



THE DEFINITIVE GUIDE TO ENCRYPTION KEY MANAGEMENT FUNDAMENTALS



WHAT IS ENCRYPTION KEY MANAGEMENT?

Encryption key management is administering the full lifecycle of cryptographic keys and protecting them from loss or misuse. The lifecycle includes: generating, using, storing, archiving, and deleting of keys. Protection of the encryption keys includes limiting access to the keys physically, logically, and through user/role access.

SHORTCUTS

[Introduction](#)

[Types of Encryption Keys and How They Work](#)

[How Encryption Key Systems Work](#)

[The Full Life-Cycle of Keys](#)

[Segregated Roles in Key Management](#)

[The Domains to Secure Encryption Keys](#)

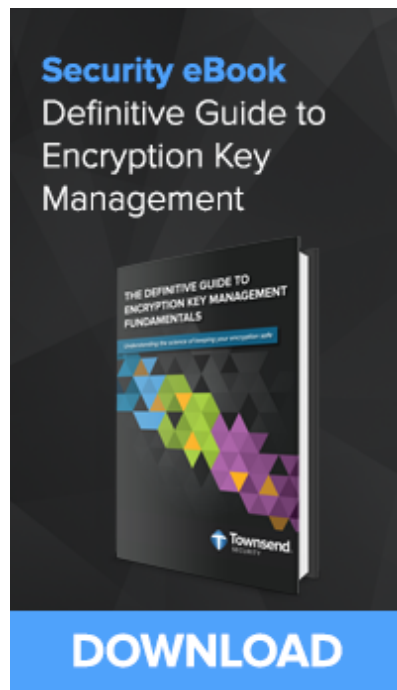
[Communication Protocols](#)

[Platforms for Housing the Encryption Key Manager](#)

[Encryption Key Management in Meeting Compliance](#)

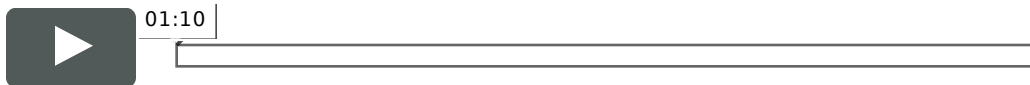
[Bonus Content: A Brief History - the Need for Encryption Key Management](#)

[Click here to view this eBook offline](#)



[^Back to Top](#)

INTRODUCTION



“The proper management of cryptographic keys is essential to the effective use of cryptography for security. Keys are analogous to the combination of a safe. If a safe combination is known to an adversary, the strongest safe provides no security against penetration. Similarly, poor key management may easily compromise strong algorithms.”

~ [NIST Recommendation for Key Management](#)

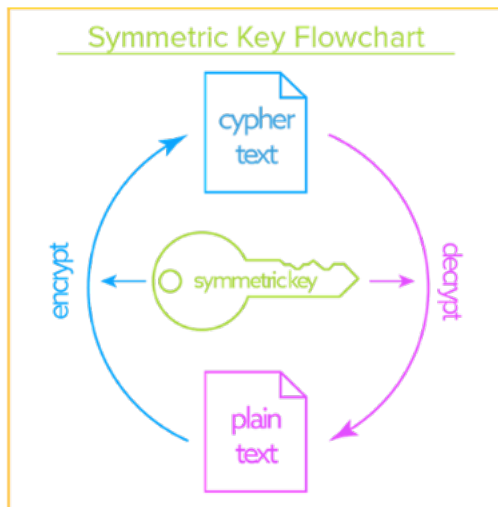
NIST’s statement paints an accurate picture. Like a safe’s combination, your encryption keys are only as good as the security you use to protect them. There is an entire physical and digital cryptosystem that must be must be accounted for as well as each key’s full lifecycle. Therefore, a robust [encryption key management](#) system and policies includes:

- Key lifecycle: key generation, pre-activation, activation, expiration, post-activation, escrow, and destruction
- Physical access to the key server(s)
- Logical access to the key server(s)
- User/Role access to the encryption keys

Let's get started with a brief overview of the types of encryption keys.

