DKMS Requirements Summary

2017-09-30
A submission to Rebooting the Web of Trust #5
Drummond Reed, Evernym

Contributors:
Jason Law, Daniel Hardman, Nathan George, Devin Fisher, Mike Lodder, and Brent Zundel—Evernym
Manu Sporny and David Longley—Digital Bazaar
Kent Seamons—BYU Internet Security Research Lab
Stephen Wilson—Lockstep Consulting
Christopher Allen—Alacrity Management

Acknowledgements

Work on the DID and DKMS specifications has been funded in part by a Small Business Innovation Research (SBIR) grant from the U.S. Department of Homeland Security Science and Technology Directorate. The content of this specification does not necessarily reflect the position or the policy of the U.S. Government and no official endorsement should be inferred.

Table of Contents

1. Introduction **3**

2. Analysis of the Literature **4**

3. Analysis of NIST 800-130 Requirements **5**

3.1. Introduction 5

3.2. Basic Framework Requirements (Section 2) 5

3.3. Goals of the CKMS Design (Section 3) 6

3.4. Security Policies (Section 4) 6

3.5. Roles and Responsibilities (Section 5) 8

3.6. Cryptographic Keys and Metadata (Section 6) 8

3.7. Interoperability and Transitioning (Section 7) 9

3.8. Security Controls (Section 8) 9

3.9. Testing and System Assurances (Section 9) 10

3.10. Disaster Recovery (Section 10) 10

3.11. Security Assessment (Section 11) 11

3.12. Technological Challenges (Section 12) 11

3.13. Conclusions 12

4. Analysis of Other Key Management Guidelines 13

4.1. NIST 800-57 Part 1: Recommendation for Key Management 13

4.2. OWASP Key Management Cheat Sheet 13

4.3. KMIP (Key Management Interoperability Protocol) 14

5. Analysis of Unique DKMS Requirements 15

6.1. Decentralization and Decentralized Identifiers (DIDs) 15

6.2. Privacy and Pseudonymity 16

6.3. Usability 16

6.4. Multiple Trust Models 17

6.5. Delegation and DID Guardianship 17

6.6. Portability 17

6.7. Extensibility 18

6.8. Simplicity 18

6.9. Open System and Open Standard 19

6. Conclusion 20

1. Introduction

This document summarized our research into the user, business, and technology requirements for a DKMS (Decentralized Key Management System) in comparison to the requirements for conventional CKMS (Cryptographic Key Management Systems) as detailed in NIST Special Publication 800-130: *A Framework for Designing Cryptographic Key Management Systems*.

This document contains the following sections:

1. **Analysis of the Literature Search:** Highlights from BYU Internet Security Research Lab's search of academic and industry research documents on the subjects related to distributed systems, blockchains, and key management.
2. **Analysis of NIST 800-130 Requirements.** A summary of our work to define the DKMS requirements according to the framework defined in the NIST 800-130 requirements for designing conventional CKMS.
3. **Analysis of Other Key Management Guidelines.** A summary of inputs from other key management publications and standards.
4. **Analysis of Unique DKMS Requirements.** Our analysis of the requirements that we believe to be unique to the DKMS model of decentralized key management based on DIDs (decentralized identifiers) rooted on distributed ledgers.
5. **Conclusion.** Our final recommendations regarding DKMS requirements.

2. Analysis of the Literature

As part of our research on requirements, Evernym worked with BYU Internet Security Research Lab (ISRL) to compile a categorized list of papers from academia and research on the following topics:

1. Key Management
    a. Key Exchange
    b. Key Storage
    c. Key Usage / Forward Secrecy
    d. Key Transparency
    e. Revocation
2. Certificates and PKI
    a. Public Key Infrastructure
    b. Automated Certificate Issuance
    c. Web of Trust
    d. Bitcoin Key Management
3. Usability
    a. Secure Messaging
    b. Secure Email
4. Password Management
5. Recovery and Backup
    a. Personal Knowledge Questions
    b. Social Recovery
    c. Account Hijacking
    d. Backup Authentication Schemes

