Associated data is expressed in DNS zone

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

documents	files
Fully decentralized namespace without delegation	Hierarchical, delegatable namespaces based on centralized root registries of top-level domain names (TLDs)
Secured by DID method-specific processes and infrastructure (e.g. blockchains)	Secured by trusted root registries and traditional PKI (DNSSEC) ¹²
Cryptographically-verifiable	Verifiable using DNS security extensions (DNSSEC)
Used as authority component in DID URLs	Used as authority component in "http" and "https" Web addresses as well as e-mail addresses and other identifiers
Governed by the authority for each DID method (anyone can create a DID method)	Governed by the Internet Corporation for Assigned Names and Numbers (ICANN)
Fully under the control of the DID controller	Ultimately controlled by ICANN and the registry operator for each DNS TLD

Comparison with Uniform Resource Names (URNs) and other Persistent Identifiers

As we explained previously, DIDs meet the functional requirements of URNs, i.e., they are persistent identifiers that always identify the same entity and cannot be reassigned. There are many other types of persistent identifiers that differ from DIDs in ways that make them generally less suitable for SSI applications. The following table lists other types of persistent identifiers and how they compare to DIDs.

Table 8.x: Comparison of DIDs with other types of persistent identifiers

Other URNs	<ul style="list-style-type: none"> • Either not resolvable, or the resolution process and metadata varies with each type • Not cryptographically verifiable
Universally Unique Identifiers (UUIDs, also called Globally Unique Identifiers or GUIDs)	<ul style="list-style-type: none"> • Not resolvable • Not cryptographically verifiable
Persistent URLs (PURLs) Handle System (HDLs) Digital Object Identifiers (DOIs), Archival Resource Keys (ARKs), Open Researcher and Contributor ID (ORCID)	<ul style="list-style-type: none"> • Not decentralized, i.e. creating and using these identifiers depends on a central or hierarchical authority • Not cryptographically verifiable

¹² https://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Figure 8.11 is a visual way of depicting how most of the other URIs in use today compare to DIDs. Note that it does not include a circle for *cryptographically verifiability* because DIDs are the only identifier that offers that property. However it does include a circle for one property DIDs do *not* have: *delegatability*. That is the ability for one identifier authority to delegate a subnamespace to another identifier authority. An example is a domain name like `maps.google.com`, where the `.com` registry delegates to Google and Google delegates to its maps service.

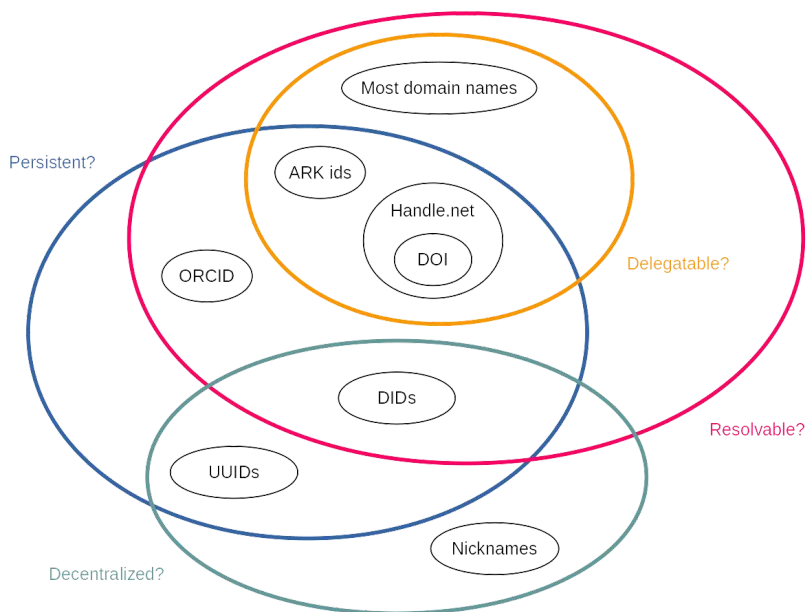


Figure 8.x: Compared to other identifiers, DIDs are persistent, resolvable, and decentralized (cryptographic verifiability is not shown since none it does not apply to any of the other identifiers)

Types of DIDs

Since their invention, interest in DIDs has grown exponentially (for reasons we explain in much greater detail in the next section). Although the first DID methods were closely tied to blockchains and distributed ledgers, as DIDs started evolving, many more types of DIDs have been developed. A series of presentations at the first meeting of the W3C DID Working Group in September 2019 in Fukuoka, Japan, described the various DID methods that have been developed at that point as falling into the broad categories described in Table 8.4.

Table 8.x: The broad categories of DID methods that have been developed as of September 2019

Category	Description & Examples
Ledger-based DIDs	<p>The "original" category of DID methods involves a blockchain or other distributed ledger technology (DLT), which serves the purpose of a registry that is not controlled by a single authority. This registry is typically public and globally accessible. A DID is created/updated/deactivated by writing a transaction to the ledger, which is signed with the private key of the DID controller.</p> <pre>did:sov:WRfXPg8dantKVubE3HX8pw did:btcr:xz35-jzv2-qqs2-9wjt did:ethr:0xE6Fe788d8ca214A080b0f6aC7F48480b2AEfa9a6 did:vl:test:nym:3AEJTDMSxDDQpyUftjuoeZ2Bazp4Bswj1ce7FJGybCUu</pre>
Ledger Middleware ("Layer 2") DIDs	<p>An improvement to classic ledger-based DID methods, this category adds an additional storage layer such as a distributed hash table (DHT) or traditional replicated database system "on top" of the base layer blockchain. DIDs can be created/updated/deactivated at this second layer without requiring a base layer ledger transaction every time. Instead, multiple DID operations are batched into a single ledger transaction, increasing performance and decreasing cost.</p> <pre>did:ion:test:EiDk2RpPVuC4wNANUTn_4YXJczjzi10zLG1XE4AjkcGOLA did:elem:EiB9htZdL3stukrklAnJ0hrWuCdXwR27TND07Fh9HGWDGg</pre>
Peer DIDs	<p>This is a special category for a DID method that does not require a globally shared registration layer such as a blockchain. Instead, a DID is created and subsequently exists only within a relationship between a limited number of participants. The DIDs that are part of the relationship are exchanged via a peer-to-peer protocol, resulting in private connections between the participants.</p> <pre>did:peer:1zQmZMygzYqNwU6Uhmewx5Xepf2VLp5S4HLSwwgf2aiKZuwa</pre>
Static DIDs	<p>There is also a category of DID methods that are "static", i.e. they can only be created and resolved, but not updated or deactivated. Such DID methods tend to not require complex protocols or storage infrastructure. For example, a DID may simply be a "wrapped" public key, from which an entire DID document can be resolved algorithmically, without requiring any data other than the DID itself.</p> <pre>did:key:z6Mkfriq1MqLBoPWecGoDLjguo1sB9brj6wT3qZ5BxkKpuP6</pre>
Alternative DIDs	<p>A number of other innovative DID methods have been developed that do not fall into any of the previous categories. They demonstrate that the DID concept is flexible enough to be layered on top of existing Internet protocols, such as Git, the Interplanetary File System (IPFS), or even the Web itself.</p> <pre>did:git:625557b5a9cdf399205820a2a716da897e2f9657</pre>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

	did:ipid:QmYA7p467t4BGgBL4NmyHtsXMoPrYH9b3kSG6dbgFYskJm did:web:uport.me
--	---

The architectural level: why DIDs work

Having explained what DIDs are and how they work, let’s now go a step deeper into *why* they work. What is there so much interest in this new type of identifier?

