10

Decentralized key management

by Dr. Sam Smith

Chapter 9 covered the overall topic of self-sovereign identity (SSI) digital wallets and agents, but the function at the very core of digital wallets—cryptographic key management—is deep enough to merit its own chapter. Although thousands of papers and dozens of books have been written on the subject of key management, for this chapter on ***decentralized key management*** *we called on Dr. Sam Smith, who is not only one of the most prolific thinkers and authors in SSI (he is the author of Meta-Platforms and Cooperative Network-of-Network Effects in our Landmark Essays section), but he is the inventor of Key Event Receipt Infrastructure (KERI), covered in the final section of this chapter. Sam received his Ph.D. in Electrical and Computer Engineering from Brigham Young University in 1991, spent 10 years at Florida Atlantic University reaching full professor status, then retired to become a full-time entrepreneur and strategic consultant. He has over 100 refereed publications in the areas of machine learning, AI, autonomous vehicle systems, automated reasoning, blockchains, and decentralized systems.*

Chapter 9 began with this overarching definition of digital wallets:

> *A digital wallet consists of software (and optionally hardware) that enables the controller of the wallet to generate, store, manage, and protect cryptographic keys, secrets, and other sensitive private data.*

We followed that by saying that a digital wallet is the **nexus of control** for every actor in SSI. The essence of that control is **key management**. As the Wikipedia article on the subject states:[1]

> *Key management refers to management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use, crypto-shredding (destruction) and replacement of keys. It includes cryptographic protocol design, key servers, user procedures, and other relevant protocols.[1]*
>
> *Successful key management is critical to the security of a cryptosystem. It is the more challenging side of cryptography in a sense that it involves aspects of social engineering such as system policy, user training, organizational and departmental interactions, and coordination between all of these elements, in contrast to pure mathematical practices that can be automated.*

In this chapter we will cover:

- Why any form of digital key management is hard
- Standards and best practices for conventional key management
- The starting point for key management architectures: roots-of-trust
- The special challenges of *decentralized* key management
- The new tools that verifiable credentials (VCs), decentralized identifiers (DIDs), and self-sovereign identity (SSI) bring to decentralized key management
- Key management for ledger-based DID methods
- Key management for peer-based DID methods
- Fully autonomous decentralized key management with Key Event Receipt Infrastructure (KERI)

The final section on KERI is a special feature of this book as it summarizes the technical architecture of KERI, one of the most comprehensive solutions for decentralized cryptographic key management available at the time of publication.

Why any form of digital key management is hard

People new to cryptography and public/private key infrastructure often wonder why all the fuss about keys. Isn't managing digital keys similar to managing physical keys, of which we usually have a small set we personally control on a key ring or fob of some kind?

[1] https://en.wikipedia.org/wiki/Key_management

While there is clearly an analogy between physical and digital keys, in reality the differences are dramatic. To wit:

- **Digital keys can be stolen remotely.** Stealing a physical key requires having physical access to where the key is stored or who is carrying it. Digital keys that are not properly protected can be stolen remotely over a network. Even when they are well-guarded, digital keys can still be stolen using side-channel attacks[2] (but those are very hard to pull off).
- **You may not be able to tell if a digital key has been stolen.** A stolen physical key is easy to spot (unless the thief can quickly copy and replace it—a real challenge). But if an attacker is able to gain access to a digital key, they can copy it in milliseconds without you ever even knowing.
- **Digital locks are much harder to pick.** The alternative to stealing a physical key is just breaking the lock. For many real-world assets, like a car or a home, that is entirely feasible. Breaking a digital lock protected with strong encryption is nearly impossible.
- **The value that digital keys can unlock may be vastly greater than in the physical world**. Most assets protected by a physical key—a car, a house, a bank vault—have a value proportional to the strength of the physical security provided for the asset. But with digital assets, a single key could potentially unlock billions of dollars of value in the form of cryptocurrencies, digital fiat currencies, or some other form of digital asset.[3]
- **If lost or stolen, digital keys can be irreplaceable.** This is the real kicker. It is nearly impossible to protect a physical asset with enough physical security that it cannot be broken given enough time and money. But digital assets can be protected with encryption so strong (even quantum-proof) that in theory it can withstand all the computing power in the universe for the rest of time (or at least the next few millennia). So digital keys can be almost immeasurably more valuable than physical keys. In 2019, the Wall Street Journal estimated that one-fifth of all bitcoin is missing because the private keys have been irretrievably lost.[4] At the time we wrote these lines this was about $40 *billion.*[5]
- **With SSI, your digital keys will become the “keys to your digital life”.** It is hard to say that about your physical keys. Yes, they are important; they unlock your car, your house, your mailbox, your office, and your safe deposit box. But if you lost the whole set, it would only take you a few days or weeks to replace all of them. If you lost all the keys in a mature SSI digital wallet (and did not have a recovery method), it could put your digital life on hold for months.

[2] https://en.wikipedia.org/wiki/Side-channel_attack

[3] Anyone protecting high-value digital assets should be using multi-sig because its security scales much faster than the size of the digital asset. For example there are no reported cases of exploiting the Gnosis multi-sig Ethereum wallet despite some of those wallets holding billions in assets. https://gnosis-safe.io/

[4] https://www.wsj.com/articles/a-fifth-of-all-bitcoin-is-missing-these-crypto-hunters-can-help-1530798731

[5] The price of bitcoin was around $10.800 on the 15th of September 2020.

The bottom line: control over your digital keys—as well as the rest of the contents of your digital wallet—is probably the single most critical element of SSI architecture.

Standards and best practices for conventional key management

Thankfully digital key management is not new—we have decades of experience deploying it with conventional PKI—and, more recently, with cryptocurrency keys and wallets.

Moreover, since key management is fundamental to cybersecurity infrastructure, research bodies like the U.S. National Institute of Standards and Technology (NIST) have published extensive recommendations on the subject. Several of the best known from NIST are:

- **NIST Special Publication 800-130: A Framework for Designing Cryptographic Key Management Systems (CKMS)**[6]—a 112 page publication that provides a comprehensive guide to every topic in key management
- **NIST Special Publication 800-57: Recommendation for Key Management**[7]—a three-part series that NIST is constantly updating:
    - Part 1—General
    - Part 2—Best Practices for Key Management Organizations
    - Part 3—Application Specific Key Management Guidance

Following are examples of some of the guidelines from section 2 of NIST 800-57 Part 2:[8]

- *Because the compromise of a cryptographic key compromises all of the information and processes protected by that key, it is essential that client nodes be able to trust that keys and/or key components come from a trusted source, and that their confidentiality (if required) and integrity have been protected both in storage and in transit.*
- *In the case of secret keys, the exposure of a key by any member of a communicating group or on any link between any pair in that group compromises all of the information that was shared by the group using that key. As a result, it is important to avoid using a key from an unauthenticated source, to protect all keys and key components in transit, and to protect stored keys for as long as any information protected under those keys requires protection.*
- *Cryptographic confidentiality and integrity mechanisms are most commonly used to establish trust anchors that enforce trust policies and practices. A trust anchor is an authoritative entity for which trust is assumed and not derived. For example, in a public key infrastructure (PKI), a trust anchor is an authoritative entity represented by a public key and associated data.*[9]

---

[6] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-130.pdf
[7] https://csrc.nist.gov/projects/key-management/key-management-guidelines
[8] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf
[9] See the next section on trust anchors and root-of-trust. Reliance on administrative root-of-trust is one of the weaknesses of conventional public key infrastructure. This is an area where the NIST guidelines are not current.

- *Symmetric keys are often revoked by the use of Compromised Key Lists (CKLs). Certificate Revocation Lists (CRLs) are commonly used to revoke public key certificates, thus revoking the private key corresponding to the public key in the certificate. Regardless of whether symmetric or asymmetric keys are used, a means of revoking keys is required.*
- *A revoked key notification may include the revocation reason and an indication of when the revocation was requested. The inclusion of the revocation reason can be useful in risk decisions regarding the information that was received or stored using those keys.*
- *A key may also be suspended from use for a variety of reasons, such as an unknown status of the key or due to the key owner being temporarily unavailable (e.g., the key owner is on extended leave).*

Section 2.3.9 of the latest version of NIST 800-57 Part 2 includes this guidance about centralized vs. decentralized key management:

> *A CKMS can be either centralized or decentralized in nature. For a PKI, the public key does not require protection, so decentralized key management can work efficiently for both large-scale and small-scale cases. The management of symmetric keys, particularly for large-scale operations, often employs a centralized structure.*
>
> *Centralized CKMS key-management structures tend to be more structurally rigid than decentralized key-management structures, but the choice of how to establish keys, store and account for them, maintain an association of keys with the information protected under those keys, and dispose of keys that are no longer needed is a decision to be made by an organization's security management team.*

As you might expect, many different standards and protocols have been developed for different aspects of key management. For example, section 2.3.10 of NIST 800-57 Part 2 includes a list of 14 Requests for Comments (RFCs) for key management from the Internet Engineering Task Force (IETF).[10] NIST Special Publication 800-152 contains requirements for the design, implementation, or procurement of a CKMS meeting the standards of the U.S. Federal Government.[11] The Organization for Structured Information Standards (OASIS) has been developing the Key Management Interoperability Protocol (KMIP) since 2010.[12] It has become the industry standard for interoperability of centralized key management servers, which are typically deployed by an enterprise to standardize and automate key management across a large number of applications and services.

[10] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf
[11] https://doi.org/10.6028/NIST.SP.800-152
[12] https://en.wikipedia.org/wiki/Key_Management_Interoperability_Protocol

The starting point for key management architecture: roots-of-trust

Whether a key management architecture is centralized, federated, or decentralized, it all begins with a **root-of-trust** (aka **trust root** or **trust anchor**). The root-of-trust is the starting point in a **chain of trust**[13] because it is the only point in the chain where trust does not need to be *derived* (meaning verified by some means). Instead trust is *assumed* in the root-of-trust, i.e., verifiers simply accept axiomatically that the root-of-trust can be trusted.

In conventional PKI architecture, the root-of-trust is represented by a special digital certificate called the **root certificate**.[14] As the Wikipedia entry for *trust anchor*[15] explains:

> *In X.509 architecture, a root certificate would be the trust anchor from which the whole chain of trust is derived. The trust anchor [certificate] must be in the possession of the trusting party beforehand to make any further certificate path validation possible.*
>
> *Most operating systems provide a built-in list of self-signed root certificates to act as trust anchors for applications. The Firefox web browser also provides its own list of trust anchors. The end-user of an operating system or web browser is implicitly trusting in the correct operation of that software, and the software manufacturer in turn is delegating trust for certain cryptographic operations to the certificate authorities[16] responsible for the root certificates.*

The reason SSI represents such a sea change in key management is that it starts with a different set of assumptions about roots-of-trust as illustrated in Figure 10.1.