This survey of the literature reveals some surprising insights about the requirements for DKMS:

1. **There have been no prior usability studies of key recovery and key portability.** This suggests that the opportunity for a decentralized key management system is relatively new, due to recent emergence of large-scale public distributed ledgers.
2. **Usability studies of secure messaging and secure email have shown both achieve usable end-to-end encryption, but neither demonstrate usable key management.** Users of secure messaging apps such as WhatsApp consistently fail to correctly validate keys, and usable secure email tools handle key management so users don't have to.
3. **There has never been a successful system that provides usable key management to end users.** This illustrates the core usability challenge facing DKMS.
4. **Account recovery using security questions has been shown repeatedly to be vulnerable to attack.** This reinforces that DKMS account recovery and key recovery need to take a very different approach—which it must anyway, because DKMS key management cannot rely on centralized service providers to begin with.

3. Analysis of NIST 800-130 Requirements

3.1. Introduction

NIST Special Publication 800-130, *A Framework for Designing Cryptographic Key Management Systems*, is the most comprehensive framework of its kind. It contains 258 individual requirements in 11 categories covering every aspect of cryptographic key management from key generation to quantum computing threats.

From the introduction:

> *This Framework for Designing Cryptographic Key Management Systems (CKMS) is a description of the topics to be considered and the documentation requirements (henceforth referred to as requirements) to be addressed when designing a CKMS. The CKMS designer satisfies the requirements by selecting the policies, procedures, components (hardware, software, and firmware), and devices (groups of components) to be incorporated into the CKMS, and then specifying how these items are employed to meet the requirements of this Framework.*
>
> *A CKMS consists of policies, procedures, components and devices that are used to protect, manage and distribute cryptographic keys and certain specific information, called (associated) metadata herein. A CKMS includes all devices or sub-systems that can access an unencrypted key or its metadata. Encrypted keys and their cryptographically protected (bound) metadata can be handled by computers and transmitted through communications systems and stored in media that are not considered to be part of a CKMS.*

It was precisely this comprehensive coverage of all aspects of key management system design that led Evernym to select NIST 800-130 as the starting point for the DKMS requirements.

The following sections provide a high level section-by-section summary of our analysis.

3.2. Basic Framework Requirements (Section 2)

DKMS has the same requirement	4	80%
DKMS has a modified requirement	1	20%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**5**	**100%**

This section discusses the motivation, intent, properties, and limitations of a Cryptographic Key

Management Framework. It is highly aligned with the intent of the DKMS specifications. The only modified requirement was:

> **FR:2.5** The CKMS design **shall** specify all major devices of the CKMS (e.g., the make, model, and version).

Our modified requirement reflects the overall theme of many of the modified requirements:

> Since the DKMS is a generalized and extensible framework, it is not possible to specify all the devices that it will encompass. What the DKMS can specify is requirements for the types of devices it is intended to support, e.g., mobile phones, laptops, desktops, servers. etc.

In other words, NIST 800-130 is intended mainly as a framework for producing a specific instance of a CKMS, not necessarily another framework. While DKMS is an instance of a CKMS, it also functions in some respects as a framework. This is further discussed in the Conclusion, below.

3.3. Goals of the CKMS Design (Section 3)

DKMS has the same requirement	7	43.75%
DKMS has a modified requirement	6	37.5%
Requirement is not applicable to DKMS	3	18.75%
Total requirements in this section	**16**	**100%**

Again, there is a high degree of alignment between the goals of NIST 800-130 and those of DKMS. The only reason for the higher number of modified requirements is that some NIST 800-130 requirements apply only to specific CKMS that are able to identify all:

- Intended applications
- Intended users and responsibilities
- Target devices
- Third party testing programs
- User interfaces
- Error prevention features

While the DKMS specifications can define general requirements for these components and features, as a framework it cannot be fully prescriptive for all these requirements.