To answer this, we must delve more deeply into the core problems they solve, which are less problems of identity and more problems of *cryptography*.

The core problem of Public Key Infrastructure (PKI)

Since it was first conceived, PKI has one hard problem at its very core. It is not a problem with cryptography per se, i.e., with the math involved with public/private keys or encryption/decryption algorithms. Rather it is a problem with *cryptographic infrastructure*, i.e., how we can make public/private key cryptography easy and safe for people and organizations to use at scale.

This is not an easy problem—in fact it has vexed PKI ever since the term was invented. The reason lies in the very nature of how public/private key cryptography works. To understand this, let’s take a look at the basic PKI “trust triangle” (figure 8.x). It shows that it’s not enough to think about public/private “key pairs”. You have to see each key pair in relation to its controlling authority (“controller”), be that a person, an organization, or even a thing (if the thing has the capacity to generate key pairs and store them in a digital wallet).

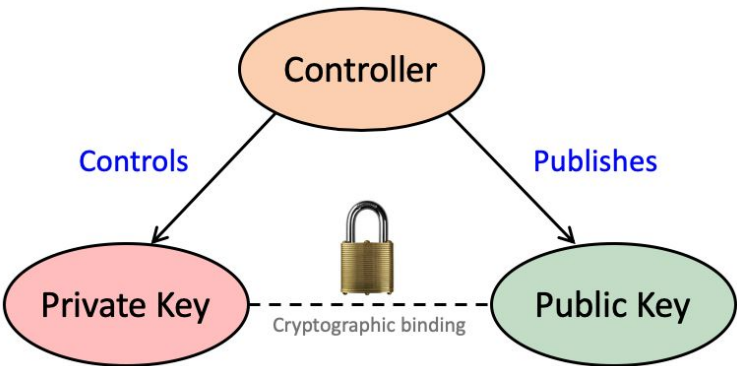


Figure 8.x: The basic trust triangle at the heart of all public/private key cryptography

Public and private keys are bound to each other mathematically such that neither can be forged—each can be used only for a specific set of functions defined by a specific

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

cryptographic algorithm. But *both* types of keys are intrinsically related to the controller. Regardless of the algorithm, these two fundamental roles are highlighted in figure 8.x. The private key must be reserved for the exclusive use of the controller (or its delegates) and must never be revealed to anyone else. By contrast the public key is just the opposite: it **must** be shared with any party that wishes to securely communicate with the controller. It is the only way to encrypt messages to—or verify messages from—that controller.

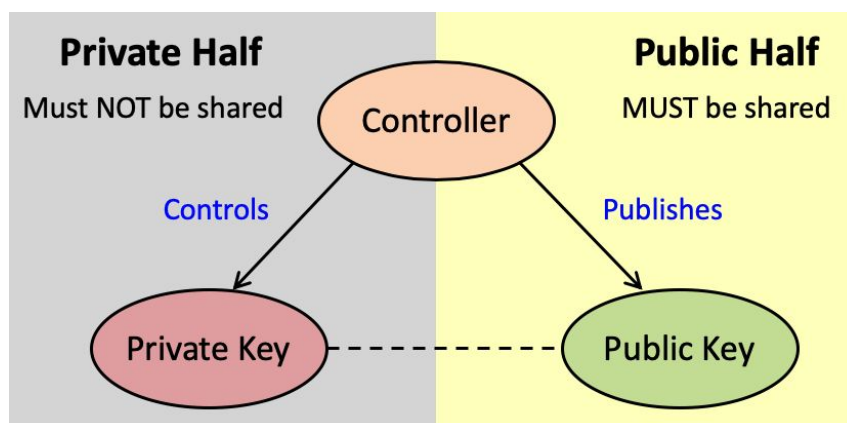


Figure 8.x: The fundamental roles of public keys vs. private keys in PKI

Although the task of keeping a private key private is not trivial by any means, that is not the hard problem at the heart of PKI. Rather the hard problem is on the *other* side of the PKI trust triangle as shown in figure 8.14.

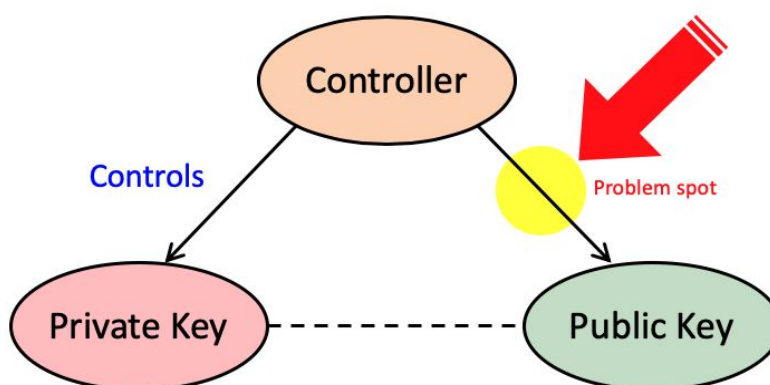


Figure 8.14: The core problem at the heart of PKI: how do you bind a public key to its controller?

The problem is simply this: *how do you strongly bind a public key to its controller* so that any party relying on that public key (the **relying party**) can be sure they are dealing with the real controller? After all, if you can fool a relying party into accepting the public key for controller B when the relying party thinks it is the public key for controller A, then for all

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

intents and purposes controller B can fully impersonate controller A. The cryptography will all work perfectly and the relying party will never know the difference—until they become the victim of whatever cybercrime controller B is perpetrating.

Therefore, as a relying party, it is essential to know *you have the correct public key at the correct point in time* for any controller you are dealing with. And that indeed is a hard problem because, whereas public keys are purely digital entities whose cryptographic validity can be verified in milliseconds, controllers are **not**. They are real people, organizations, or things that exist in the real world. So the only way to digitally bind a public key to a controller is to add one more piece of the puzzle: a digital identifier for the controller.

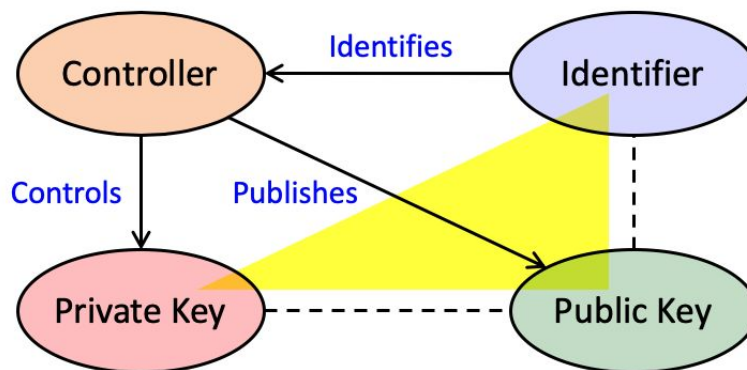


Figure 8.15: The *real* PKI trust triangle includes a digital identifier for the controller

The *real* PKI trust triangle is the yellow triangle shown in figure 8.15. The new piece is an identifier for the controller that can be bound to the public key in such a way that relying parties can be confident the public key belongs to the controller and nobody else.