[13] https://en.wikipedia.org/wiki/Chain_of_trust
[14] https://en.wikipedia.org/wiki/Root_certificate
[15] https://en.wikipedia.org/wiki/Trust_anchor
[16] https://en.wikipedia.org/wiki/Certificate_authority

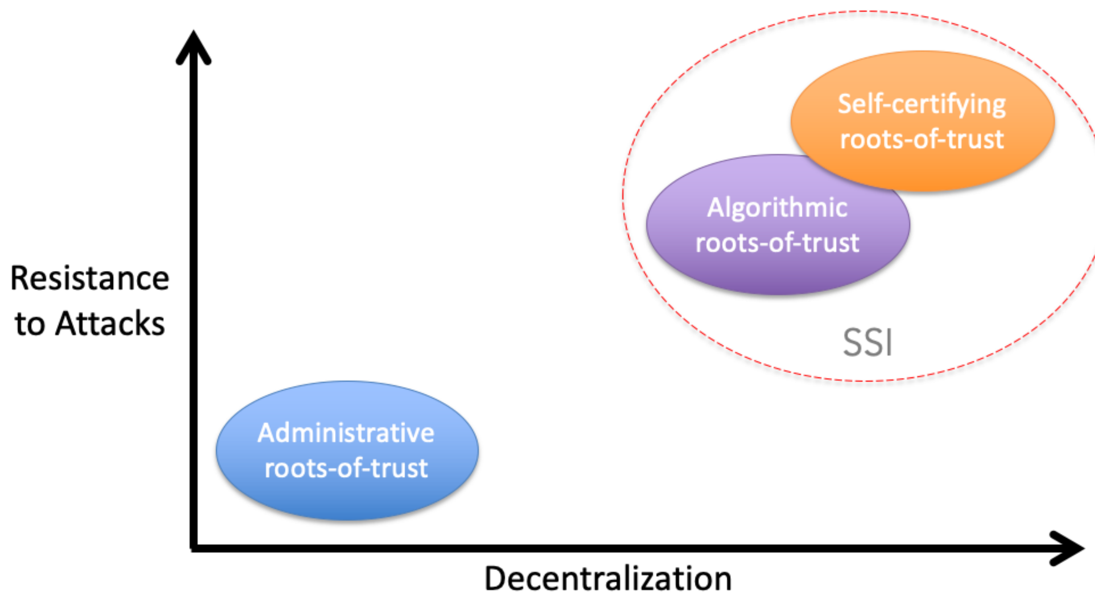

Self-certifying roots-of-trust
Algorithmic roots-of-trust
SSI
Resistance to Attacks
Administrative roots-of-trust
Decentralization

Figure 10.1: SSI starts with different assumptions about a root-of-trust—rather than administrative roots, SSI uses algorithmic or self-certifying roots

- **Administrative roots-of-trust** are what is used in conventional PKI—namely, certificate authorities (CAs) staffed by humans who follow rigorous procedures (Certification Practice Statements)[17] to ensure the quality and integrity of the digital certificates that they issue. Assumed trust in administrative roots-of-trust is based on the service provider's reputation as attested by industry certifications and accreditations.
- **Algorithmic roots-of-trust** are based on computer algorithms designed to create secure systems where no single party is in control yet all parties can agree on a shared source of truth. Blockchains, distributed ledgers, and distributed file systems like InterPlanetary File System (IPFS) are all examples of algorithmic roots-of-trust. (See Chapter 5 for a complete description about how different SSI architectures use different algorithmic roots-of-trust.) Although all algorithmic roots-of-trust are based on cryptography, the assumed trust requires more than that—it is based on the reputation of the system as a whole, e.g., the number and size of participants, the history of the project, how long the ledger has been running, whether there have been any security issues, and the prospect (or history) of forking. However as we explain in Chapter 15, there are plenty of disagreements about which of these approaches is most trustworthy.
- **Self-certifying roots-of-trust** are based solely on secure random number generation and cryptography. In the case of SSI, this means DIDs that can be generated using only a digital wallet. The most secure self-certifying roots-of-trust use special hardware such as a secure enclave or a trusted processing module (TPM)

[17] https://en.wikipedia.org/wiki/Certification_Practice_Statement

to generate and key pairs and store private keys. Assumed trust in self-certifying roots-of-trust is based on the specifications, testing, certification, and reputation of the hardware and software.

The reasons these distinctions are so important is summarized in Table 10.1:

Table 10.1: Summary of the differences between the three root-of-trust types

Property	Administrative root-of-trust	Algorithmic root-of-trust	Self-certifying root-of-trust
Centralized / single point of failure	Yes	**No**	**No**
Requires human involvement in validation	Yes	**No**	**No**
Requires the involvement of external parties	Yes	Yes	**No**

In short, the paradigm shift to SSI and decentralized key management is the shift from *administrative* roots-of-trust—which are inherently centralized and subject to human fallibility—to *algorithmic* and *self-certifying* roots-of-trust which can be partially or fully automated and decentralized. In fact the only difference between algorithmic and self-certifying roots-of-trust is the role of any third party at all (to be discussed later in this chapter).

The special challenges of decentralized key management

While decades of work have gone into centralized key management practices, decentralized key management is a much newer topic. It barely existed until the first version of Decentralized Identifiers (DIDs—see Chapter 8) was published as a community specification in December 2016. Since DIDs are both decentralized and cryptographically verifiable, they demanded a decentralized solution for managing the associated public/private keys. The growing interest in DIDs led the U.S. Department of Homeland Security (DHS) in 2017 to award a research contract on decentralized key management to SSI vendor Evernym.[18] As the announcement summarized:

> *Through a project titled "Applicability of Blockchain Technology to Privacy Respecting Identity Management," Evernym is developing a DKMS—a cryptographic key management approach used with blockchain and other distributed-ledger technologies—to boost online authentication and verification. Within a DKMS, the*

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

initial “root-of-trust” for all participants is a distributed ledger that supports a decentralized identifier—a new form of root identity record.

DKMS stands for “decentralized key management system” (in contrast to CKMS, “cryptographic key management system”). In the two-year research project, Evernym assembled a group of cryptographic engineers and key management experts to produce a document called *DKMS Design and Architecture* that was published as part of the Hyperledger Indy project at the Linux Foundation.[19] The introduction states:

> *DKMS (Decentralized Key Management System) is a new approach to cryptographic key management intended for use with blockchain and distributed ledger technologies (DLTs) where there are no centralized authorities. DKMS inverts a core assumption of conventional PKI (public key infrastructure) architecture, namely that public key certificates will be issued by centralized or federated certificate authorities (CAs).*

As stated in section 1.3 of this document, DKMS is designed to provide the following major benefits:

1. **No single point of failure.** Since DKMS uses either algorithmic or self-certifying roots-of-trust, there is no reliance on a central CA or other registration authority whose failure can jeopardize large swaths of users.

2. **Interoperability.** DKMS will enable any two identity owners and their applications to perform key exchange and create encrypted P2P connections without reliance on proprietary software, service providers, or federations.

3. **Portability.** With DKMS, users can avoid being locked into any specific implementation of a DKMS-compatible wallet, agent, or agency. Users should—with the appropriate security safeguards—be able to use the DKMS protocol itself to move the contents of their wallet (though not necessarily the actual cryptographic keys) between compliant DKMS implementations.

4. **Resilient trust infrastructure.** DKMS incorporates all the advantages of distributed ledger technology for decentralized access to cryptographically verifiable data. It then adds on top of it a distributed web of trust where any peer can exchange keys, form connections, and issue/accept verifiable credentials from any other peer.[20]

5. **Key recovery.** Rather than app-specific or domain-specific key recovery solutions, with DKMS robust key should be built recovery directly into the infrastructure, including agent-automated encrypted backup, DKMS key escrow services, and social recovery of keys—for example by backing up or sharding keys across trusted DKMS connections and agents.[21]

[19] http://bit.ly/dkms-v4
[20] The DKMS Design and Architecture document was published before the invention of KERI, however it is compatible with KERI’s fully decentralized key management architecture. See the final sections of this chapter.
[21] See Chapter 9 for more about the role of SSI digital wallets and agents in key recovery.

To provide these benefits, however, DKMS needs to address the following challenges:

1. **There cannot be any "higher authority" to fall back on.** It is surprising how much simpler you can make a system if you know you can ultimately fall back on a centralized authority. But with DKMS, there is no "password reset" option. If there was some outside authority to whom you can turn to to replace your keys, then that authority can always take away your keys—or their systems can be compromised to break into your keys. So a DKMS system must be designed to be failsafe for the key holders from the start.
2. **DKMS cannot come from a single company—or even a single consortia.** It must be based entirely on open standards that any open source project or commercial vendor can implement—much like the W3C Verifiable Credentials (VC) and Decentralized Identifier (DID) standards that are already foundational for SSI. This eliminates the proprietary approaches of some of the popular secure chat products today, e.g., Apple iMessage and Facebook Messenger.
3. **DKMS cannot dictate a single cryptographic algorithm or ciphersuite[22] that everyone must use.** Many problems can be solved by everyone agreeing on the same cryptography. But there are simply too many options—and the field is advancing too fast—for DKMS to lock into a single type of cryptography. DKMS must be able to accommodate the evolutionary advancement of cryptographic algorithms and protocols.
4. **DKMS key and wallet data must be portable across different technical implementations from different vendors.** As it is often said in SSI circles: "If it's not portable, it's not self-sovereign." Furthermore, portability must be proven against formal interoperability testing, not just marketing slogans.
5. **DKMS cannot assume any specialized knowledge or skills on behalf of end-users.** DKMS-enabled digital wallets and agents must be as easy—or easier—to use as modern browsers and email clients. Most of all, they cannot require end-users to understand anything at all about cryptography, blockchains, or SSI—or even the concept of public/private keys. It just needs to work—AND be secure.

Some developers reading these requirements might throw up their hands and say it cannot be done. However a growing community of architects, cryptographers, and usability experts are intent not just on doing it, but on baking the solution deep into the infrastructure of the Internet so it is available to everyone, just as email and the Web are today.

The new tools VCs, DIDs, and SSI bring to decentralized key management

SSI depends on decentralized key management but it also brings new tools to the table to enable it. In this section we will list the specific contributions from VCs (Chapter 7), DIDs (Chapter 8), and digital wallets and agents (Chapter 9).