3.4. Security Policies (Section 4)

DKMS has the same requirement	22	81%
DKMS has a modified requirement	5	19%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**27**	**100%**

The key difference in this section is that NIST 800-130 assumes that a specific CKMS will define an overall CKMS Security Policy covering the entire CKMS. As a framework, DKMS can cover any number of security domains, so it cannot specify a single overall security policy. It can, however, work in conjunction with trust frameworks—potentially even global trust frameworks—to apply specific security policies. And these trust frameworks can be layered precisely as recommended in this figure from page 19 of NIST 800-130:

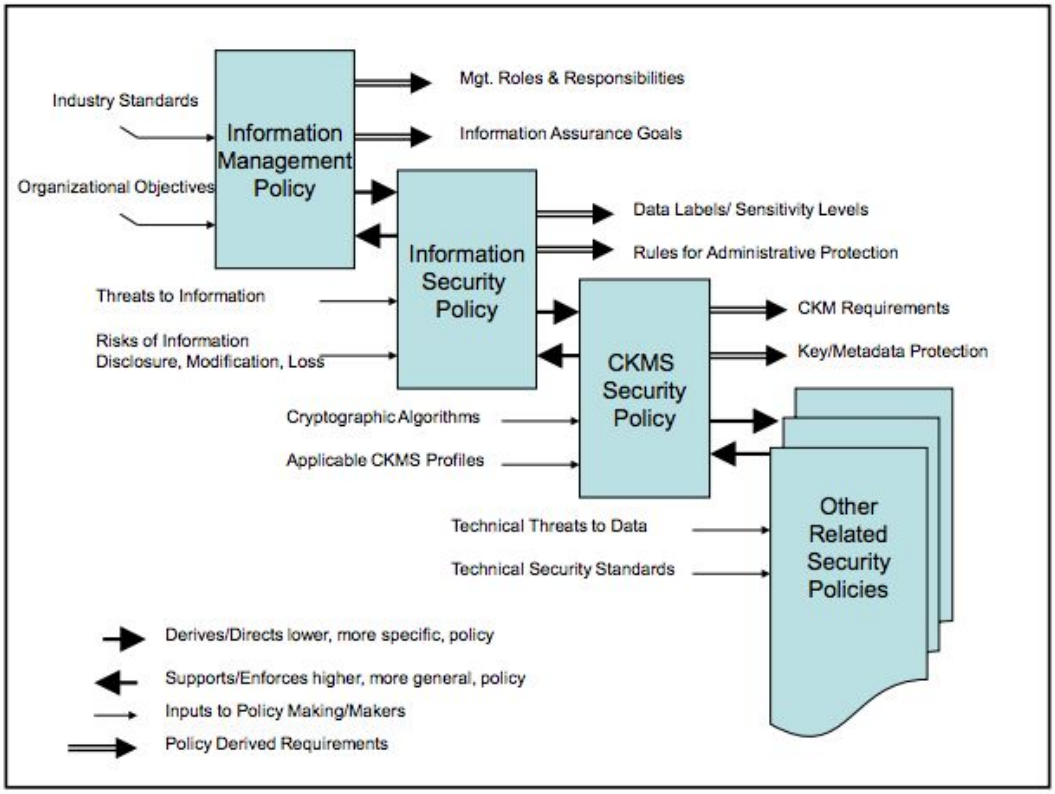

Figure 7: Related Security Policies

3.5. Roles and Responsibilities (Section 5)

DKMS has the same requirement	4	80%
DKMS has a modified requirement	1	20%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**5**	**100%**

Again, there is a high degree of alignment in this section. The only modified requirement is this one:

> **FR:5.5** The CKMS design **shall** specify all automated provisions for identifying security violations, whether by individuals performing authorized roles (insiders) or by those with no authorized role (outsiders).