What figure 8.15 reveals is that this identifier-binding problem actually has two parts:

1. How do you strongly bind the identifier to the controller?
2. How do you strongly bind the public key to the identifier?

These two problem spots are called out in figure 8.16:

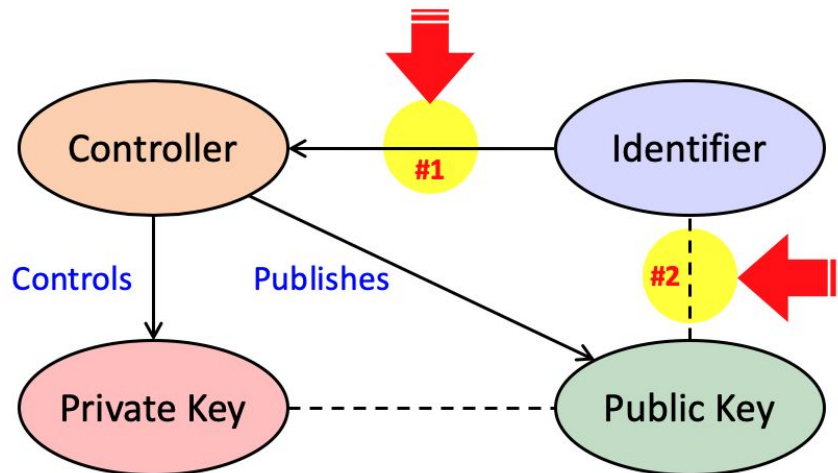


Figure 8.16: The two problem spots when it comes to strongly binding an identifier to a controller

These are the two problem areas that PKI has struggled with since it was born in the 1970s. In the balance of this section we'll look at four different solutions to these two problems:

1. The conventional PKI model
2. The web-of-trust model
3. Self-certifying identifiers
4. DIDs

Solution #1: The conventional PKI model

The first solution is the conventional PKI model for issuing **digital certificates** ("certs"). This is the predominant model that has evolved over the last 40 years. Probably the best known example is the SSL/TLS PKI that uses X.500 certs to provide secure connections in our browsers using the HTTPS protocol (the lock you see in your browser address bar).

The conventional PKI solution to the first problem—identifier-to-controller binding—is to use one of the most suitable existing identifiers and then follow industry best practices for ensuring its strong binding to a controller. Table 8.5 summarizes the available choices of identifiers and highlights the ones used most often in X.500 certificates.

Identifier	Challenges with Strong Binding
Phone Number	Reassignable, limited #, hard to register
IP Address	Reassignable, hard to register
Domain Name	Reassignable, spoofable, DNS poisoning
Email Address	Reassignable, spoofable, weak security
URL	Depends on a Domain Name or IP Address
X.500 Dist. Name	Hard to register

X.500 Certs

Table 8.5: The different identifier choices for conventional PKI—highlighted are those typically used in X.500 digital certificates

The primary advantage of a URL (based on a domain name or IP address) is that there are automated tests that can be performed to ensure that the controller of the public key also controls the URL. However these tests cannot detect homographic attacks¹³ (using look-alike names or look-alike characters from different international alphabets) or DNS poisoning¹⁴.

The primary advantage of an X.500 Distinguished Name (DN) is that it can be administratively verified to belong to the controller. However this verification must be performed manually and thus is always subject to human error. Furthermore, it is not an easy process to register an X.500 DN—certainly not something the average Internet user can be expected to undertake.

The conventional PKI approach to the second problem—public-key-to-identifier binding—seems obvious in the context of cryptography: *digitally sign a document* containing both the public key and the identifier. That is the origin of the **public key certificate**¹⁵ (a specific kind of digital certificate). This solution is illustrated in figure 8.17:

¹³ https://en.wikipedia.org/wiki/IDN_homograph_attack

¹⁴ https://en.wikipedia.org/wiki/DNS_spoofing

¹⁵ https://en.wikipedia.org/wiki/Public_key_certificate

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

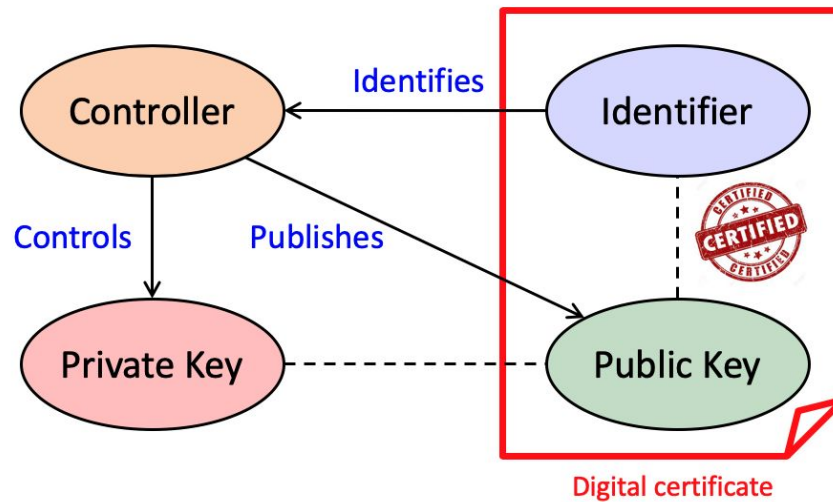


Figure 8.17: Conventional PKI solves the public-key-to-identifier binding problem using digital certificates signed by some type of certificate authority

The question, of course, is *who digitally signs this digital certificate*? This introduces the whole notion of a **trusted third party (TTP)**¹⁶—someone the relying party must trust to sign the digital certificate or else the whole idea of PKI falls apart. The conventional PKI answer is a **certificate authority (CA)**—a service provider whose entire job is to follow a specified set of practices and procedures to confirm the identity of a controller and the authenticity of its public key before issuing a digital certificate that binds the two and signing it with the CA’s private key.

Different PKI systems use different certification programs for CAs; one of the best known is the **WebTrust** program originally developed by American Institute of Certified Public Accountants and now run by the Chartered Professional Accountants of Canada¹⁷. Certification is obviously critical for CAs because acting as a TTP is inherently a human process—it cannot be automated (if it could, a TTP would not be needed). And unfortunately humans make mistakes.

¹⁶ https://en.wikipedia.org/wiki/Trusted_third_party

¹⁷

<https://www.cpacanada.ca/en/business-and-accounting-resources/audit-and-assurance/overview-of-webtrust-services>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

But being human is only one issue with TTPs. Table 8.6 lists the other drawbacks.

Drawbacks of TTPs	Description
Cost	Inserting a TTP (that must perform work only humans can do) into trust relationships adds costs that someone must pay for
Friction	Introducing a TTP requires additional work on the part of all the parties to a trust relationship
Single point of failure	Each TTP becomes an attack point because a single breach can compromise all of its digital certificates
Identifiers changing	If an identifier changes, the old digital certificate must be revoked and a new one issued
Public keys changing	When a public key is rotated (as most security policies require periodically), the old digital certificate must be revoked and a new one issued

Table 8.6: The drawbacks of using conventional PKI to solve the identifier binding problem

Despite all these drawbacks, conventional PKI has so far been the only commercially viable solution to the identifier binding problem. But as the Internet has climbed in usage and commercial value—and rate cybercrime with it—so has the demand for a better solution.