[22] https://en.wikipedia.org/wiki/Cipher_suite

#1: Separating identity verification from public key verification

In addition to decentralized roots-of-trust, the primary innovation enabling DKMS is the ability of DIDs to separate the verification of the public key of a DID controller from verification of other identity attributes such as the controller's legal name, URL, address, government ID number, and so on. With conventional PKI, these two steps are bound together in the issuance of an X.509 digital certificate by a certificate authority (CA). This is shown in Figure 10.2 (taken from the in-depth explanation of how DIDs work in Chapter 8).

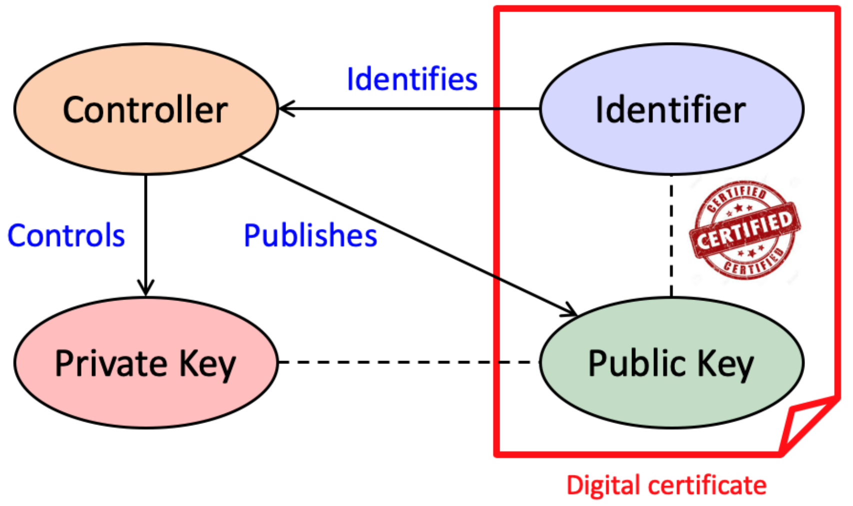


Figure 10.2: How conventional PKI-based digital certificates bind together identification of an entity with verification of the entity's public key

With SSI, a DID is generated from a public/private key pair itself using either an algorithmic root-of-trust or a self-certifying root-of-trust. This means the DID controller can always provide proof-of-control of their DID by using their own private key to digitally sign their own DID document as shown in Figure 10.3 (also taken from Chapter 8).

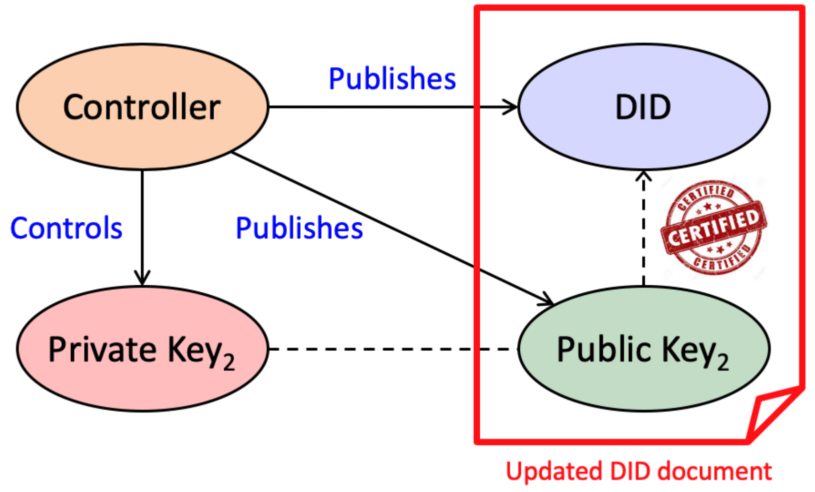


Figure 10.3: DIDs enable identity controllers to prove their own public keys without the use of an intermediary by digital signing their own DID documents

If the DID method uses a self-certifying root-of-trust, the key generation and rotation operations may take place entirely under the aegis of the DID controller—in their own digital wallet or some other key generation and signing system they control. If the DID method uses an algorithmic root-of-trust, a second step is needed: a transaction with an external verifiable data registry (VDR) such as a blockchain. However in both cases these steps can be performed automatically by the DID controller's agent without human intervention. By eliminating the need for a "human in the loop", there are two other major benefits: 1) the cost of these steps can drop to nearly zero, and 2) the scale at which DIDs can be generated and used increases dramatically. The combination removes any barrier to DID controllers having as many DIDs as they need.[23]

#2: Using VCs for proof of identity

If DIDs and DID documents can handle the challenge of key verification, that leaves VCs to do what they do best: convey third-party attestions of the real-world identity attributes of the DID controller. This is what a verifier needs to establish real-world business or social trust.

Furthermore, by separating identity verification from public key verification, the number and diversity of issuers for identity verification attributes should grow much larger. This gives both DID controllers and verifiers a wider range of choices and lowers the cost for everyone.

#3: Automatic key rotation

Another core key management problem that DIDs help solve—as explained in depth in Chapter 8—is automated key rotation. Because the DID is itself an immutable identifier, all DID methods (except a special category called "static") define how the DID controller can change the public/private key pair(s) associated with the DID by publishing an update to the associated DID document. The specifics of how this is done varies with different DID methods, but they all follow the same principle, which is that the DID controller can accomplish key rotation without the need to rely on any external administrator.

#4: Automatic encrypted backup with both offline and social recovery methods

The fact that SSI digital wallets cannot appeal to a higher authority to "reset the password" or "replace the keys" means that backup and recovery must be built directly into the infrastructure. This can be accomplished by building backup and recovery functions directly into digital wallets and agents as described in Chapter 9, or by leveraging special key recovery capabilities designed into specific DID methods—or both. See the final section of

[23] See Chapter 8 on DIDs for more about how random number generation and cryptographic algorithms enable an almost infinite number of public/private key pairs and DIDs to be created without collusion—this is the essence of what enables decentralization.

this chapter for the sophisticated decentralized key recovery architecture built into KERI.

#5: Digital guardianship

With centralized key management systems, key servers operated by corporations or governments can serve a wide population of users with different levels of capabilities. With decentralized key management, a solution needs to cover individuals who lack the physical, mental, or economic capability to operate their own devices and manage their own keys. This critical aspect of SSI infrastructure is referred to as **digital guardianship**, a topic covered in more detail in Chapter 11 on SSI Governance Frameworks.

Ironically, from a high level, digital guardians can look very much like centralized key management systems. Under the hood, however, they are very different. Guardians typically host individual cloud wallets for each person depending on them, called the **dependent**. Guardians are usually issued **guardianship credentials** by an official authority to authorize their guardianship role; they in turn issue **delegation credentials** to staff persons or contractors to authorize their actions. Lastly, digital guardians usually operate under a governance framework that places strict legal requirements on their roles as information fiduciaries.[24]

Since any person or organization can operate as a digital guardian—and digital guardianship uses the same open standards and infrastructure as the rest of SSI—it extends the ability to control and manage digital keys to those who otherwise could not do it on their own.

Key management with ledger-based DID methods (algorithmic roots-of-trust)

All DID methods rely on a root-of-trust as shown in Figure 10.1—a starting point for proving the chain of trust based on a public/private key pair. Although the key pair itself is usually generated in secure hardware using a long random number,[25] most DID methods do not rely on this root-of-trust alone (i.e., they are not *self-certifying*). They require a second step: using the private key to digitally sign a transaction in a distributed ledger or blockchain to "record" the DID and the initial associated public key. Once that record is created, the ledger becomes the *algorithmic* root-of-trust for the DID.

This means verifiers must check with the ledger in order to verify the current public key and any other contents of the DID document associated with the DID. In other words, verifiers must trust:

1. The consensus algorithm and the operation of the particular ledger, i.e., its ability to withstand a 51% attack[26] or any other form of corruption or attack.
2. The security of the resolver used to access records on the ledger.

[24] See *Information Fiduciaries and the First Amendment*. https://lawreview.law.ucdavis.edu/issues/49/4/Lecture/49-4_Balkin.pdf
[25] See https://tools.ietf.org/html/rfc4086 for how to do this securely
[26] https://www.investopedia.com/terms/1/51-attack.asp

3. The genesis records used by the resolver (or the verifier) to verify resolution results.

Given the success of large, well-established public blockchains like Bitcoin and Ethereum, together with the well-known mechanisms for verifying lookups from those ledgers, these are widely considered to be strong algorithmic roots-of-trust. In addition, for many DIDs, it is desirable for them to be publicly resolvable and verifiable. Thus it is no surprise that, as of August 2020, 95% of the 63 DID methods registered in the W3C DID Specification Registry use DID methods based on an algorithmic root-of-trust, i.e., some type of public ledger.[27]

But ledger-based DID methods also have several downsides:

1. **Dependency on another party or network.** Although the ultimate root-of-trust is still the key pair used to generate the DID and update the DID document on the ledger, a ledger-based DID method requires a DID controller to depend on a distributed ledger and its associated governance mechanisms to be trustworthy. To the extent a DID controller can count on the ledger to be incorruptible and always available, that risk may be small, but it is still non-zero. For example, all distributed ledgers are subject to 51% attacks, forking, or changes in their governance or regulatory status.
2. **Non-portability ("ledger-lock").** Ledger-based DIDs are "locked" to a specific ledger and cannot be moved if problems develop with the ledger or its governance—or if the DID controller desires to use other DID methods.
3. **Potential conflicts with the GDPR "right-to-be-forgotten".** While not an issue with DIDs meant for use by organizations (or things), DIDs and public keys for personal use are considered "personal data" under the EU General Data Protection Regulation (GDPR) and thus subject to the **right of erasure**, popularly known as "the right to be forgotten". This can be a serious issue for immutable public ledgers. Because these ledgers co-mingle transactions from all users, transactions for a given DID may not be removable without destroying the integrity of the ledger for all the other users.[28]

Key management with peer-based DID methods (self-certifying roots-of-trust)

Once DIDs and DKMS started to catch on, it was not long until some security architects realized that, while ledger-based DIDs have many advantages, the use of a ledger is not technically required to gain the benefits of DIDs. Given that the ultimate root of trust is the long random number upon which a public/private key pair is based—and that this root-of-trust exists only in the DID controller's digital wallet—these architects saw that for many scenarios, DIDs and DID documents that were *self-certifying* could be generated entirely

[27] https://w3c-ccg.github.io/did-method-registry/

[28] For an in-depth discussion of this problem, see the Sovrin Foundation white paper *Innovation Meets Compliance: Data Privacy Regulation and Distributed Ledger Technology*. https://sovrin.org/data-protection/

within a digital wallet and exchanged directly—peer-to-peer.[29]