This is overly prescriptive for DKMS as a framework. Thus the modified requirement:

> As a generalized and extensible framework, the DKMS specifications cannot specify all automated provisions for identifying security violations. It can define standards and best practices, but it should also encourage market forces to innovate improvements in security violation detection and remediation.

3.6. Cryptographic Keys and Metadata (Section 6)

DKMS has the same requirement	76	61.3%
DKMS has a modified requirement	45	36.3%
Requirement is not applicable to DKMS	3	2.4%
Total requirements in this section	**124**	**100%**

This is by far the largest section of NIST 800-130, encompassing nearly half the requirements. The alignment in this section almost perfectly mirrors the overall alignment (see the Conclusion, below). Again, the primary difference in alignment is the assumption in NIST 800-130 that all conforming CKMS will be concrete instances and not frameworks. This is represented in the first requirement in this section:

> **FR:6.1** The CKMS design **shall** specify and define each key type used.

The modified requirement explains why the DKMS specifications need to take a more extensible approach:

> While the DKMS specifications will define a standard set of key types, they are intentionally extensible to support new key types as they are introduced by new distributed ledger and encryption technologies. A primary purpose of the DKMS specifications is to enable prompt and safe propagation of these new key types among DKMS wallets and agents.

This extensibility is actually a primary feature of the DKMS design, since it introduces a dimension of evolutionary resilience that is needed to maintain a highly scalable cybersecurity infrastructure over time.

Most of the other modified requirements are due to the difference that the NIST 800-130 expects a CKMS designer can define all hardware, software, and policies used throughout the system, whereas DKMS as a framework is not able to be fully prescriptive.

3.7. Interoperability and Transitioning (Section 7)

DKMS has the same requirement	8	100%
DKMS has a modified requirement	0	0%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**8**	**100%**

This is the only section in which NIST 800-130 and DKMS requirements are in 100% alignment. This highlights how interoperability is a paramount requirement of DKMS. However DKMS also has a strong requirement for portability that is not represented in the 8 requirements in this section. See the *Analysis of Unique DKMS Requirements* below for more.

3.8. Security Controls (Section 8)

DKMS has the same requirement	6	31.6%
DKMS has a modified requirement	13	68.4%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**19**	**100%**

By contrast, this section has the lowest level of alignment. The reason is simply that most CKMS will be highly prescriptive about the security controls throughout, whereas DKMS as a

framework is not designed for that level of prescription. The first requirement in this section is a good example:

> **FR:8.1** The CKMS design **shall** specify each of its CKMS devices and their intended purposes.

The DKMS modified requirement is:

> As a generalized and extensible framework, the DKMS specifications cannot specify specific devices. The DKMS specifications shall specify classes of devices and the recommended requirements for those devices as well as conformance requirements for trust frameworks that reference the DKMS specifications.

This applies uniformly to all the modified requirements in this section—they all apply, just at a more general level with DKMS.

3.9. Testing and System Assurances (Section 9)

DKMS has the same requirement	9	52.9%
DKMS has a modified requirement	8	47.1%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**17**	**100%**

The same analysis applies to this section. In addition, to the extent DKMS is itself a framework, it may require its own certification and testing programs, perhaps in conjunction with specific trust frameworks. See the *Analysis of Unique DKMS Requirements* below for more.

3.10. Disaster Recovery (Section 10)

DKMS has the same requirement	6	50%
DKMS has a modified requirement	3	25%
Requirement is not applicable to DKMS	3	25%
Total requirements in this section	**12**	**100%**

There is only moderate alignment of the requirements in this section for two reasons:

1. At the macro level, the disaster recovery requirements of a conventional CKMS do not apply to distributed ledger technologies, especially those for public blockchains that

operate at global scale.
2. At the micro level (individual identity owners), the disaster recovery requirements are different because a DKMS has no centralized control or management.

However, disaster recovery is extremely important in both conventional CKMS and in DKMS, so the two will only differ in specific strategies for achieving it.