Solution #2: The web-of-trust model

One alternative to the conventional PKI model was formulated early on by one of the pioneers in public/private key cryptography, Phillip Zimmermann, inventor of Pretty Good Privacy (PGP). He coined the term “web of trust” for this model because it didn’t rely on centralized CAs, but rather on individuals who knew each other directly and therefore could individually sign each others public keys—effectively creating *peer-to-peer digital certificates*. Here’s Mr. Zimmermann’s description of this model from the 1992 manual for PGP version 2.0¹⁸:

As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

¹⁸ https://en.wikipedia.org/wiki/Public_key_infrastructure#Web_of_trust

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

Figure 8.18 is a visual depiction of how the web-of-trust model¹⁹ works.

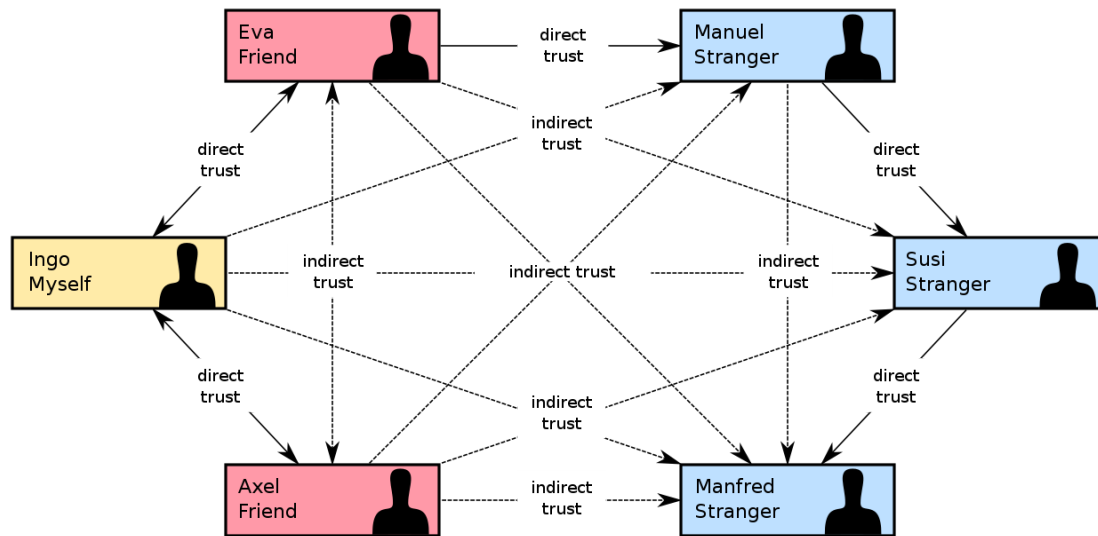


Figure 8.18: A visual diagram of how a web-of-trust can be constructed.

Since the original formulation, hundreds of academic papers have been written exposing issues and proposing improvements for the web-of-trust model²⁰. The primary challenge, however, is that the only thing it really changes in the conventional PKI model is *who* signs the digital certificate. It turns the problem of “who do you trust” (the TTP problem) into the problem of “who do you trust who knows someone else you can trust”, i.e., how do you discover a “trusted path” to the digital certificate you want to verify. So far no one has developed a reasonably secure, scalable, adoptable solution to this problem.

Solution #3: Self-certifying identifiers

The drawbacks of both the conventional and web-of-trust PKI models ultimately led to a very different approach: remove the need for a TTP altogether by replacing it with yet another clever use of cryptography. In other words, rather than trying to reuse an existing identifier for the controller (e.g., domain name, URL, X.500 Distinguished Name) and then binding it to the public key, reverse the whole process and *generate an identifier for the controller based on the public key itself* (either directly, or via a transaction with a blockchain, distributed ledger, or similar system).

This new approach to constructing the PKI trust triangle can be visualized in figure 8.19.

¹⁹ https://en.wikipedia.org/wiki/Web_of_trust

²⁰ https://en.wikipedia.org/wiki/Web_of_trust#WOT_assisting_solutions

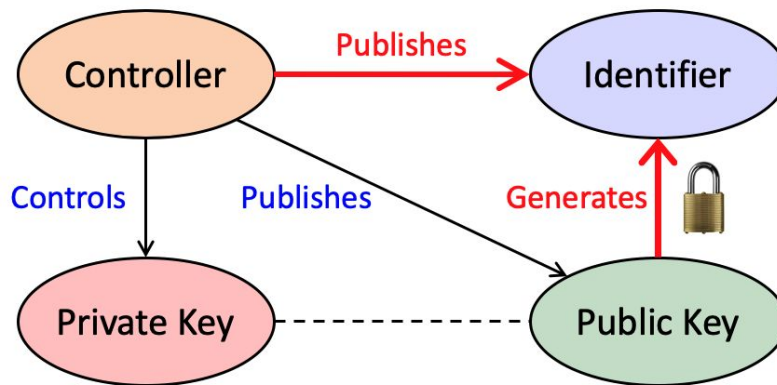


Figure 8.19: Self-certifying identifiers take a completely different approach to the identifier binding problem by generating the identifier for the controller from the public key

There are two obvious benefits of this approach. First, it solves the public-key-to-identifier binding problem by binding the public key to the identifier the same way the public key is bound to the private key: cryptographically. This makes the identifier **self-certifying** because no external certificate authority or digital certificate is needed to certify the binding with the public key—it can be done cryptographically in milliseconds.

Secondly, it solves the identifier-to-controller binding problem because *only the controller of the private key can prove control of the identifier*. In other words, under the self-certifying identifier model, *the controller controls all three points of the real PKI trust triangle* because all three values are generated cryptographically using key material that only the controller possesses.

As powerful as this solution appears, completely self-certifying identifiers have one major Achilles heel: the controller's identifier needs to change every time the public key is rotated. As we will explain further in the chapter devoted to key management, key rotation—switching from one public/private key pair to a different one—is a fundamental security best practice in all types of PKI. Thus the inability for basic self-certifying identifiers to support key rotation has effectively prevented their adoption as an alternative to conventional PKI.

Solution #4: DIDs

What if there was a way to generate a self-certifying identifier once, and then be able to continue to verify it after every key rotation? Enter the DID.

First, the controller generates the original self-certifying identifier—the DID—once, based on the genesis public/private key pair as shown in figure 8.19. Next the controller publishes the original DID document containing the DID and the public key as shown in figure 8.20.