This led to the development of the `did:peer:` method defined by the *Peer DID Method Specification*,[30] first published by Daniel Hardman in 2018. As of August 2020, this specification had 15 contributing authors and had moved to the *Identifier and Discovery Working Group* at the Decentralized Identity Foundation for further standardization.[31] To quote from the Overview:

> *Most documentation about decentralized identifiers (DIDs) describes them as identifiers that are rooted in a public source of truth like a blockchain, a database, a distributed filesystem, or similar. This publicness lets arbitrary parties resolve the DIDs to an endpoint and keys. It is an important feature for many use cases.*
>
> *However, the vast majority of relationships between people, organizations, and things have simpler requirements. When Alice (Corp|Device) and Bob want to interact, there are exactly and only 2 parties in the world who should care: Alice and Bob. Instead of arbitrary parties needing to resolve their DIDs, only Alice and Bob do. Peer DIDs are perfect in these cases. In many ways, peer DIDs are to public, blockchain-based DIDs what Ethereum Plasma or state channels are to on-chain smart contracts — or what Bitcoin's Lightning Network is to on-chain cryptopayments. They move the bulk of interactions off-chain, but offer options to connect back to a chain-based ecosystem as needed. Peer DIDs create the conditions for people, organizations and things to have full control of their end of the digital relationships they sustain.*

The specification goes on to list these benefits of peer DIDs:

- *They have no transaction costs, making them essentially free to create, store, and maintain.*
- *They scale and perform entirely as a function of participants, not with any central system's capacity.*
- *Because they are not persisted in any central system, there is no trove to protect.*
- *Because only the parties to a given relationship know them, concerns about personal data and privacy regulations due to third-party data controllers or processors are much reduced.*
- *Because they are not beholden to any particular blockchain, they have minimal political or technical baggage.*
- *They can be mapped into the namespaces of other DID ecosystems, allowing a peer DID to have predictable meaning in 1 or more other blockchains (see Grafting). This creates an interoperability bridge and solves a problem with blockchain forks fighting over the ownership of a DID.*

---

[29] Self-certifying identifiers (SCIDs) will be explained further in the following section.
[30] https://identity.foundation/peer-did-method-spec/index.html
[31] https://identity.foundation/working-groups/identifiers-discovery.html

- *Because they avoid a dependence on a central source of truth, peer DIDs free themselves of the often-online requirement that typifies most other DID methods, and are thus well suited to use cases that need a decentralized peer-oriented architecture. Peer DIDs can be created and maintained for an entire lifecycle without any reliance on the internet, with no degradation of trust. They thus align closely with the ethos and the architectural mindset of the local-first*[32] *and offline-first*[33] *software movements.*

Key rotation and key recovery with peer DIDs are a matter of each peer, as the controller of their own peer DID, communicating updates to their peer DID document to the other peer. This is the purpose of the **peer DID protocol** defined in section 4 of the Peer DID Method Specification. It defines the standard four DID CRUD (create, read, update and deactivate) operations peers must perform:

1. **Create / register** peer DIDs and DID documents with each other.
2. **Read /resolve** peer DIDs.
3. **Update** peer DID documents for key rotation, service endpoint migration, or other changes.
4. **Deactivate** a peer DID to end a peer relationship.

Peer DIDs bypass the need for an algorithmic root-of-trust because they are based directly on the self-certifying root-of-trust used to generate the initial key pair without having to rely on a network. Since any well-designed SSI digital wallet can provide this function—and protect the resulting private key—peer DIDs eliminate the need for any external dependencies. They are fully "portable" and can be as decentralized and scalable as the Internet itself (if not more so).[34] This design also favors censorship resistance, an attribute highly valued by many in the decentralized technology community.

The only downside is that peer DIDs are not publicly discoverable and resolvable. But what if there was a DID method that relied only on a self-certifying root-of-trust, yet provided the best of both worlds: publicly discoverable/resolvable DIDs *and* peer DIDs?

[32] https://www.inkandswitch.com/local-first.html
[33] http://offlinefirst.org/
[34] The Internet's TCP/IP protocol still relies on federated identifiers (IP addresses) and routing tables that ultimately have a centralized root managed by ICANN. Peer DIDs have no central root.

Fully autonomous decentralized key management with Key Event Receipt Infrastructure (KERI)

As Figure 10.1 illustrates, the ideal root-of-trust from a security point of view—superior to both administrative and algorithmic—is a *self-certifying* root-of-trust that does not have to rely on a network. Properly implemented, it is both the most decentralized and the most resistant to attack. Every DID controller's wallet can serve as their own self-certifying root-of-trust, and these wallets can live anywhere on the network—ideally on edge devices, where they are hardest to attack remotely.

The Peer DID method (discussed in the previous section) applies this architecture by using a simple type of **self-certifying identifier** (SCID). A SCID is derived from a public/private key pair using one or more applications of cryptographic one-way functions.[35] The SCID is now bound to that key pair, and only the holder of the private key can prove control of the SCID.

Many other blockchains use this same technique—for example this is how a Bitcoin user proves control of a Bitcoin address. The difference is that SCIDs do not require a blockchain—or any other infrastructure—to verify the binding with the public key. Anyone with the SCID and the public key can verify the binding using cryptography alone. This is what is meant by *self-certifying*.

The next step is to *make the entire DID method self-certifying*, i.e., not just the initial SCID, but all key rotations after that. That was the inspiration for **Key Event Receipt Infrastructure (KERI).** In KERI architecture, **the history of all uses or changes to the public/private key pair can be compiled** to enable universal self-certifying proofs of the binding between the SCID and the associated public/private key pairs. With KERI architecture, the SCID is agnostic about where it is registered or discovered—it is completely portable and can form the root of a namespace for other SCIDs from the same controller.[36]

As far as we know, KERI is the first identifier and key management system to propose completely autonomous, portable, cryptographically verifiable identifiers that can be as public or private as required. While this might sound similar to the original vision for Pretty Good Privacy (PGP),[37] PGP key sharing infrastructure needed to be manually set up and maintained by humans via key signing parties.[38] Key rotation was a manual, laborious (and error-prone) process as well.[39]

With KERI, we can finally—25 years later—achieve Phil Zimmermann's vision by leveraging the distributed computing infrastructure and blockchain-inspired cryptographic engineering.

---

[35] See Chapter 6.

[36] See the following sections for details of how this works.

[37] https://en.wikipedia.org/wiki/Pretty_Good_Privacy

[38] https://en.wikipedia.org/wiki/Key_signing_party

[39] https://www.vice.com/en_us/article/vvbw9a/even-the-inventor-of-pgp-doesnt-use-pgp

KERI aims to be the first decentralized key management architecture that can be adapted to any underlying digital wallet or key management server and can be interoperable across all of them. A DID method based on KERI inherits all these features. So from the standpoint of SSI, KERI is positioned to deliver the greatest degree of self-sovereignty of all of the options for DIDs and DKMS, including anything rooted on shared ledgers.

In this section we will explain the basics of KERI architecture by way of summarizing its seven major benefits. For the full technical details, please refer to the KERI technical paper.[40]

#1: Self-certifying identifiers as a root-of-trust

As explained above, the starting point for KERI architecture is **self-certifying identifiers** (SCIDs). The KERI technical paper defines a SCID as:

> *A self-certifying identifier cryptographically binds an identifier to a key-pair.*

SCIDs were introduced in Chapter 8 on DIDs because they are the subclass of DIDs that depend exclusively on a self-certifying root-of-trust—they do not require either an administrative or an algorithmic root-of-trust. The basic concept is illustrated in this diagram from Chapter 8:

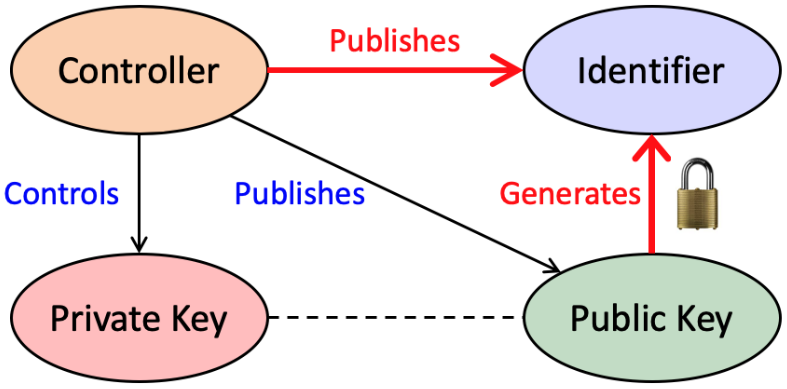


Figure 10.4: Self-certifying identifiers (SCIDs) generate the identifier for the controller directly from the public/private key pair without the need for any external administrator or algorithmic root-of-trust

The identifier is self-certifying because, given the associated public key, anyone can instantly verify that the identifier was generated from the public/private key pair using a one-way function such as a hash function. The diagram on the left side of Figure 10.5 (from the KERI technical paper) shows how the controller starts the process by instructing the digital wallet to generate a large random number (using a secure source of entropy as

[40] KERI is defined in a 138 page technical white paper that is being standardized by the Identifier and Discovery Working Group at the Decentralized Identity Foundation.

described in IETF RFC 4086).[41] Then the digital wallet generates a cryptographic key pair. Finally, the digital wallet derives the identifier (SCID) from the key pair. The result is a SCID whose binding with the public key can be verified instantly using cryptography alone—no need for a ledger, administrator, or any other external source of truth.

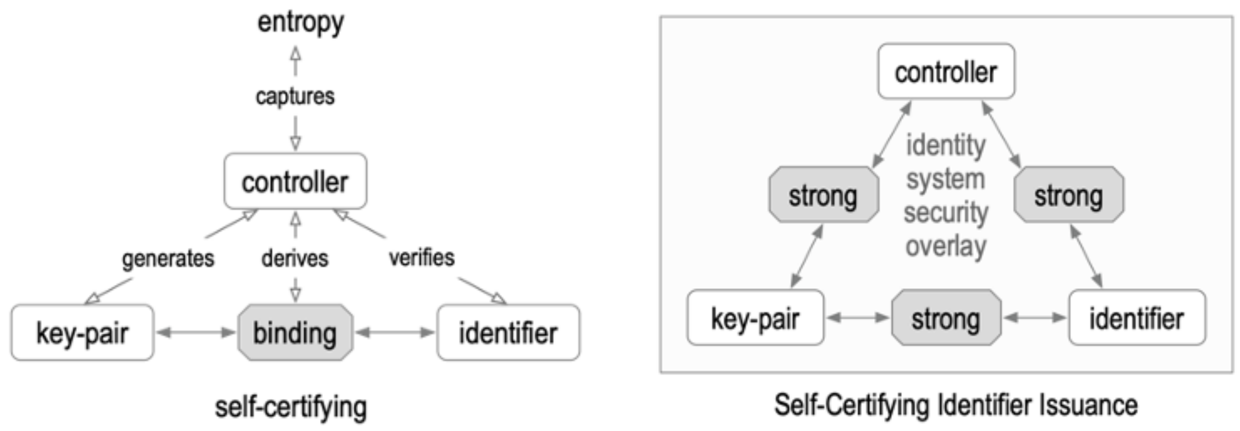


Figure 10.5: The process for generating a SCID (left) and the resulting bindings between the controller, the cryptographic key pair, and the SCID (right)

SCIDs are 100% portable identifiers[42] because the controller can "take them anywhere" and prove control without a verifier needing to trust anything but the cryptography and the security of the controller's digital wallet.