3.11. Security Assessment (Section 11)

DKMS has the same requirement	15	93.75%
DKMS has a modified requirement	1	6.25%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**16**	**100%**

Despite the differences between conventional CKMS and DKMS, there is very high alignment on the requirements for security assessment of both.

3.12. Technological Challenges (Section 12)

DKMS has the same requirement	6	75%
DKMS has a modified requirement	2	25%
Requirement is not applicable to DKMS	0	0%
Total requirements in this section	**8**	**100%**

There is also very high alignment within this final section, particularly when you consider the only two modified requirements are those that deal with “external access to CKMS devices”, which only partially applies to a decentralized key management system.

3.13. Conclusions

The overall alignment between NIST 800-130 requirements for CKMS design and the DKMS design requirements are shown in this table and in Figure 1:

DKMS has the same requirement	163	63%
DKMS has a modified requirement	85	33%
Requirement is not applicable to DKMS	10	4%
Total requirements	**258**	**100%**

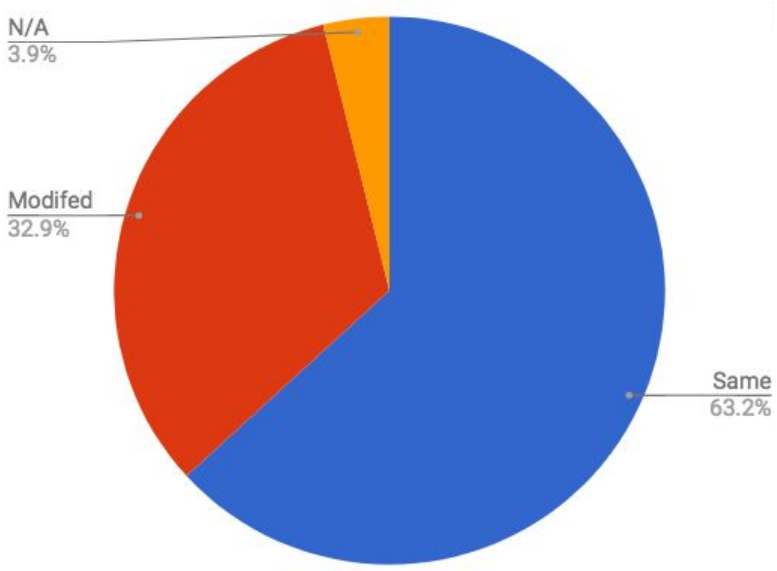

Figure 1: Alignment between NIST 800-130 and DKMS requirements

As the notes on each section reveal, the primary reason for the modified or N/A requirements is that NIST 800-130 is intended to be a framework for designing concrete CKMS instances, while DKMS is both *an instance of a CKMS* and *a framework for defining conformant DKMS products* from an open market of vendors and service providers. As a “subframework”, the DKMS specifications cannot be as prescriptive as most concrete CKMS instance specifications based on NIST 800-130.

That said, the vast majority (96%) of the 258 NIST 800-130 requirements apply directly or with relatively little modification to DKMS. Very few do not apply, and those that do not are requirements that assume that a CKMS is operated by a centralized authority or uses a specific physical plant, set of devices, or set of keys.

Overall, in our work we were struck by both the comprehensive coverage and remarkable level of detail achieved by the NIST 800-130 requirements. There is good reason it has proved itself the gold standard for CKMS design, and this will be of great benefit as we proceed with DKMS design and architecture.

4. Analysis of Other Key Management Guidelines

4.1. NIST 800-57 Part 1: Recommendation for Key Management

http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

NIST 800-57 is not a requirements document, but a set of recommendations for CKMS designers and developers. From the Introduction:

> *Part 1, General, contains basic key management guidance. It is intended to advise developers and system administrators on the "best practices" associated with key management. Cryptographic module developers may benefit from this general guidance by obtaining a greater understanding of the key management features that are required to support specific, intended ranges of applications. Protocol developers may identify key management characteristics associated with specific suites of algorithms and gain a greater understanding of the security services provided by those algorithms. System administrators may use this document to determine which configuration settings are most appropriate for their information.*

While it does not introduce specific new requirements, the advice in NIST 800-57 should be taken into account in the design and architecture stage of DKMS.