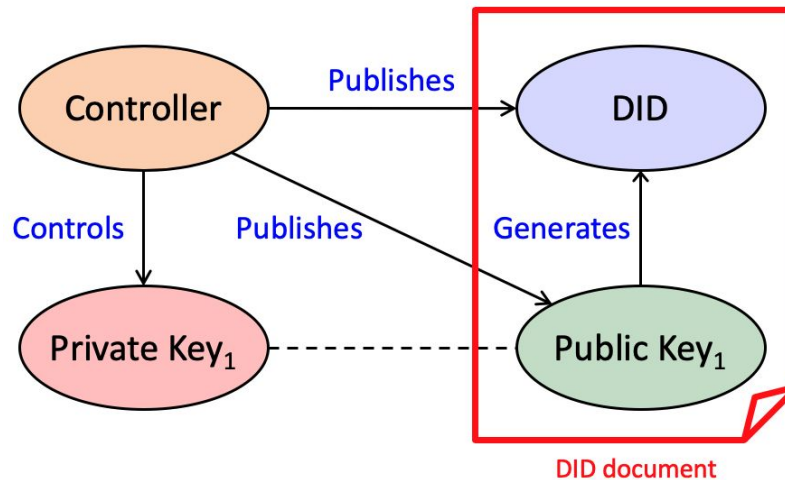


Figure 8.20: The controller publishes the original DID and public key is the original DID document

At this point anyone with access to the DID document can cryptographically verify the binding between the DID and the associated public key—no TTP or other external authority is needed.

Now, when the controller needs to rotate the key pair, the controller creates an *updated DID document* and signs it with the *previous private key* as shown in figure 8.21. Again, this is *self-certification* because it is performed only by the controller—no external authority is needed.

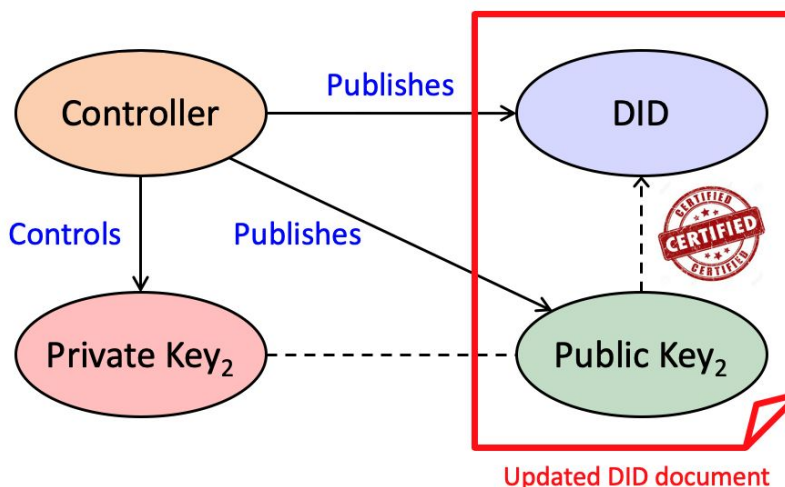


Figure 8.21: The controller publishes an updated DID document containing the original DID and the new public key and then digitally signs it with the original private key, creating a chain-of-trust between the DID documents

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

This “chain of trust” between DID documents can be traced back through any number of updates to the original DID document with the original self-certifying identifier. Essentially each DID document serves as a new digital certificate for the new public key as shown—but without the need for a CA or any other TTP to certify it.

Four benefits of DIDs that go beyond PKI

So DIDs allow us to finally achieve broad adoption of self-certifying identifiers and still enjoy key rotation and other essential features of conventional PKI—without the drawbacks. But the benefits of DIDs do not stop there. In this section we’ll cover four benefits of DIDs that go *beyond* what is offered by PKI as we know it today.

Beyond PKI benefit #1: Guardianship and controllership

To begin with, DIDs provide a clean way to identify entities *other* than the controller of the DID. Conventional PKI generally assumes the registrant of the digital certificate (the controller of the private key) is the party identified by the digital certificate. However there are many situations where this is not the case. Take a newborn baby. If he/she were to need a DID—for example, to be the subject of a birth certificate issued as a verifiable credential—the newborn is in no position to have his/her own digital wallet. He/she would need one of its parents (or another guardian) to issue this DID on its behalf. In this case the entity identified by the DID—the DID subject—is explicitly *not* the controller as shown in figure 8.22.

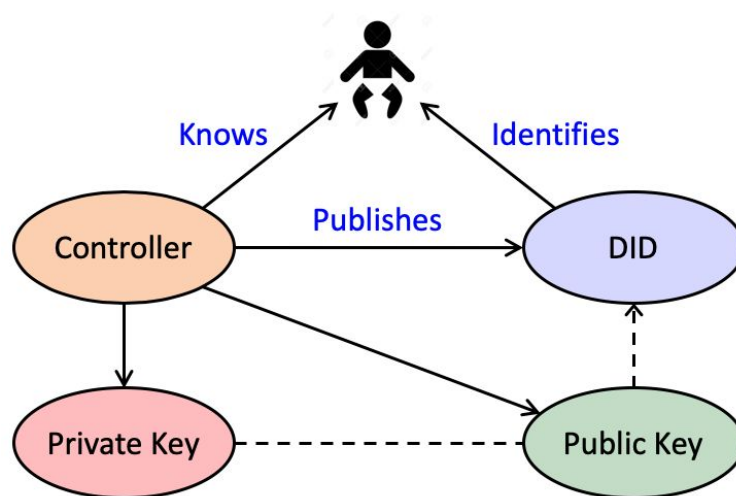


Figure 8.22: An example of a DID used to identify a DID subject (a newborn baby) that is *not* the controller of the DID.

Of course, newborns are just one case where a DID subject cannot be its own controller. There are dozens more just among humans: elderly parents, dementia patients, refugees,

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

homeless, anyone without digital access. All of these need the concept of **digital guardianship**²¹—a third party who accepts the legal and social responsibility to manage a digital wallet on behalf of a DID subject, called the **dependent**. SSI digital guardianship is a very broad, deep, and rich subject that is explored separately in the chapter on governance frameworks.

Digital guardianship only applies to humans, however. What about all the non-human entities in the world? The vast majority are not in a position to issue DIDs either. For example:

1. **Organizations of all kinds.** Every legal entity in the world that's not an individual human needs some form of identifier in order to operate within the law. Today they have business registration numbers, tax ID numbers, domain names, and URLs. Tomorrow they will have DIDs.
2. **Man-made things.** Virtually everything on the Internet of Things (IoT) can benefit from having one or more DIDs, but relatively few connected things (e.g., smart cars, smart drones) will be smart enough to have their own digital agents and wallets generating their own DIDs—and even then they will still be controlled by humans.
3. **Natural “things”.** Animals, pets, livestock, rivers, lakes, geological formations—not only do these have identities but in many jurisdictions they also have at least a limited set of legal rights. So they too can benefit from DIDs.

All of these represent categories of entities that need *third-party controllers*—a relationship that has been called **controllership**²² to differentiate it from guardianship of a human being. With guardianship and controllership, we can now extend the benefits of PKI to literally every entity that can be identified.

Beyond PKI benefit #2: Service endpoint discovery

The second added benefit of DIDs is their capacity to enable **discovery**, i.e., a way to determine how to interact with a DID subject. To do this, the DID controller publishes one or more service endpoint URLs in a DID document (see figure 8.10 for an example). This three-way binding is shown in figure 8.23.