Because all other capabilities in KERI depend on the integrity and strength of SCIDs, the KERI technical paper defines several specific subtypes of SCIDs (basic, self-addressing, multi-sign self-addressing, delegated self-addressing, and self-signing) together with their syntactic structure, derivation code, inception statements, and generation algorithms inside a self-certifying root-of-trust.[43]

#2: Self-certifying key event logs

KERI takes its name not from the SCIDs at the heart of the architecture, but from how it handles one of the hardest problems in decentralized key management: key rotation and recovery. This approach is summarized in this sentence from page 29 of the KERI technical paper:

> *[KERI] leverages the fact that only the controller of the private key may create and order events that perform verifiable operations on the keys. As long as one complete verifiable copy of the event history is preserved, the provenance of control authority*

[41] See https://tools.ietf.org/html/rfc4086 for how to do this securely.
[42] Peer DIDs are a subclass of SCIDs that share this same benefit of full portability.
[43] See pages 8 - 20 of the KERI technical paper.

may be established.

With KERI, every rotation to the key pair associated with a SCID generates a new **key event**. The KERI protocol dictates the exact structure of a **key event message**. Every key event message includes a sequence number. Every key event message except the very first one (the **inception event**) also includes a digest (hash) of the previous key event message. The controller then digitally signs the new key event message with the new private key, producing a **key event receipt**.

The result is an ordered sequence ("chain") of key event receipts called a **key event log** that anyone can verify in much the same way they verify the sequence of transactions on a blockchain—but without needing an algorithmic root-of-trust. Figure 10.6 from page 40 of the KERI technical paper illustrates a sequence of key event messages in a key event log:

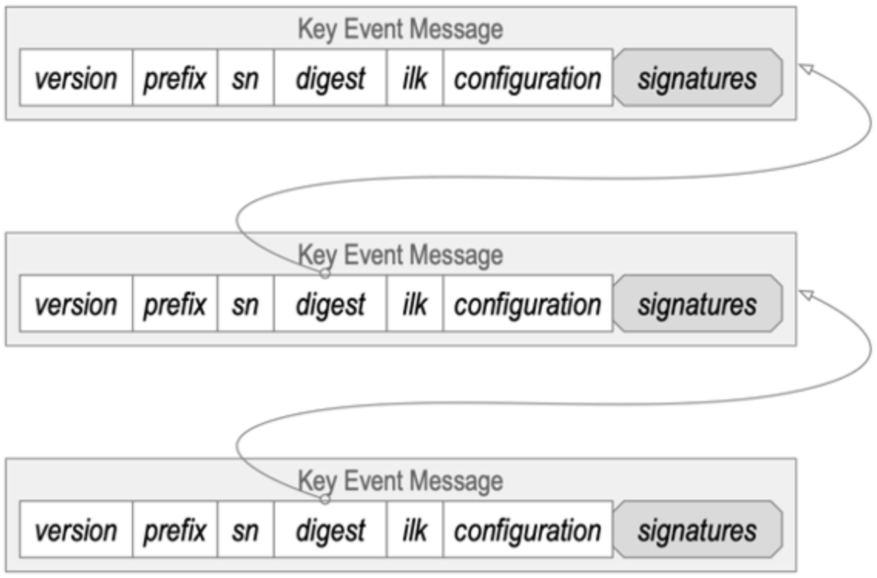


Figure 10.6: Each key event message in a key event log (except the first one) includes a sequence number and a digest of the previous key event message, creating a tamper-proof ordered sequence similar to a blockchain but without needing an algorithmic root-of-trust

#3: Witnesses for key event logs

One of KERI's primary innovations is that *other parties besides the controller can also digitally sign key event messages*. These parties are called **witnesses** because they are "witnessing" the controller's digital signature on a key event message just like they would witness a person's physical signature on a paper document (as is often legally required for high-value documents like wills or mortgages).

As shown in Figure 10.7, the KERI protocol standardizes how witnesses can receive key event messages from the controller and, if the witness can verify the key event message, the witness can then add their own signature to create their own independent copy of the key event log.

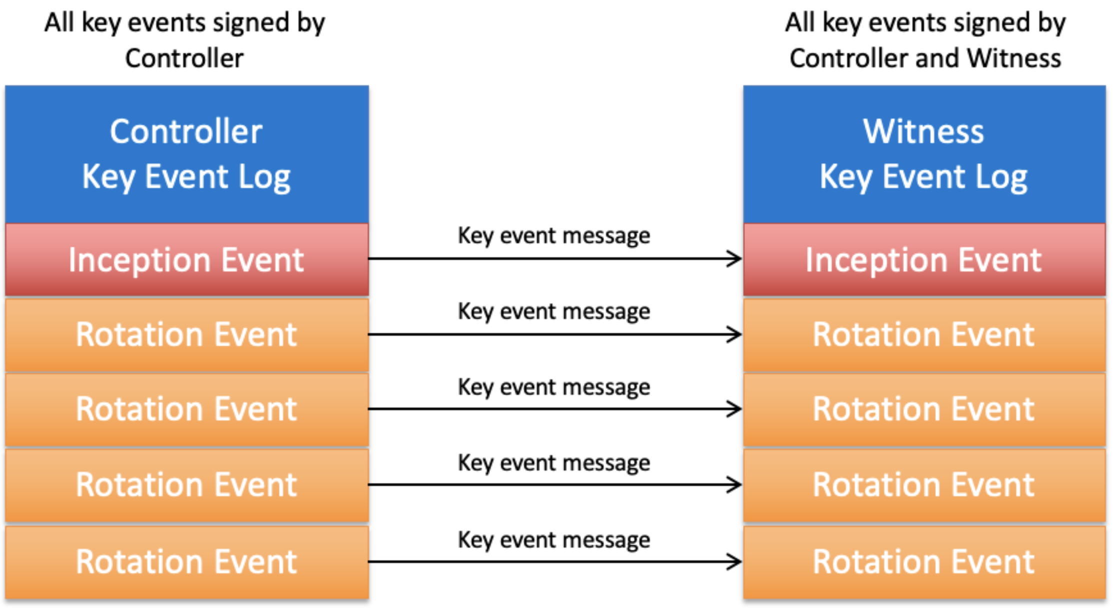


Figure 10.7: Witnesses increase the trustworthiness of a KERI key event log by: a) adding their own digital signature to each event, and b) maintaining their own independent copy of the log

Each witness becomes a **secondary root-of-trust** to the controller's primary self-certifying root-of-trust. To the extent a witness is trusted by a verifier to serve as an independent source of truth about key event messages, each additional witness increases the trustworthiness of the key event log. Again, this works just like human witnesses to the "wet ink" signing of a physical document. If you have the signature of one witness attesting when and where they saw the signer sign the document, that's good. If you have the signature of four witnesses saying when and where they saw the signer sign the document, that's better.

#4: Pre-rotation as simple, safe, scalable protection against key compromise

The challenge all key management systems must answer is not just how to rotate keys, but how to protect against the compromise of a private key—via any of the myriad ways that can happen:

- Lost or stolen device
- Security flaw in the self-certifying root-of-trust (digital wallet)
- Side-channel attack[44] on the self-certifying root-of-trust
- Social engineering attack on the controller

[44] https://en.wikipedia.org/wiki/Side-channel_attack

- Extortion attack on the controller ("rubber-hose cryptanalysis")[45]

Private key compromise is even more dangerous in decentralized key management because there is no "higher authority" than the controller of the keys. So losing control of a private key is tantamount to handing over control of all DIDs or SCIDs that depend on that private key.

For this reason, KERI builds protection against compromised private keys right into the heart of the architecture using a technique called **pre-rotation**. In short, starting with the inception event and continuing in every key rotation event, the controller publishes not only the new public key but a **cryptographic commitment[46] to the next public key** (called the **pre-rotated public key**). This commitment is in the form of a digest (cryptographic hash function—see Chapter 6) of the pre-rotated public key. This digest is included in the key event message establishing the new current public key as shown in Figure 10.8.

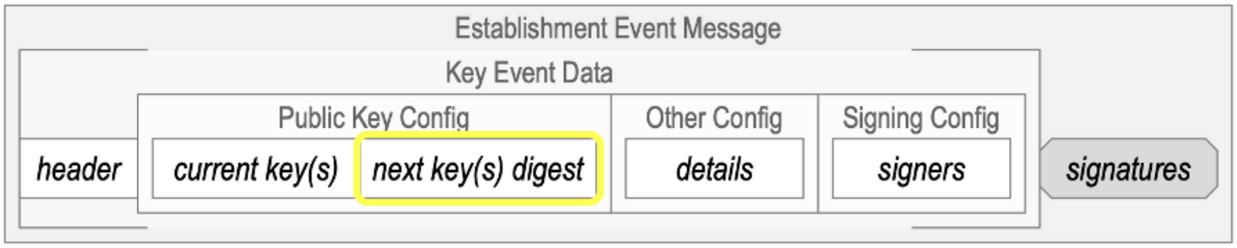


Figure 10.8: KERI uses pre-rotation of key pairs to protect against compromise of private keys

Pre-rotation enables the controller to have pre-established an entirely new and different public/private key pair for the next key rotation event. This means that an attacker who compromises the current private key will not have any ability to "take over" the SCID by rotating to a new public key because the next public key has already been committed to.

Indeed, the only way the attacker can take over the SCID is *to steal the pre-rotated private key*. But a number factors make this extremely difficult:

1. **The attacker does not even know what the pre-rotated public key is** because all that has been published is its digest.
2. **The pre-rotated key pair does not need to be exposed in any signing operations** until the next key rotation event.
3. **The pre-rotated key pair can be safely stored offline ("air-gapped") under very high security** because it is not needed until it is placed into active service after the next key rotation event.
4. **Each pre-rotated key pair can be used to safely generate the next one** prior

[45] Although the origin of this term is tongue-in-cheek, this attack vector is deadly serious because in most cryptosystems, the human user is the weakest link. See https://en.wikipedia.org/wiki/Rubber-hose_cryptanalysis
[46] https://en.wikipedia.org/wiki/Commitment_scheme

to going into active service.
5. **Pre-rotation can even be quantum secure** as long as the digest function uses a quantum-secure cryptographic hash function.