4.2. OWASP Key Management Cheat Sheet

https://www.owasp.org/index.php/Key_Management_Cheat_Sheet

This is a concise and direct set of recommendations for secure usage of cryptographic keys from the Open Web Application Security Project (OWASP). From the introduction:

> *This Key Management Cheat Sheet provides developers with guidance for implementation of cryptographic key management within an application in a secure manner. It is important to document and harmonize rules and practices for:*
>
> 1. *Key life cycle management (generation, distribution, destruction)*
> 2. *Key compromise, recovery and zeroization*
> 3. *Key storage*
> 4. *Key agreement*

Like NIST 800-57, the OWASP Key Management Cheat Sheet does not introduce any specific

new requirements, but it does guide developers with references to specific publications and standards from NIST and others for dealing with particular challenges in key management. This guidance should be reflected in the design and architecture of the DKMS specifications.

4.3. KMIP (Key Management Interoperability Protocol)

https://en.wikipedia.org/wiki/Key_Management_Interoperability_Protocol

KMIP is an OASIS Technical Committee focused on specifications for interoperability of key management products and services. From the Wikipedia page:

> *The **Key Management Interoperability Protocol** (**KMIP**) is an extensible communication protocol that defines message formats for the manipulation of cryptographic keys on a key management server. Keys may be created on a server and then retrieved, possibly wrapped by other keys. Both symmetric and asymmetric keys are supported, including the ability to sign certificates. KMIP also allows for clients to ask a server to encrypt or decrypt data, without needing direct access to the key.*
>
> *The KMIP standard was first released in 2010 and has since become the industry standard for key management. Vendors have demonstrated commercially available clients and servers at every recent RSA Conference.*

While the KMIP Technical Committee has not produced specific requirements documents (that we could find), the design of the KMIP protocol and key management artifacts is highly relevant to the DKMS design precisely because the entire focus is interoperability. The biggest difference is that KMIP is focused on server-centric key management for enterprises, whereas DKMS architecture is “identity owner centric”, which in the case of individuals means starting with edge devices and working outwards.

Nonetheless, we anticipate that DKMS interoperability architecture will be significantly influenced by KMIP architecture and the design choices made by the KMIP Technical Committee.

5. Analysis of Unique DKMS Requirements

As comprehensive as NIST 800-130 is, it is still rooted in the worldview of conventional PKI (public key infrastructure). This worldview assumes a CKMS:

- Has a centralized body defining, implementing, and controlling usage of the CKMS (even if it is a federated system).
- Has a bounded deployment target with a bounded set of roles, domains, and devices.
- Has a specific trust model with a fixed set of security and privacy policies.
- Has the ability to train and educate all necessary personnel on the usage of the CKMS.

A DKMS cannot make any of these assumptions. In many ways this is analogous to how a local area network compares to the Internet. The local area network can make all the assumptions above; the Internet cannot make any of them.

Yet the Internet not only exists, it has forever altered computing and communications.

A DKMS has similar potential with regard to the deployment of cryptographic key infrastructure—it can literally enable “the Internet of keys”. But as such, it has a unique set of requirements that go beyond those in NIST 800-130. Those requirements are enumerated in this section.

6.1. Decentralization and Decentralized Identifiers (DIDs)

The DKMS design MUST NOT assume any reliance on a centralized authority for the system as a whole.

The DKMS design MUST assume all participants are independent actors identified with DIDs conformant with the DID Data Model and Generic Syntax specification but otherwise acting in their own decentralized security and privacy domains.

The DKMS design MUST support options for decentralized key recovery.