²¹ <https://sovrin.org/guardianship/>

²² See the definition of “Controllership” in <https://sovrin.org/library/glossary/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

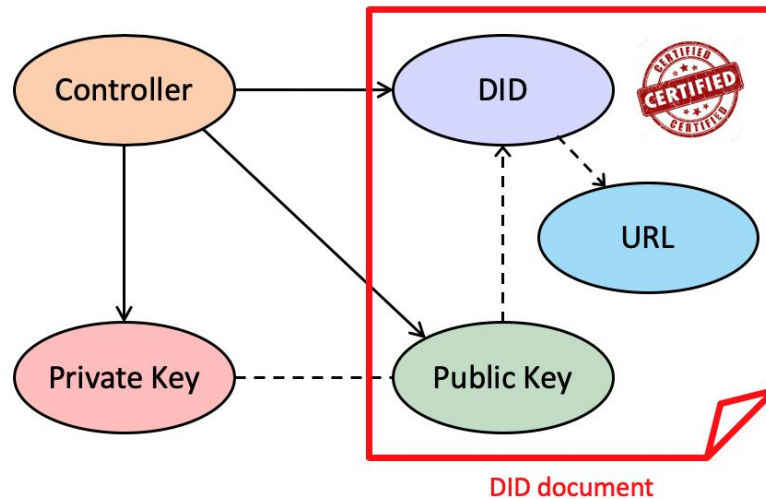


Figure 8.23: The three-way binding between DIDs, public keys, and service endpoint URLs.

While this makes DID documents generally useful for many types of discovery, it is essential for discovering the **agent endpoints** necessary to remotely establish DID-to-DID connections (below) and communicate via the DIDComm protocol (Chapter 5). In fact one could say that DID-based discovery of agent endpoint URLs is as essential to SSI as DNS-based discovery of IP addresses is to the Web.

Beyond PKI benefit #3: DID-to-DID connections

What made SSL/TLS the largest PKI in the world is that the public key in an X.509 digital certificate can be used for a secure HTTPS connection between a web server and a browser. In other words, it was demand for secure e-commerce, e-banking, e-health and other online transactions that drove the growth of the SSL/TLS PKI.

The same will be true in spades for SSI. Because DID documents can include both public keys and service endpoint URLs, every DID represents the opportunity for its controller to create an instant, secure, private, peer-to-peer connection with any other DID controller. Even better, unlike static public key certificates that must be obtained in advance from a CA, DIDs can be generated immediately, locally, on-the-fly as they are needed for new connections. In fact one DID method has been created exclusively for this purpose: the **Peer DID method**²³.

Developed by Daniel Hardman and the Hyperledger Aries community, peer DIDs do not need a blockchain, distributed ledger, or any other external database. They are generated locally in the DID controller’s digital wallet and exchanged directly, peer-to-peer, using a

²³ <https://openssi.github.io/peer-did-method-spec/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

protocol based on Diffie-Hellman key exchange.²⁴ This creates a connection between the two parties that is not like any other connection in traditional network architecture. Table 8.7 highlights the five special properties of DID-to-DID connections.

Table 8.7: The five properties of DID-to-DID connections

Property	Description
Permanent	The connection will <i>never break</i> unless one or both parties want it to.
Private	All communications over the connection can be automatically encrypted and digitally signed by the private keys for the DIDs.
End-to-end	The connection has no intermediaries—it is secure from “DID to DID”.
Trusted	The connection supports verifiable credential exchange to establish higher trust in any relevant context to any level of assurance.
Extensible	The connection can be used for any application that needs secure, private, reliable digital communications via the DIDComm protocol or any other protocol supported by both agents.

DID-to-DID connections, whether between peer DIDs (the default) or public DIDs, are the centerpiece of Layer Two of the Trust over IP stack described in Chapter 5. Agents at this layer communicate using the secure DIDComm protocol in much the same way web browsers and web servers communicate over the secure HTTPS protocol. DIDs “democratize” the SSL/TLS PKI plumbing so now secure connections can be available to anyone, anytime, anywhere.

Beyond PKI benefit #4: Privacy by Design

At this point it should be obvious how much DIDs can help increase security on the Internet. But while security is necessary for privacy, it is not sufficient. Privacy is more than just preventing private information from being snooped or stolen. It is also ensuring that parties with whom you *do* choose to share your private information (doctors, lawyers, teachers, governments, companies that sell you products and services) protect that information and do not use or sell it without your permission.

So how do DIDs help with *that*?

²⁴ https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

The answer may surprise you. Conventional identifiers for people are assigned one time by third parties: government ID numbers, health ID numbers, driving license numbers, mobile phone numbers. When you share these as part of your personal data, they make it easy for you to be tracked and correlated across many different relying parties. They also make it easy for those parties to share information and compile “digital dossiers” about you.

DIDs turn that whole equation upside down. Rather than you sharing the same ID number with many different relying parties, you generate and share your own **pairwise peer DID** each time you form a relationship with a new relying party. That relying party will be the only one in the world that knows that DID (and its public key and service endpoint URL). And that relying party will do the same for you.

So rather than you having a single DID, similar to a government-issued ID number, you will have *thousands of DIDs*—one per relationship. Each pairwise-unique peer DID gives you and your relying party your own **permanent private channel** connecting the two of you as envisioned by Philip Zimmermann with PGP. The first advantage of this channel is that you can authenticate each other—in both directions—automatically just by exchanging messages signed by each of your private keys. Spoofing or phishing a relying party with whom you already have a connection will become nearly impossible.

The second major advantage is that peer DIDs and private channels give you one simple, standard, verifiable way to share **signed personal data**—personal data for which you have granted specific permissions data to the relying party. On your side, the benefit is convenience and control—at a glance you can see what you shared with whom and why. On the relying party side, the benefit is fresh, first-person data with cryptographically-verifiable, GDPR-auditable consent—plus an easy, secure way to support all the other GDPR-mandated personal data rights (access, rectification, erasure, objection).²⁵

The third major benefit is that, with signed data, we can finally protect both individuals and relying parties from the damage caused by the massive data breaches we read about almost daily (Target, Equifax, Sony, Yahoo, Capital One). What motivates criminals to break into those data silos is the value of that personal data—primarily because the criminals can use it can break accounts all over the Internet.

As those accounts convert over to using pairwise peer DIDs and signed personal data, the value of that personal data *to anyone else but the relying party who has explicit signed permission* disappears. In fact, if you cannot cryptographically prove you have permission to use the data, then not only does the data become worthless—it becomes *toxic*. Mere possession of unsigned personal data could become a crime. Like toxic waste, it will be something companies, organizations, and even governments will want to get rid of as quickly as possible.

²⁵ https://en.wikipedia.org/wiki/General_Data_Protection_Regulation#III_Rights_of_the_data_subject
©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

So now you see why many in the SSI community consider DIDs, verifiable credentials, and the Trust over IP stack to be a sea change in privacy on the Internet. While we are still in the very early stages of implementing this new approach, it finally gives individuals new tools for controlling the use of their personal data. And this control, when bundled with the rest of the tools in the Trust over IP stack for building and maintaining digital trust, might actually be a path to putting the privacy genie back in the bottle—a feat that many people believed was impossible.

The Semantic Level: What DIDs Mean

The meaning of an address

Addresses do not exist on their own. They only exist in the context of a network that uses them. Whenever we have a new type of address, it is because we have a *new type of network* that needs this new address to do something that could not be done before. Table 8.7 shows this progression over the past few hundred years.