[^Back to Top](#)

TYPES OF ENCRYPTION KEYS

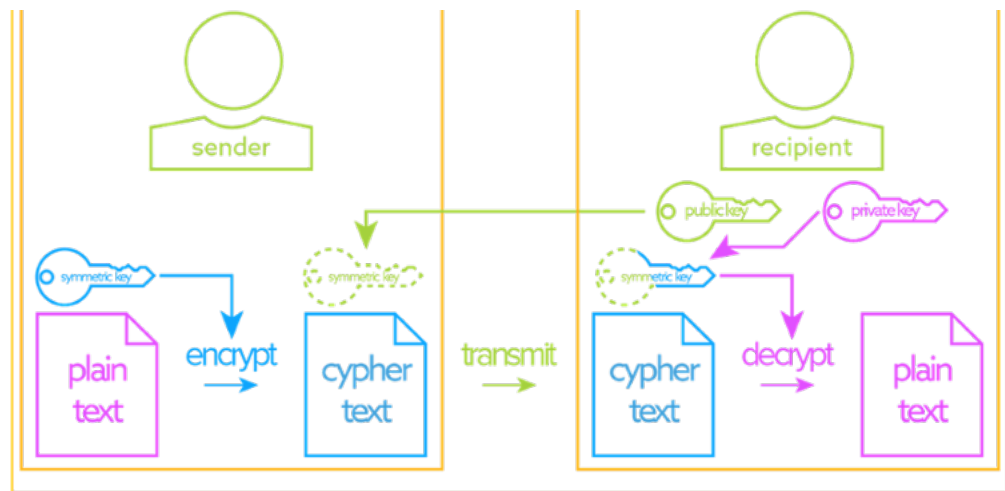


Symmetric Keys: Data-at-Rest

In symmetric key cryptography, the same encryption key is used to both encrypt and decrypt the data. This means of encryption is used primarily to protect data at rest. An example would be to encrypt sensitive data into ciphertext while it is stored in a database and decrypt it to plaintext when it is accessed by an authorized user, and vice versa.

versa.

Asymmetric Key Flowchart



Asymmetric Keys: Data-in-Motion

Asymmetric keys, on the other hand, are a pair of keys for the encryption and decryption of the data. Both keys are related to each other and created at the same time. They are referred to as a public and a private key:

- **Public Key:** this key is primarily used to encrypt the data and can be freely given as it will be used to encrypt data, not decrypt it.
- **Private Key:** this key is used to decrypt the data that its counterpart, the public key, has encrypted. This key must be safeguarded as it is the only key that can decrypt the encrypted data.
- Asymmetric keys are primarily used to secure data-in-motion. An example might be a virtual private network (VPN) connection. With a VPN:
 - an AES symmetric session key is used to encrypt the data
 - a public key is used to encrypt the session key
 - once the encrypted data is received, the private key is used to decrypt the session key
 - so that it can be used to decrypt the data.