Section 9.3.1 on page 62 of the KERI technical paper summarizes why this pre-rotation architecture is so secure:

> *For many exploits, the likelihood of success is a function of exposure to continued monitoring or probing. Narrowly restricting the exposure opportunities for exploits in terms of time, place, and method, especially if the time and place happens only once, makes exploits extremely difficult. The exploiter has to either predict the one time and place of that exposure or has to have continuous universal monitoring of all exposures. By declaring the very first pre-rotation in the inception event, the window for its exploit is as narrow as possible. Likewise, each subsequent rotation event is a one-time-and-place signing exposure of the former (pre-rotated) rotation key.*

But if an attacker compromised an existing private key, could they not immediately publish their own conflicting key event message asserting a new pre-rotated key pair for which the attacker controls the private key? **Not if the controller already has one or more witnesses for the controller's earlier key rotation event message.** Those witnesses will recognize the duplicate sequence number and reject the attacker's later key rotation event message (and ideally notify the controller of a potential private key compromise).

What if the controller itself is malicious? Couldn't the controller publish two conflicting key rotation event messages, each with the same sequence number and timestamp but with two different pre-rotated key pair digests? Again, the witnesses (or any verifiers) could see these duplicitous events and flag the SCID as no longer being trustworthy.

The power of pre-rotation may be better understood after contrasting it with the use of hierarchically-derived keys. Many cryptocurrency wallets begin with the generation of a random seed. This seed is then used to derive all of the public/private keys pairs controlled by the wallet, and the value of the seed grows as more keys are derived from it. The seed must be stored securely, since compromise of the seed also results in compromise of every derived keypair. KERI inverts this process with pre-rotation. Instead of storing the root seed and needing to protect it forever, pre-rotation creates the next key pair, which must be stored securely only until it is time to use it.

Pre-rotation is a powerful key management security technique. For a deeper explanation, please see the KERI technical paper (Section 9, pages 58 - 74).

#5: System-independent validation ("ambient verifiability")

DID methods that rely on an algorithmic root-of-trust, such as a distributed ledger, produce DIDs that can only be verified by reference to that root-of-trust. The KERI technical paper refers to this dependency as "**ledger-lock**". Such DIDs are not portable to another source of verification, i.e., a different distributed ledger, a distributed file system, a centralized registry, a peer-to-peer protocol, or any other potential source of truth.

By contrast, KERI depends exclusively on a *self-certifying root-of-trust*—the controller's digital wallet. So KERI SCIDs and key event logs are self-verifying. All that is required is a copy of the complete key event log from *any* potential source—the controller itself, or any witness to whom a verifier has access. As stated on page 12 of the KERI technical paper:

> *[The key event log] is **end verifiable**. This means that the log may be verified by any end user that receives a copy. No trust in intervening infrastructure is needed to verify the log and validate the chain of transfers and thereby establish the current control authority. Because any copy of the record or log of transfer statements is sufficient, any infrastructure providing a copy is replaceable by any other infrastructure that provides a copy, that is, any infrastructure may do...*
> *This enables the use of **ambient infrastructure** to provide a copy of the log. The combination of end verifiable logs served by ambient infrastructure enables **ambient verifiability**, that is, anyone can verify anywhere at any time.*

This results in a very robust, flexible, decentralized infrastructure where every controller can choose the witnesses they feel are needed to provide the level of assurance required by verifiers in any particular context. It not only frees DID Methods based on KERI SCIDs and key event logs from "ledger lock", but it frees issuers and verifiers from needing to agree on the governance of a verifiable data registry (VDR) such as a blockchain or distributed ledger. The KERI technical paper summarizes this "separation of control" on page 81:

> *...the design principle of separating the loci-of-control between controllers and validators removes one of the major drawbacks of total ordered distributed consensus algorithms, that is, shared governance over the pool of nodes that provide the consensus algorithm. Removing the constraint of forced shared governance allows each party, controller and validator, to select the level of security, availability, performance specific to their needs. The validator's confirmation service may be further enhanced to provide duplicity detection and other protections as needed.*

From the standpoint of the ToIP four-layer architecture introduced in Chapter 2, this means both governance and technology for Layer 1 public utilities can be simpler, faster, less expensive, and more general-purpose.

The KERI technical paper goes into great depth on the protocol, configuration, and operation of KERI ambient verifiability infrastructure. See section 10 on Protocol Operational Modes (pages 74-83), section 11 on KERI Agreement Algorithm for Control Establishment (pages 83-100), and section 12 on Event Semantics and Syntax (pages 100-112).

#6: Delegated self-certifying identifiers for enterprise-class key management

For personal use, an individual should be able to generate and manage as many SCIDs as needed in their digital wallet. But when we graduate to enterprise usage, the scale and complexity of key management increases dramatically. As discussed throughout this book, enterprises need to be able to easily yet safely delegate use of DIDs, VCs, and the attendant key management to directors, officers, employees, contractors, and anyone else taking actions on behalf of the organization.

This **delegated key management** enables the organization to "tree out" from its own self-certifying root-of-trust to establish "subroots" for delegates, where each serves as its own self-certifying root-of-trust. To quote from page 48 of the KERI technical paper:

> *A common use case would be to delegate signing authority to a new identifier. The signing authority may be exercised by a sequence of revocable signing keys distinct from the keys used for the root identifier. This enables horizontal scalability of signing operations. The delegation operation may also authorize the delegated identifier to make delegations of its own. This enables a hierarchy of delegated identifiers that may provide a generic architecture for decentralized key management infrastructure (DKMI). When applied recursively, delegation may be used to compose arbitrarily complex trees of hierarchical (delegative) verifiable key event streams.*

In the KERI protocol, delegation may be performed using a **key interaction event**, so-named because it does not involve the inception or rotation of the primary SCID, but rather is used to perform operations that do not affect the establishment of control authority of the primary SCID. In this case a key interaction event is used to authorize the inception or rotation of a delegated SCID. Figure 10.9 depicts a key interaction event message containing the "delegation seal" for a new delegated SCID.

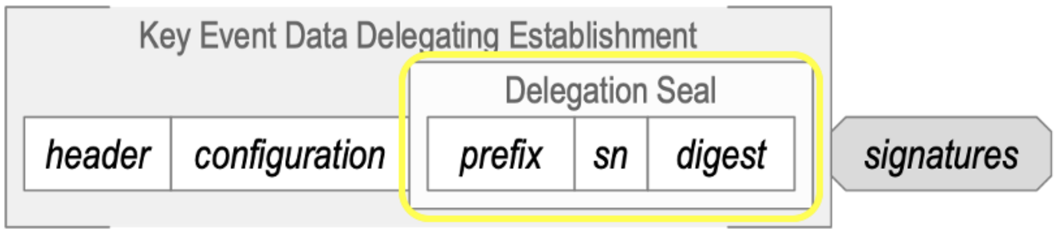


Figure 10.9: KERI key events can include delegation events, where one SCID delegates to another SCID to produce a delegation tree

In addition to delegation, key interaction events (and their logs) can be used for tracking and verifying other operations with a key pair, such as generating a digital signature for an electronic document. Figure 10.10 shows a series of key interaction events from one delegate controller (Delegator C) to another (Delegate D):

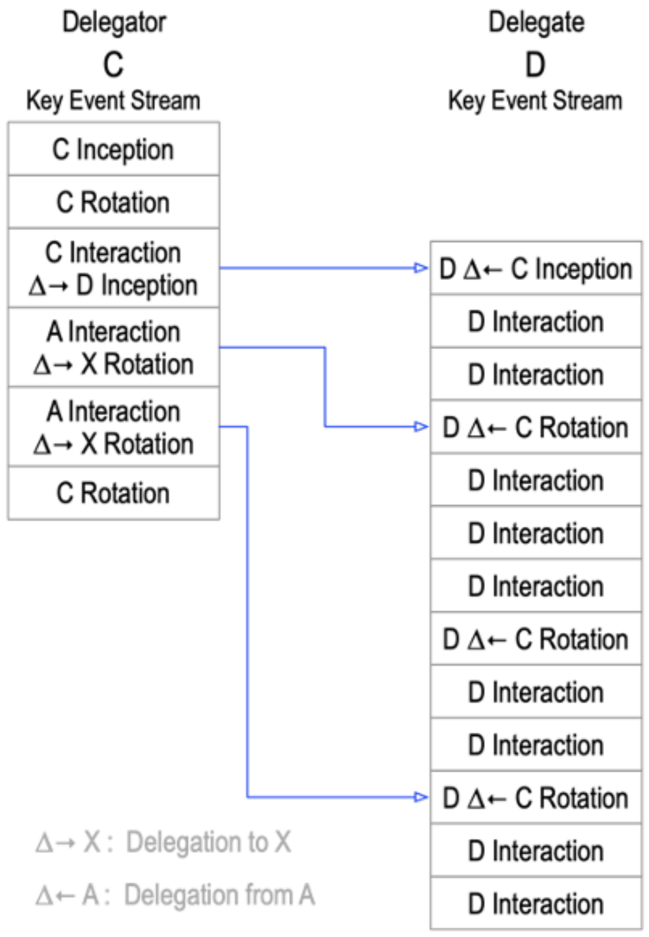


Figure 10.10: A series of key interaction events to perform the inception followed by the rotation of a delegated SCID

Delegated SCIDs can use whatever delegated self-certifying root-of-trust provides the appropriate level of security. Some may need to be delegated to hardware security modules (HSMs) or trusted platform modules (TPMs); others can be hosted on servers locally or in the cloud; still others may be safe enough on edge devices that use secure enclaves.

Delegated SCIDs can also have witnesses—either witnesses that are shared across the entire enterprise or dedicated witnesses for special types of keys or functions. Section 9.5 (pages 69-74) of the KERI technical paper covers the different delegation modes and deployment architectures—including univalent, bivalent, and multivalent—that should be robust enough to serve even large multinational enterprises.

#7: Compatibility with the GDPR "right to be forgotten"

KERI also provides a solution to the longstanding dichotomy in SSI between: a) immutable public blockchains where data lives forever, and b) an individual's "right to be forgotten"—the right granted under the EU GDPR and other data protection regulations for individuals to have their personal data deleted from any system where it is no longer legally required to be stored.