Table 8.7: An historical perspective on the evolution of addresses for different types of networks

Origin	Address Type	Network
Pre-history	Human name	Human networks (family, clan, tribe, etc.)
~1750	Postal address	Postal mail network
1879	Telephone number	Telephone network
1950	Credit card number	Payment network
1964	Fax number	Fax (facsimile) network
1971	Email address	Email network
1974	IP address	Internet (machine-friendly)
1983	Domain name	Internet (human-friendly)
1994	Persistent address (URN)	World Wide Web (machine-friendly)
1994	Web address (URL)	World Wide Web (human-friendly)
2003	Social network address	Social network
2009	Blockchain address	Blockchain or distributed ledger network
2016	DID	DID network (aka trust network)

So the real meaning of a DID comes down to what can be done with it on a **DID network**.

DID networks and trust networks

Just as everything on the Internet has an IP address—and everything on the Web has a URL—everything on a DID network has a DID. But that begs the question: *why* does everything on a DID network need a DID? What new communications network functionality do DIDs enable that could not be done before?

The short answer is that DIDs were invented to support both the *cryptographic trust* and the *human trust* required for the four-layer architecture of any **trust network** based on the **Trust over IP stack** introduced in Chapter 5 and shown again here.

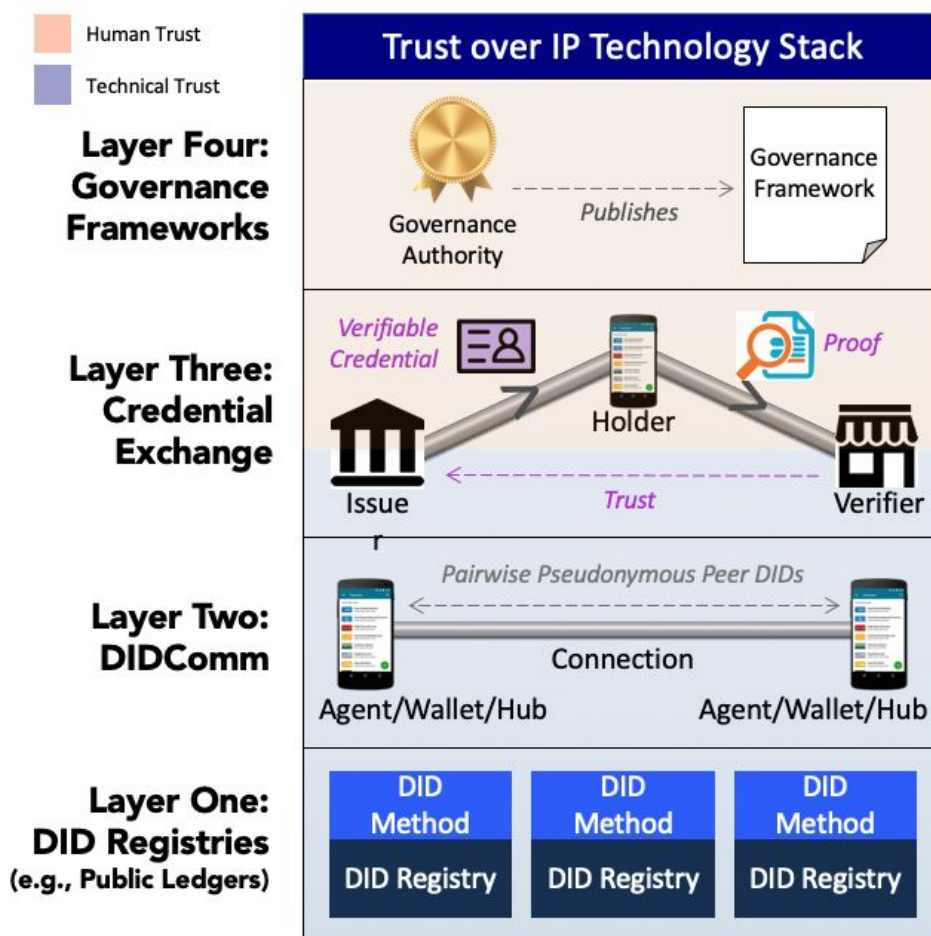


Figure 8.24: DIDs are foundational to all four levels of the Trust over IP stack

DIDs are essential to each layer of the stack as follows:

1. **Layer 1: DID Registries.** DIDs published on public blockchains like Bitcoin,

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

Ethereum or distributed ledgers as Sovrin, ION, Element, and Veres One—or on distributed file systems like IPFS—can serve as publicly-verifiable trust roots for participants at all higher layers. They literally form the foundation of a trust layer for the Internet.

2. **Layer 2: DIDComm.** By definition DIDComm is a P2P protocol between agents identified by DIDs. By default these are pairwise pseudonymous peer DIDs issued and exchanged following the Peer DID specification so they exist only at layer 2. However they can also be public DIDs from layer 1.
3. **Layer 3: Credential Exchange.** As covered in the verifiable credentials chapter, DIDs are integral to the process of issuing and verifying digitally-signed verifiable credentials as well as to the discovery of service endpoint URLs for credential exchange protocols.
4. **Layer 4: Governance Frameworks.** As will be covered in that chapter, DIDs are the “anchor points” for discovery and verification of governance authorities (as legal entities) and governance frameworks (as legal documents), as well as for every other participant role defined in a specific governance framework. DIDs also enable verifiable credentials to persistently reference the governance frameworks under which they are issued, and for governance frameworks to reference each other for interoperability.

In short, DIDs are the first widely-available, fully-standardized identifiers designed explicitly for building and maintaining digital trust networks that are protected by cryptography “all the way down”.

Why isn't a DID human-meaningful?

Many people have asked: if DIDs are the latest and greatest identifier for the cutting edge of communications on the Internet—then why aren't they more human-friendly? The answer lies in a conundrum called **Zooko's Triangle**²⁶ named after Zooko Wilcox-O'Hearn²⁷ who coined it in 2001. It is a trilemma that states that an identifier system can only achieve at most two of the following three properties:

1. **Human-meaningful:** All of the identifiers are semantic names (low-entropy) from ordinary human language.
2. **Secure:** Identifiers in the system are guaranteed to be unique, i.e. each identifier on is bound to only one specific entity.
3. **Decentralized:** Identifiers can be generated and correctly resolved to the identified entities without the use of a central authority or service.

²⁶ https://en.wikipedia.org/wiki/Zooko%27s_triangle

²⁷ Zooko worked in the 90s with famous cryptographer David Chaum developing DigiCash and also founded Zcash, a cryptocurrency aimed at using cryptography to provide enhanced privacy for its users. The academic cryptography, SSI and public blockchain space overlap on many levels as you read through his book.