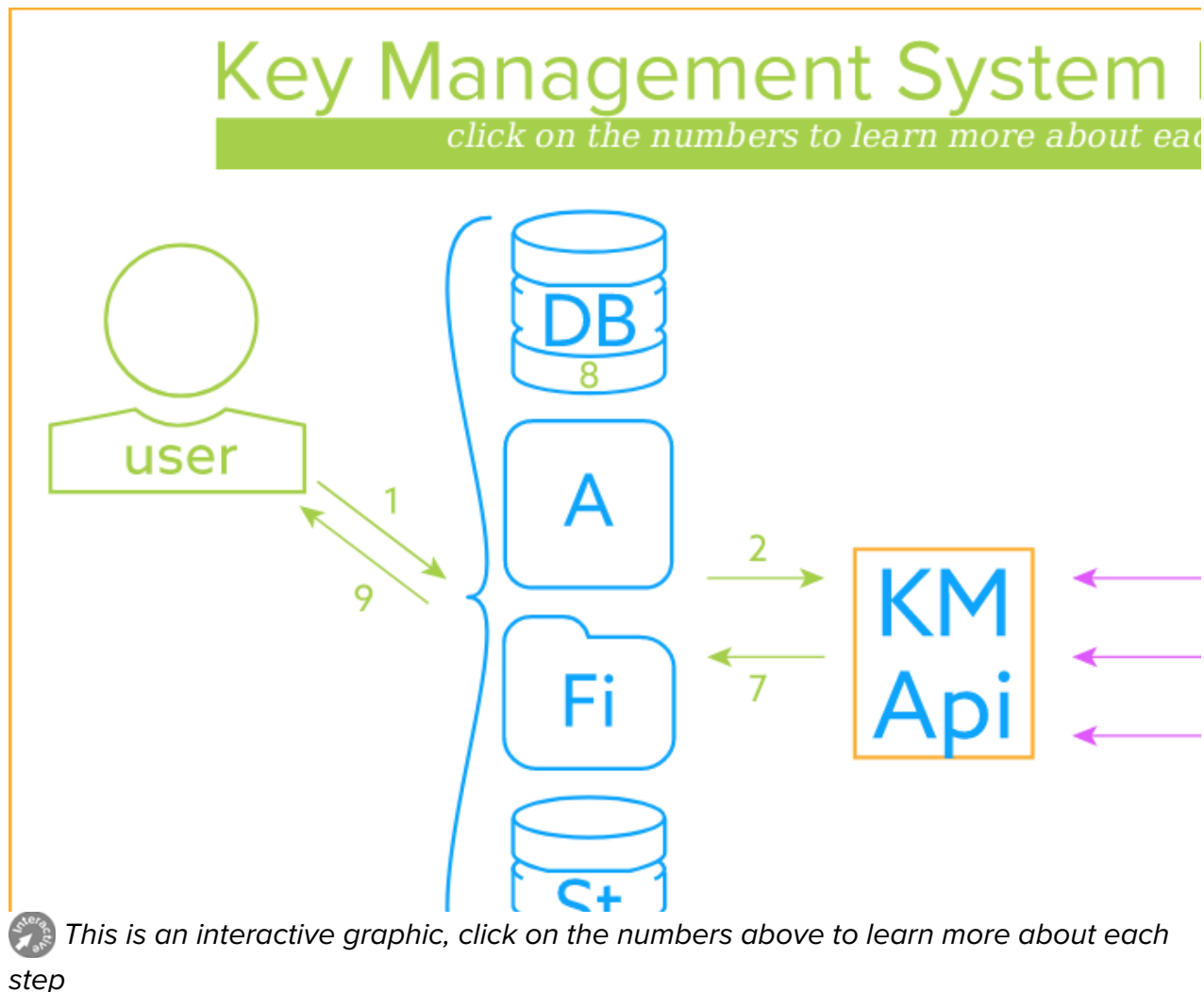
[^Back to Top](#)

HOW ENCRYPTION KEY SYSTEMS WORK

Symmetric Key Systems

First, let's establish a few definitions:

- **Data encryption key (DEK):** is an encryption key whose function it is to encrypt and decrypt the data.
- **Key encryption key (KEK):** is an encryption key whose function it is to encrypt and decrypt the DEK.
- **Key management application program interface (KM API):** is an application interface that is designed to securely retrieve and pass along encryption keys from a key management server to the client requesting the keys.
- **Certificate Authority (CA):** is an entity that creates public and private keys, creates certificates, verifies certificates and performs other PKI functions.
- **Transport layer security (TLS):** is a cryptographic protocol that provides security, through mutual authentication, for data-in-motion over a computer network.
- **Key Management System (KMS):** is the system that houses the key management software