Since a DID that identifies an individual (regardless of the DID method) and its associated public key are both considered personal data under GDPR—*even if the DID is pseudonymous*—then writing that DID and its DID document to an immutable public ledger where it cannot be deleted appears to create an irreconcilable conflict. The Sovrin Governance Framework Working Group spent most of 2019 working with Sovrin stewards (organizations that run nodes of the Sovrin public permissioned blockchain), legal counsel, and GDPR experts trying to solve this problem. The result was a 35-page paper proposing how and why an individual's right to assert a self-sovereign identity should not be in conflict with that same individual’s right to use an immutable public blockchain to secure that self-sovereign identity.[47]

However because the EU Commission and other data protection regulators have yet to rule directly on the matter, this remains an area of regulatory uncertainty (and a potential inhibitor to SSI adoption). So a clear-cut alternative would be very welcome.

KERI provides that alternative. As we have emphasized throughout the preceding sections, the primary root-of-trust for a KERI SCID and key event log is *not* a blockchain or distributed ledger. Rather it is a self-certifying root-of-trust—the digital wallet alone. If an algorithmic root-of-trust like a blockchain is used as a KERI witness, it serves only as an *optional* secondary root-of-trust.

Such a secondary root-of-trust is fine if the key event log is for a public organization where there are no GDPR issues (GDPR applies only to the personal data of individuals). However if the controller is an *individual*—and therefore the SCID and key event log are considered *personal data*—then the obvious solution is: **do not use an immutable public ledger as a witness**.

Instead use any of the myriad other options for witnesses: a distributed database, a replicated directory system, a cloud storage service with automated failover—or all of the above. All of these systems permit the deletion of stored data. And since KERI allows the key event log for one SCID to be deleted without affecting any other SCID, it becomes easy for a witness to comply with the right to be forgotten. *In fact the process can be fully automated because the controller can issue a single KERI protocol command instructing all witnesses to perform the deletion.*

In short, KERI can eliminate the tension between GDPR and SSI so both can meet their intended goals.

[47] *Innovation Meets Compliance: Data Protection Regulation and Distributed Ledger Technology*, Sovrin Foundation, December, 2019. https://sovrin.org/data-protection/

KERI standardization and the KERI DID Method

As should be clear at this point, KERI is actually broader than DIDs. KERI can be used with any type of SCID. And the KERI protocol specifies all the key event message types necessary to support KERI's decentralized key management architecture. This is why standardization of KERI is underway in the Identifiers and Discovery Working Group (IDWG) at the Decentralized Identity Foundation (DIF).[48]

However, KERI is fundamentally compatible with DID architecture and thus can also be implemented either: a) as its own DID method, or b) as an option within other DID methods. Defining a KERI DID method is one of the action items for the DIF IDWG. The current plan is to use the following DID method name:

```
did:keri:
```

KERI support is also planned to be included in the DID method for Hyperledger Indy-based public permissioned blockchains. In this case, KERI-based SCIDs will be a sub-namespace that can be supported on any Indy blockchain using the following syntax:

```
did:indy:[network]:[method-specific-id]

did:indy:[network]:keri:[scid]
```

where:

1. **[network]** is the identifier of a specific Hyperledger Indy-based ledger.
2. **[method-specific-id]** is the identifier of a *non-KERI* DID.
3. **[scid]** is a KERI-based self-certifying identifier.

This forwards-compatible approach allows any Indy network to incorporate KERI SCIDs that return DID documents containing KERI key event receipts.

Other DID methods can also include forwards-compatibility with KERI by taking the same approach of reserving a sub-namespace exclusively for KERI SCIDs.

[48] https://identity.foundation/working-groups/identifiers-discovery.html

A trust-spanning layer for the Internet

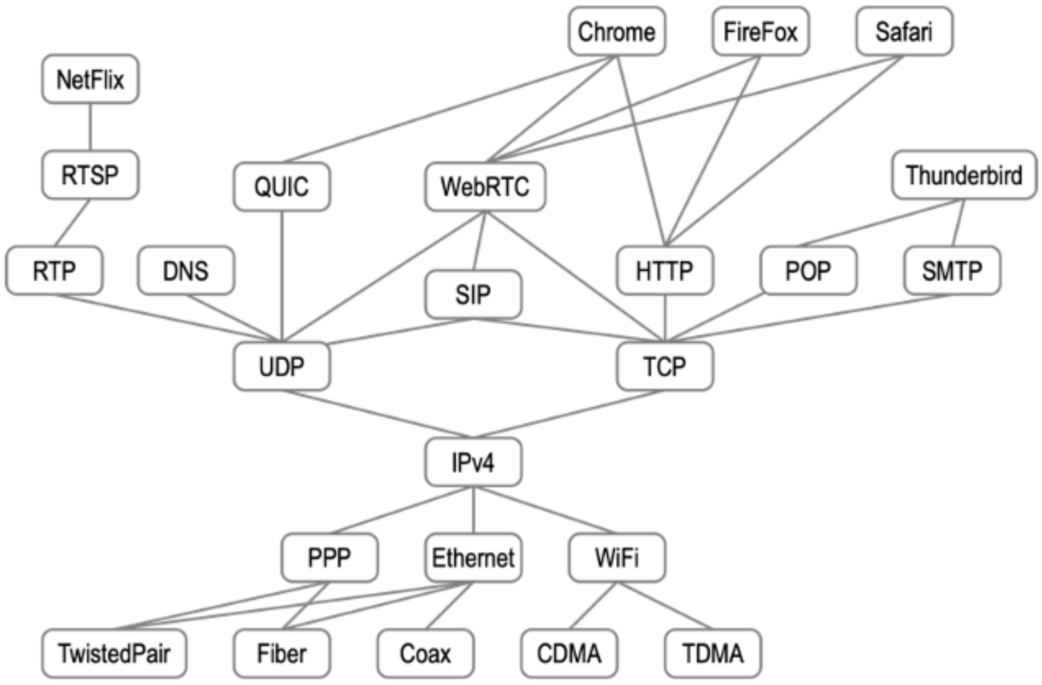
A quote from Page 18 of the KERI technical paper summarizes its trust architecture:

> *Because at issuance self-certifying identifiers make a universally unique cryptographically strong binding between the identifier and a key-pair, there may be no other verifiable source-of-truth besides the controller who created the key-pair and thereby holds the private key.*

This is why KERI is a significant contribution to decentralized key management: it enables every digital wallet used by every controller everywhere to serve as its own self-certifying root-of-trust. And since KERI does not impose any special requirements on any device, system, database, network, or ledger to serve as a KERI witness, all of these can be *secondary* roots-of-trust.

The ability of KERI to provide universally portable, interoperable, and verifiable SCIDs and key event logs means the KERI protocol can enable a **trust spanning layer** for the Internet the same way the Internet Protocol (IP) created a **data spanning layer** for the Internet.

This is a profound concept. It takes all of section 5 of the KERI technical paper (pages 25-29). However the essence of the idea can be conveyed in a few diagrams from that section. The first one, Figure 10.11, shows how the dependencies between the various protocols in the Internet protocol suite[49] form an "hourglass" shape.



[49] https://en.wikipedia.org/wiki/Internet_protocol_suite

Figure 10.11: The Internet protocol suite forms a natural "hourglass" shape with the IP protocol forming the "waist" in the middle

Figure 10.12 simplifies Figure 10.11 to make the hourglass shape more apparent.

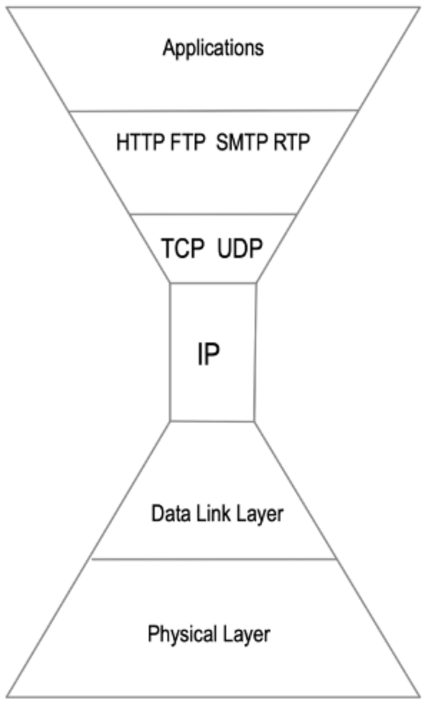


Figure 10.12: A more abstract diagram of Figure 10.11 that shows how the IP protocol is at the middle of the hourglass formed by the supporting protocols below it and the supported protocols above it

The "hourglass theorem" of protocol stack design is wonderfully articulated in a 2019 ACM paper by Micah Beck.[50] He summarizes the theorem this way:

> *The shape suggested by the hourglass model expresses the goal that the spanning layer should support various applications and be implementable using many possible supporting layers. Referring to the hourglass as a design tool also expresses the intuition that restricting the functionality of the spanning layer is instrumental in achieving these goals. The elements of the model are combined visually in the form of an hourglass shape, with the "thin waist" of the hourglass representing the restricted spanning layer, and its large upper and lower bells representing the multiplicity of applications and supporting layers, respectively.*

[50] Communications of the ACM, July 2019, Vol. 62 No. 7, Pages 48-57, 10.1145/3274770. https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/fulltext

Figure 10.13 from Beck's paper illustrates how a spanning layer should be designed to be as "thin", "weak", or "restricted" as it can be and still support the applications above it.

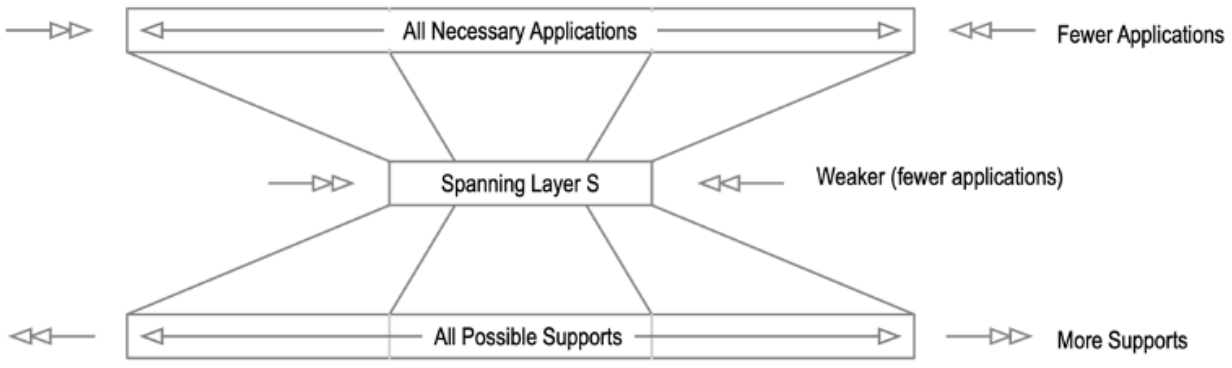


Figure 10.13: A spanning layer is the "weakest" (simplest) possible layer that still supports the necessary applications above it

Beck's paper goes on to define a formal model to explain why the hourglass theorem works. The paper also provides several examples of where the hourglass theorem has been applied in multicasting, Internet address translation, and the Unix operating system.