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

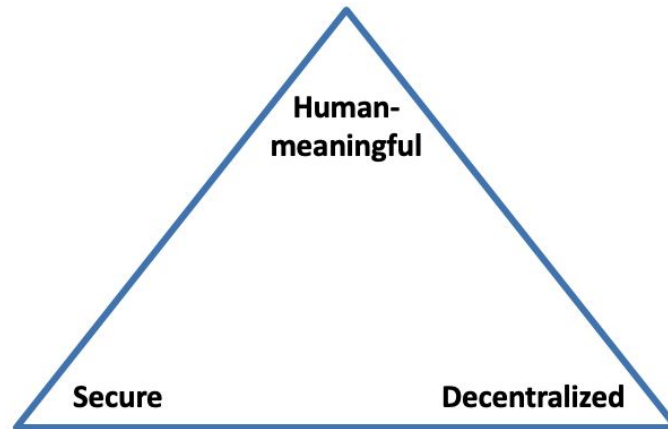


Figure 8.x: Zooko’s Triangle is a trilemma that proposes an addressing system can have at most two of these three properties

Although some believe Zooko’s Triangle can actually be solved, most Internet architects will agree it is far easier to achieve two of these three properties than all three. As this chapter makes clear, with DIDs the two properties chosen were **secure** and **decentralization** (the latter being built right into the acronym “DID”). What was given up—due to the cryptographic algorithms used to generate DIDs—was any attempt at being human-meaningful.

But the SSI community realized that while DIDs alone could not solve the human-meaningful naming problem, they could in fact anchor a promising new solution. The trick was not to do it at the public DID layer (layer 1 of the ToIP stack)—or at the peer DID layer (layer 2)—but *at the verifiable credentials layer* (layer 3). In other words, a specific class of verifiable credential could assert one or more **verifiable names** for a DID subject. By creating searchable **credential registries** for the names in these credentials, we could collectively build a naming layer that will be semantically richer, fairer, more trusted, and more distributed than the current DNS naming layer.

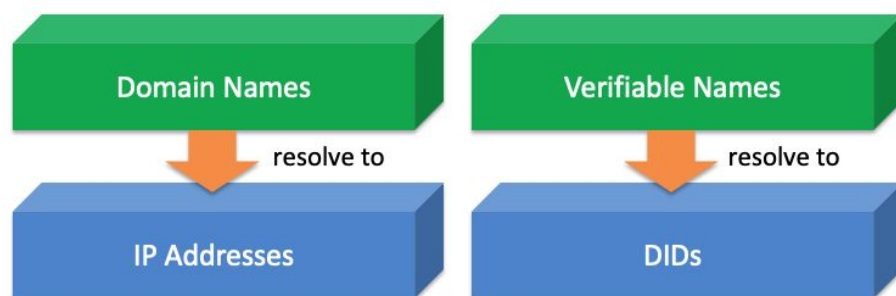


Figure 8.26: Verifiable credentials for the human-meaningful names of DID subjects can be layered over machine-friendly DIDs the same way human-meaningful DNS names were layered over machine-friendly IP addresses.

This **verifiable naming layer** would no longer need to be arbitrarily divided into top-level domain (TLD) name registries, but could capture the full richness of human name(s) in any

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://livebook.manning.com/#!/book/_____/discussion

language for any kind of DID subject: person, organization, product, concept, and so on. Furthermore, the authenticity of names—for people, for companies, for products—could be attested to in a decentralized way by issuers of all kinds so spoofing or phishing would become an order of magnitude harder than it is in today's Internet "Wild West".

What does a DID actually identify?

We saved this question for last because from a semantic standpoint it is the deepest. The easy answer is exactly what it says in the DID specification: "A DID always identifies the DID subject". And that is completely true, no matter what that subject may be: a person, organization, department, physical object, digital object, concept, software program—anything that has identity.

Where the confusion arises is with the DID document that a DID resolves to. Since a DID document is a real resource on the Web, and since it describes the DID subject, then isn't a DID document a representation of the DID subject? And if that is true, then *isn't a plain DID also a URL that identifies the DID document?*

This tips us into one of the oldest and longest debates in the Semantic Web—one so deep it has its own Wikipedia page called **HTTPRange-14**²⁸ (after the issue number it was assigned by the W3C Technical Architecture Board). The essence of the debate is that URIs may be used to identify resources on the Web ("information resources" such as Web pages or image files), or resources that are not on the Web ("non-information" resources such as people, places, or planets). A single URI however *can only identify exactly one resource*. So a DID by itself cannot at the same time serve as both an abstract identifier (URN) that identifies the DID subject and as a concrete identifier (URL) that identifies a DID document describing the DID subject.

The editors of the DID specification have proposed a solution to this dilemma, which is:

1. The DID by itself—without any other syntax allowed in a DID URL—always identifies the DID subject.
2. The DID by itself *plus a single slash ("/") character* (allowed in DID URL syntax as an "empty path") identifies the DID document.

So, for example, if the following DID was assigned to identify the planet Jupiter:

did:example:21tDAKCERh95uGgKbJNHyp

Then the following DID URL identifies the associated DID document:

did:example:21tDAKCERh95uGgKbJNHyp/

The plain DID contained within this DID URL *by itself still identifies the planet Jupiter*. It is the addition of the forward slash ("/") delimiter character that creates the DID URL that identifies the DID document as a separate resource on the Web.

This pattern of assigning different URIs to: a) a real-world entity, and b) a Web resource that describes it is well-known in the Semantic Web community. For example, it is used by

²⁸ <https://en.wikipedia.org/wiki/HTTPRange-14>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

the WebID standard²⁹ (part of the Solid project)³⁰ as well as by the semantic database project DBPedia. An example from the latter illustrates perfectly: here is the DBPedia URI that identifies the planet Jupiter *as an abstract resource*:

`https://dbpedia.org/resource/Jupiter`

And here is the URL that identifies *a Web page describing the planet Jupiter*.

`https://dbpedia.org/page/Jupiter`

Note that when the first URI (the one that identifies the planet) is dereferenced (e.g., put into a browser), the DBPedia site automatically redirects to the second URL. This way, it is possible to have separate, semantically clean identifiers for the abstract non-information resource and the concrete information resource while at the same time supporting the functionality developers expect—which is that if you start with an abstract identifier (in our case, a plain DID), you end up with a Web resource that describes it (in our case, a DID document).

While to some people this might seem like an academic debate akin to “how many angels can dance on the head of a pin”, in fact this level of semantic precision is very important when it comes to building a strong foundation for trust on the Internet. It underscores that DIDs are the **first digital identifiers designed for universal verifiable identification** of any entity of any kind anywhere.

Although it is one of the longest chapters in this book, if you’ve read it this far, you now have a much deeper understanding of why we call DIDs “the atomic building block of digital trust”. They are far more than just a new type of globally unique identifier. By relying on cryptography for both generation and verification, DIDs change the fundamental power dynamics of the Internet. Instead of scale laws pulling everything into the gravity well of the Internet giants at the center, DIDs pushes power to the very edges—to the individual digital agents and wallets where DIDs are born and DID-to-DID connections are made. This “Copernican inversion” creates the new spacetime of decentralization—a universe where entities of all kinds can be “self-sovereign” and interact as peers using all four layers of the Trust over IP stack.

For our next chapter we’ll move up to Layer Two of the stack to understand the digital wallets and digital agents required to generate DIDs, form DID-to-DID connections, exchange verifiable credentials, and handle decentralized key management.

²⁹ <https://www.w3.org/2005/Incubator/webid/spec/identity/>

³⁰ <https://solid.mit.edu/>

©Manning Publications Co. We welcome reader comments about anything in the manuscript — other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.