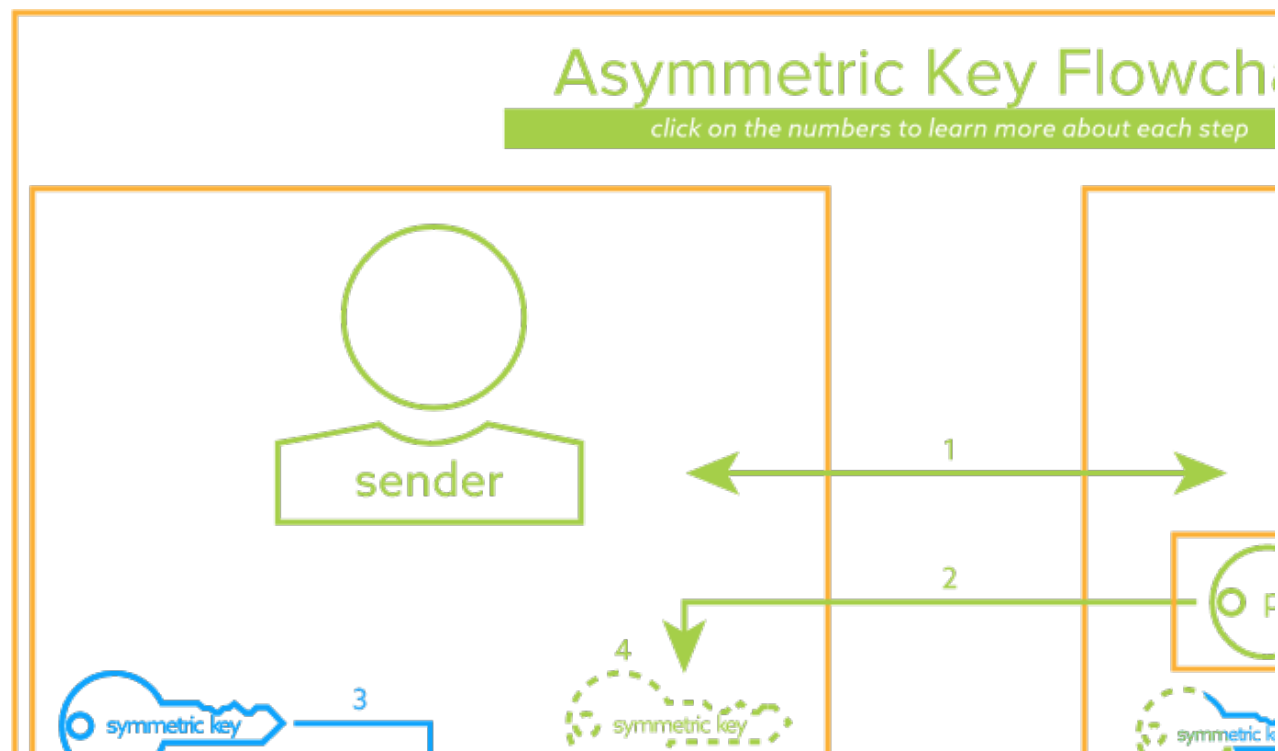



Now that we have the definitions in place, below is a step by step example of how an authorized user accesses encrypted data:

1. A user requests to access encrypted data.
2. The database, application, file system, or storage then sends a DEK retrieval request to the client (KM API).
3. Next, the client (KM API) and KM verify each other's certificates:
 - a. The client (KM API) sends a certificate to the KM for verification.
 - b. The KM then checks the certificate against the CA for authentication.
 - c. Once the client (KM API) certificate has been verified, the KM then sends its certificate to the KM API for authentication and acceptance.
4. Once the certificates have been accepted, a secure TLS connection is established between the client (KM API) and the KM.

5. The KM then decrypts the requested DEK with the KEK
6. The KM sends the DEK to the client (KM API) over the encrypted TLS session.
7. The KM API then sends the DEK to the database, application, file system, or storage.
8. The database (may) cache the DEK in temporary secure memory.
9. The database, application, file system, or storage then sends the plaintext information to the user.

Asymmetric Key Systems



 This is an interactive graphic, click on the numbers above to learn more about each step

1. The Sender and Recipient verify each other's certificates:
 - a. The sender sends a certificate to the recipient for verification.
 - b. The recipient then checks the certificate against their Certificate Authority (CA) or an external Validation Authority (VA) for authentication.
 - c. Once the sender's certificate has been verified, the recipient then sends their certificate to the sender for authentication and acceptance.