What distinguishes Distributed Key Management from conventional CKMS is the fact that the entire design assumes decentralization: there is no central authority responsible for many of the traditional functions of a key management system. This begins with identification and addressing: by definition, entities participating in a DKMS all have DIDs, and therefore have a common set of metadata available to bootstrap all other interactions.

The lack of a central authority as a fallback means that a global DKMS infrastructure must

achieve interoperability organically based on a shared set of specifications, just like the Internet.

Note that the need to maintain decentralization is most acute when it comes to key recovery: the advantages of decentralization are nullified if key recovery mechanisms reintroduce centralization.

6.2. Privacy and Pseudonymity

The DKMS design MUST NOT introduce new means of correlating participants by virtue of using the DKMS standards.

The DKMS design SHOULD increase privacy and security by enabling the use of pseudonyms, selective disclosure, and encrypted private channels of communication.

Any usage of biometrics in DKMS architecture MUST protect the biometric owner's privacy.

Conventional PKI and CKMS rarely have anti-correlation as a primary requirement. Distributed key management should ensure that participants will have more, not less, control over their privacy as well as their security.

This is especially important when it comes to biometrics. Biometrics can play a special role in the DKMS architecture because it is one aspect of an individual's unique identity that requires no effort to maintain. But this same quality means a privacy breach of biometric attributes could be disastrous because they may be unrecoverable.

6.3. Usability

The components of the DKMS design intended for usage by individual identity owners MUST be safely usable without any special training or knowledge of cryptography or key management.

In many ways this follows from decentralization: in a DKMS, there is no central authority to teach everyone how to use it. It must be automated and intuitive to a very high degree, similar to the usability achieved by modern encrypted OTT messaging products like Whatsapp, iMessage, and Signal.

According to the BYU Internet Security Research Lab, this level of usability is a necessary property of any successfully deployed system. "We spent the 1990s building and deploying security that wasn't really needed, and now that it's actually desirable, we're finding that nobody

can use it" [Guttman and Grigg, IEEE Security and Privacy, 2005]. The DKMS needs to be able to support a broad spectrum of applications, with both manual and automatic key management, in order to satisfy the numerous security and usability requirements of those applications.

Again, this requirement is particularly acute when it comes to key recovery. Because there is no central authority to fall back on, the key recovery options must not only be anticipated and implemented in advance, but they must be easy enough for a non-technical user to employ while still preventing exploitation by an attacker.

6.4. Multiple Trust Models

The DKMS design MUST NOT assume a single uniform trust model or trust framework.

The DKMS design MUST enable participants to employ multiple trust models or trust frameworks and to extend these as required.

Most CKMS systems have one specific trust model. The decentralized nature of DKMS requires it be more flexible and open to extension, particularly at the edges of the network.

6.5. Delegation and DID Guardianship

The DKMS design MUST enable key management to be delegated by one identity owner to another, including the DID concept of guardianship.

Although the DKMS infrastructure enables self-sovereign identity, not all individuals have the ability to be self-sovereign. They may be operating at a physical, economic, or network disadvantage that requires another identity owner (individual or org) to manage their keys.

Other identity owners may simply prefer to have others manage their keys for purposes of convenience, efficiency, or safety. In either case, this means DKMS architecture needs to incorporate the concept of **guardianship** as defined in the DID Data Model and Generic Syntax 1.0 specification.

6.6. Portability

The DKMS design MUST enable an identity owner's DKMS-compliant key management capabilities to be portable across multiple DKMS-compliant devices, applications, and service providers.

While the NIST 800-130 specifications have an entire section on interoperability, those requirements are focused primarily on interoperability of CKMS components with each other and with external CKMS systems. They do not encompass the need for a decentralized identity owner to be able to port their key management capabilities from one CKMS device, application, or service provider to another.

This is the DID and DKMS equivalent of telephone number portability, and it is critical not only for the general acceptance of DKMS infrastructure, but to support the ability of DID owners to act with full autonomy and independence. As with telephone number portability, it also helps ensure a robust and competitive marketplace for DKMS-compliant products and services.