Section 5 of the KERI technical paper builds on this foundation by proposing another application of the hourglass theorem to a different kind of spanning layer—a *trust spanning layer* for the Internet. This is necessary because as page 26 of the paper explains:

> *The realization that IP is both the spanning layer for the internet and has no built in security mechanism makes it obvious why security on the internet is broken; its spanning layer is insecure (not trustable)!*

In other words, we cannot fix the missing security in the IP protocol directly—at the level of the existing data spanning layer—because that ship sailed 40+ years ago. However, we *can* fix it now by adding a second higher-level spanning layer. From page 27 of the paper:

> *Because a [trust spanning layer] necessarily uses protocols above the IP layer, it cannot span the internet at the IP layer but must span somewhere above it. This gives rise to a "double waisted" or "waist and neck" shape where the [trust spanning layer] is the neck.*

Figure 10.4 is a diagram of this "double hourglass" shape showing both the IP spanning layer and the trust spanning layer.

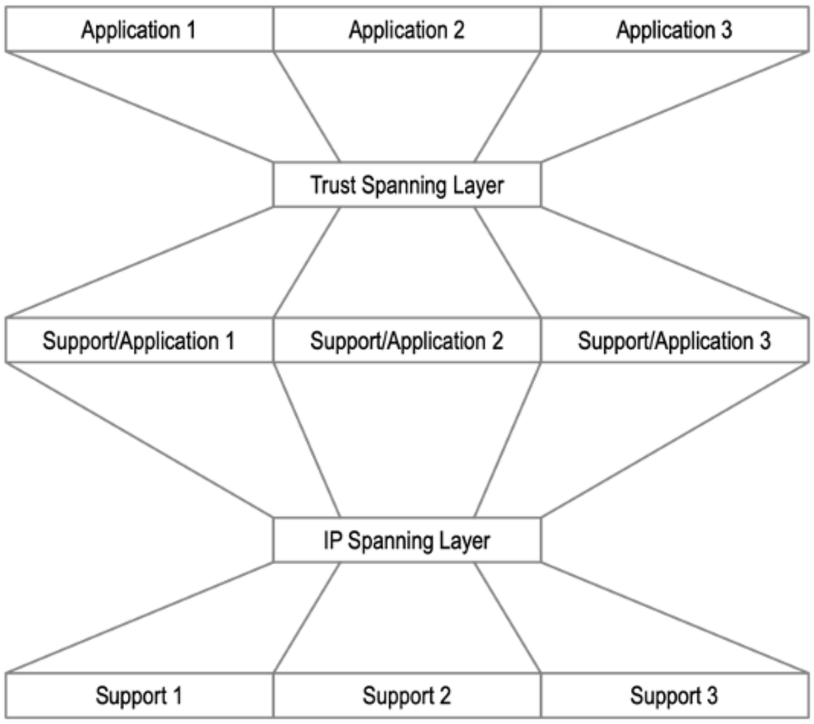


Figure 10.14: With KERI, we can have a "double hourglass"—a trust spanning layer on top of the applications supported by the IP spanning layer

This prospect—of an interoperable trust spanning layer that works everywhere across the Internet, allowing any two peers to connect and establish mutual, cryptographically verifiable trust—is enormously exciting. It aligns perfectly with the goals of the ToIP stack, shown again in Figure 10.15 with a highlight over Layers 1 and 2 where KERI can be implemented in digital wallets at Layer 2 and with public utilities serving as KERI witnesses at Layer 1.

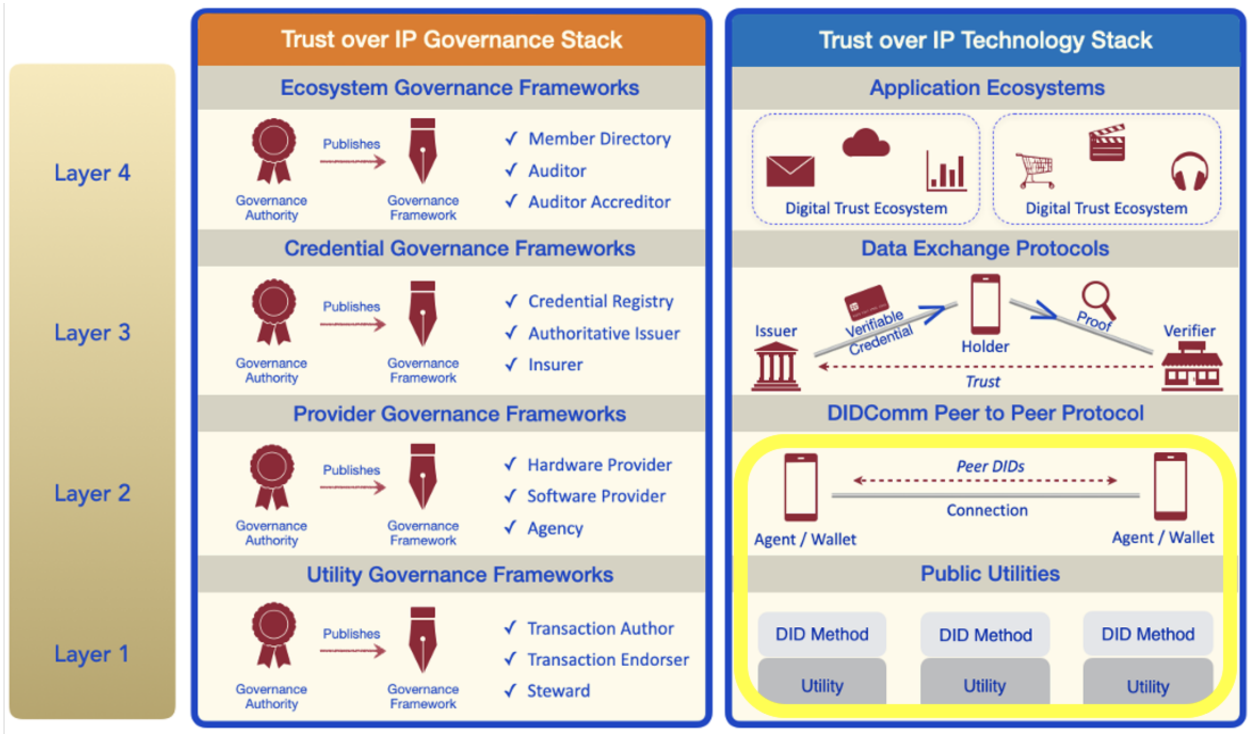


Figure 10.15: KERI fits perfectly into the ToIP stack, with its self-certifying roots-of-trust living in digital wallets at Layer 2 and public utilities serving as KERI witnesses at Layer 1

Most importantly, KERI gives us a consistent way to implement decentralized key management that can be integrated across all the devices, systems, networks, and applications that we use every day. To be sure, the development, battle-testing, deployment, and integration of KERI infrastructure will take time—just as the adoption of the Internet took time. But if KERI can deliver a trust spanning layer for the Internet, its adoption will become as inevitable as the adoption of the Internet was forty years ago.

Summary

Key takeaways from this chapter:

- All forms of cryptographic key management are hard because digital keys are just strings of bits that must be very carefully guarded, and if they are lost, stolen, or corrupted, they can be literally irreplaceable.
- Standards and protocols for conventional key management are well established, including major publications from NIST and the Key Management Interoperability Protocol (KMIP) from OASIS.
- The paradigm shift to decentralized key management is the migration from *administrative* roots-of-trust to *algorithmic* or *self-certifying roots-of-trust*. Both of the latter eliminate a dependency on trust in humans or organizational assertions of

new or rotated keys.

- With this new power comes new responsibilities—key management responsibilities that now fall directly onto the shoulders of the self-sovereign individuals because with SSI there is no "higher authority" to turn to.
- This requires a decentralized key management system (DKMS)—an initial design for which was developed in 2018-19 by Evernym under a contract from the U.S. Department of Homeland Security.
- DKMS must be a vendor-neutral open standard that anyone can implement, that supports an open-ended number of ciphersuites over time, and that enables digital wallets to be portable across vendors, devices, systems, and networks.
- DKMS cannot require any specialized knowledge on behalf of end-users and must provide built-in support for key recovery, whether offline, social, or both.
- DIDs, VCs, and SSI bring new tools to help enable decentralized key management, including separating key verification from identity verification, using VCs for proof of identity, automated key rotation, automated backup and recovery methods, and digital guardianship.
- The vast majority (95%) of DID methods currently use blockchains or distributed ledgers as algorithmic root-of-trust, but these DIDs are "ledger-locked" (not portable) and may have conflicts with the GDPR "right-to-be-forgotten".
- Peer DIDs do not use a ledger—they use a simple form of self-certifying identifier (SCID) that relies exclusively on a self-certifying root-of-trust (the digital wallet) and are shared directly peer-to-peer where each peer can verify them. They are also highly scalable and privacy-preserving—the only downside is that they are not publicly discoverable.
- KERI (Key Event Receipt Infrastructure) uses a generalized approach to SCIDs that provides a complete decentralized key management infrastructure for any application—a solution even more generalized than DIDs.
- To enable secure, self-certifying key rotation, KERI uses *key event logs*—an ordered sequence of all key rotation events in which each new record contains a digest of the previous record and is digitally signed by the new private key.
- Key event logs can be signed by other trusted parties called *witnesses* serving as secondary roots-of-trust.
- KERI supports a feature called *pre-rotation* that minimizes the potential damage if current active private key is stolen or compromised.
- KERI's self-certifying identifiers and self-certifying key event logs mean anyone anywhere can verify the current public key for a SCID—a property called *ambient verifiability*.
- KERI also supports identifier delegation so enterprises can create a scalable ecosystem of KERI-based SCIDs and key event logs.
- Since KERI does not require an immutable ledger, it can avoid GDPR issues with the "right-to-be-forgotten".
- The ability of KERI to provide universally portable, interoperable, and verifiable SCIDs and key event logs means the KERI protocol can enable a *trust spanning layer* for the Internet the same way the Internet Protocol (IP) created a *data spanning*

layer for the Internet. This fits perfectly with Layers 1 and 2 of the ToIP stack.

Having covered one of the deepest technical topics in this book, we now need just one more chapter to complete our deep-dive into SSI technology. Ironically this final chapter in Part 2 is not about technology—it is about a different kind of “code” we need alongside technology to address the human side of digital trust: **governance frameworks**.