2. Once the sender and recipient have mutual acceptance:
 - a. The sender requests the recipient's public key.
 - b. The recipient sends their public key to the sender.
3. The sender creates an ephemeral symmetric key and encrypts the file to be sent. (an ephemeral symmetric key is a symmetric encryption key used only for one session)
4. The sender encrypts the symmetric key with the public key.
5. The sender then sends the encrypted data with the encrypted symmetric key.
6. The recipient receives the packet and decrypts the symmetric key with the private key.
7. The recipient decrypts the data with the symmetric key.

[^Back to Top](#)

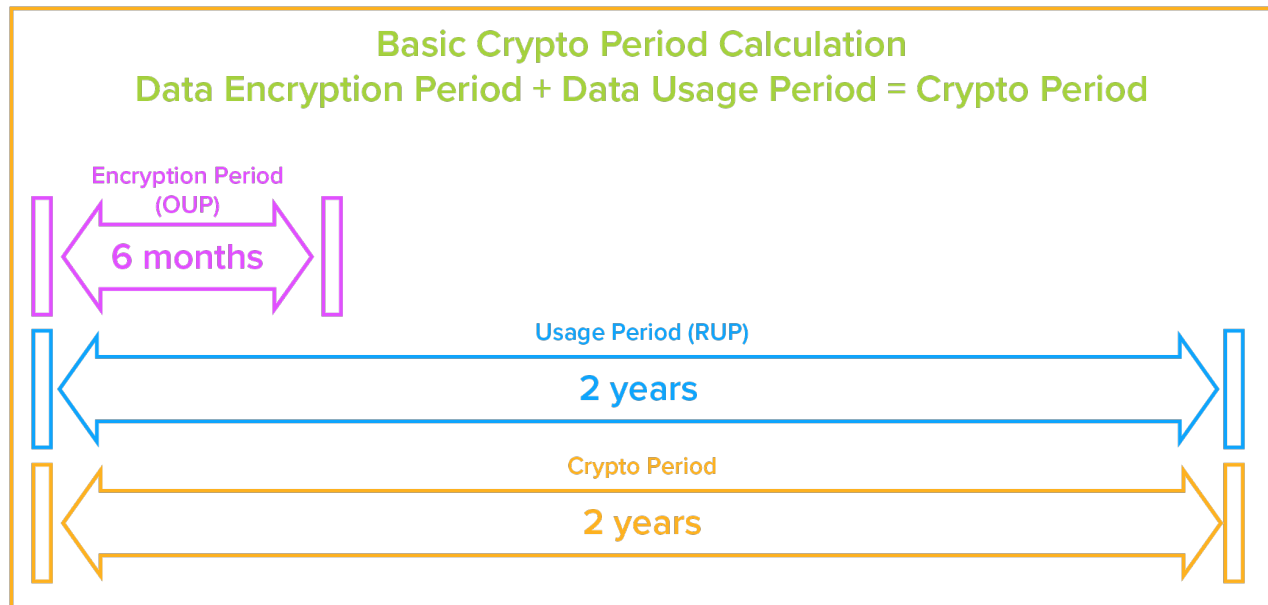
THE FULL LIFE-CYCLE OF KEYS

The encryption key life-cycle, defined by [NIST](#) as having a pre-operational, operational, post-operational, and deletion stages, requires that, among other things, a operational crypto period be defined for each key. A [crypto period](#) is the "time span during which a specific key is authorized for use" and in Section 5.3 of [NIST's Guide](#), the crypto period is determined (for example, with a symmetric key) by combining the estimated time during which encryption will be applied to data (the *Originator Usage Period (OUP)*) and the time when it will be decrypted for use (the *Recipient Usage Period (RUP)*).

So, as an example:

- let's say that a database is encrypted and for the next 6 months items are added to it. Then:
 - the OUP is 6 months

- For 2 years the database is also viewed by authorized users. Then:
 - the RUP is 2 years (and completely overlaps with the OUP)
- Therefore, the crypto period would equal 2 years and the encryption key would need to be active during that time.