6.7. Extensibility

The DKMS design SHOULD be capable of being extended to support new cryptographic algorithms, keys, data structures, and modules, as well as new distributed ledger technologies and other security and privacy innovations.

Section 7 of NIST 800-130 includes several requirements for conventional CKMS to be able to transition to newer and stronger cryptographic algorithms, but it does not go as far as is required for DKMS infrastructure, which must be capable of adapting to evolving Internet security and privacy infrastructure as well as rapid advances in distributed ledger technologies.

It is worth noting that, although the DKMS specifications will not themselves include a trust framework; rather, one or more trust frameworks can be layered over them to formalize certain types of extensions. This provides a flexible and adaptable method of extending DKMS to meet the needs of specific communities.

6.8. Simplicity

Given the inherent complexity of key management, the DKMS design SHOULD aim to be as simple and interoperable as possible by pushing complexity to the edges and to extensions.

Simplicity and elegance of design are common traits of most successful decentralized systems, starting with the packet-based design of the Internet itself. The less complex a system is, the easier it is to debug, evaluate, and adapt to future changes. Especially in light of the highly comprehensive scope of NIST 800-130, this requirement highlights a core difference with conventional CKMS design: the DKMS specification should NOT try to do everything, e.g.,

enumerate every possible type of key or role of user or application, but let those be defined locally in a way that is interoperable with the rest of the system.

6.9. Open System and Open Standard

The DKMS design MUST be an open system based on open, royalty-free standards.

While many CKMS systems are deployed using proprietary technology, the baseline DKMS infrastructure must, like the Internet itself, be an open, royalty-free system. It may, of course, have many proprietary extensions and solutions built on top of it.

6. Conclusion

Our examination of conventional CKMS requirements as enumerated in tremendous detail by NIST 800-130 revealed that two-thirds of them applied directly to a decentralized key management system, and most of the rest applied with some modification. However the requirements of conventional CKMS systems, rooted in hierarchical PKI architecture, do not completely capture all the requirements of a DKMS infrastructure precisely because it has a different “root”: distributed ledger technology (DLT), for which the starting point in design is the need for decentralization and all that it entails. This is shown graphically in Figure 2:

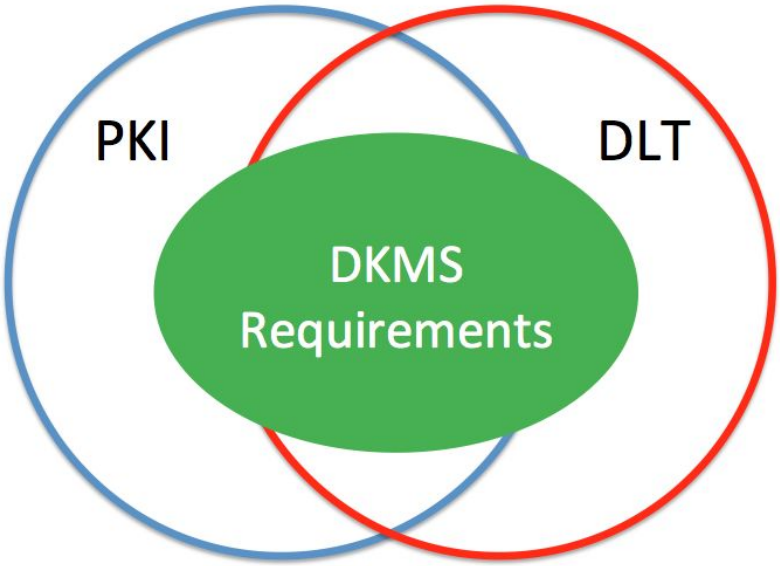

Figure 2: The requirements of DKMS span both conventional PKI and DLT architectures

However, by spanning both conventional PKI and distributed ledger technologies, DKMS infrastructure holds the promise to foster the “Internet of keys” and democratize the power of public/private key cryptography.