But, since an organization may reasonably want to encrypt and decrypt the same data for years on end, other factors may come into play to when factoring the crypto period:

You may want to limit the:

- "amount of information protected by a given key"
- "amount of exposure if a single key is compromised"
- "time available for attempts to penetrate physical, procedural, and logical access"
- "period within which information may be compromised by inadvertent disclosure"
- "time available for computationally intensive cryptanalytic attacks"

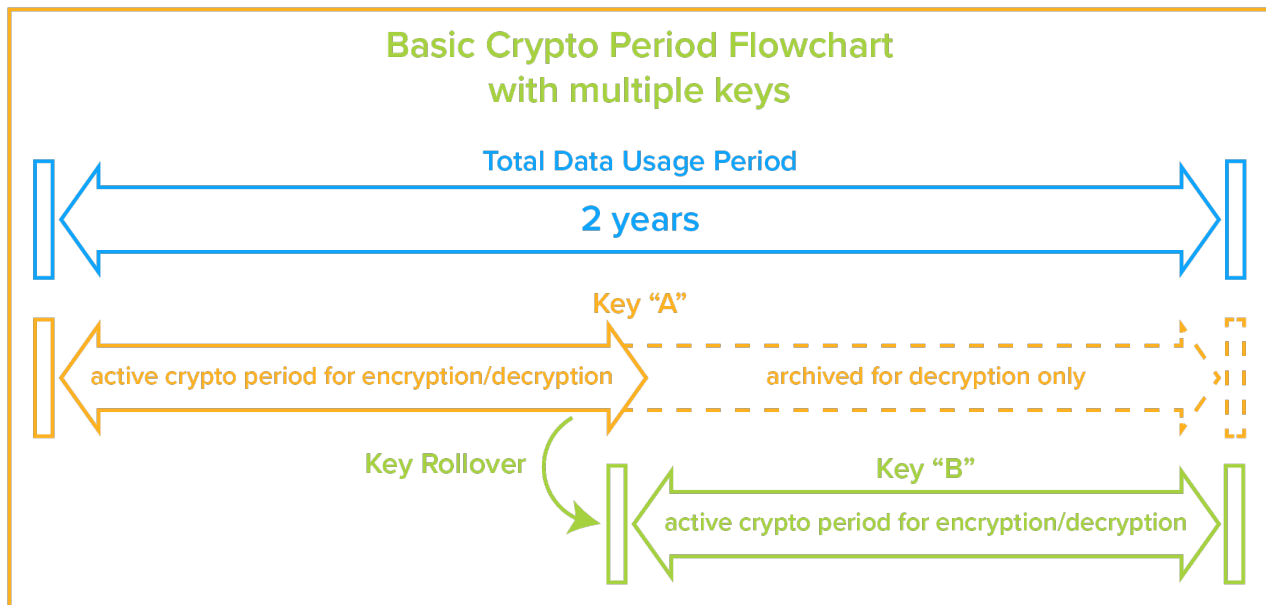
This can be boiled down to a few key questions:

- How long will the data be used
- How is the data being used
- How much data is there

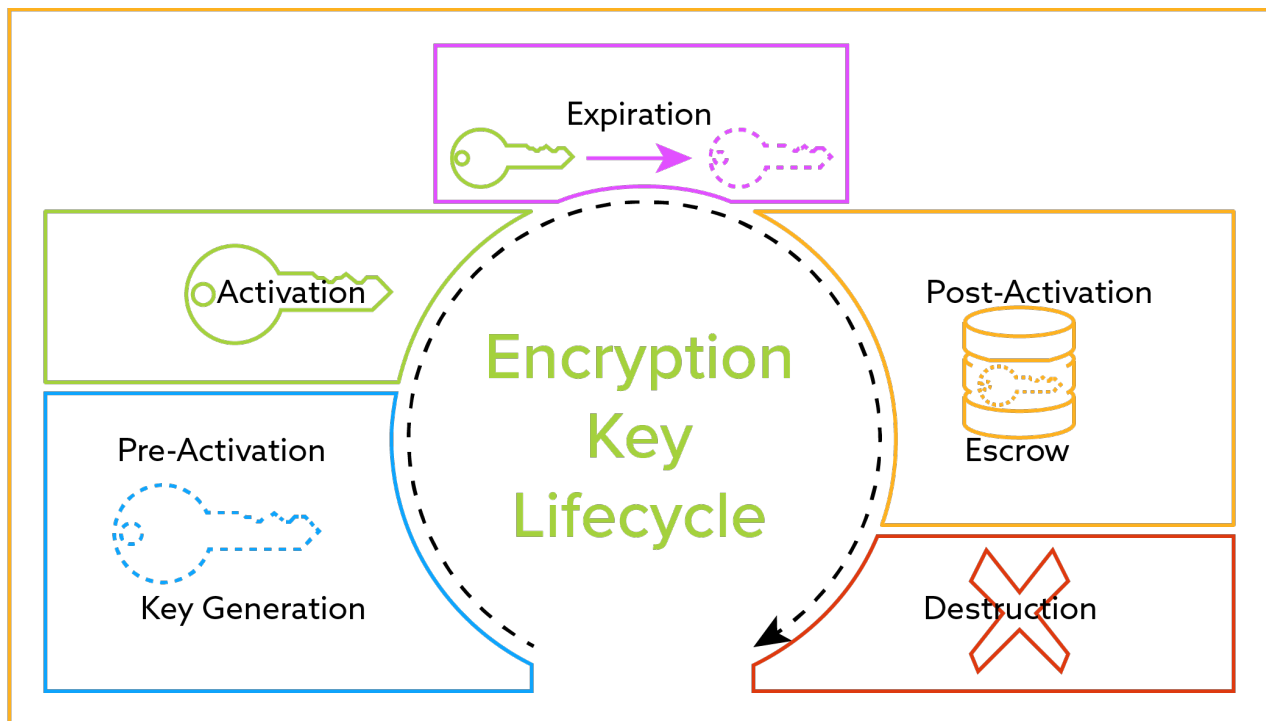
- How sensitive is the data
- How much damage will be done when the data is exposed or the keys are lost

The general rule: as the sensitivity of data being secured increases, the lifetime of an encryption key decreases.

Given this, your encryption key may have an active life shorter than an authorized user's access to the data. This means that you will need to archive de-activated keys and use them only for decryption. Once the data has been decrypted by the old key, it will be encrypted by the new key, and over time the old key will no longer be used to encrypt/decrypt data and can be deleted. (see graphic below)



See below for a more thorough understanding of a keys full life-cycle.



Key Creation (Generation & Pre-Activation)

The **encryption key is created** and stored on the key management server. The key manager creates the encryption key through the use of a cryptographically secure random bit generator and stores the key, along with all its attributes, into the key storage database. The attributes stored with the key include its name, activation date, size, instance, the ability for the key to be deleted, as well as its rollover, mirroring, key access, and other attributes. The key can be activated upon its creation or set to be activated automatically or manually at a later time. The encryption key manager should track current and past instances (or versions) of the encryption key. You need to be able to choose whether or not the key can be deleted, mirrored to a failover unit, and by which users or groups it can be accessed. Your key manager should allow the administrator to change many of the key's attributes at any time.

Key Use and Rollover (Activation through Post-Activation)

The key manager should allow an activated key to be retrieved by authorized systems and users for encryption or decryption processes. It should also seamlessly manage current and past instances of the encryption key. For example, if a new key is generated and the old one deactivated (or rolled) every year, then the key manager should retain previous versions of the key but dispense only the current instance and activate previous versions for decryption processes. Previous versions can still be retrieved in order to decrypt data encrypted with such versions of the key. The key manager will also roll the key either through a previously established schedule or allow an administrator to manually roll the key.



Key Revocation

An administrator should be able to use the key manager to revoke a key so that it is no longer used for encryption and decryption requests. A revoked key can, if needed, be reactivated by an administrator so that, In certain cases the key can be used to decrypt data previously encrypted with it, like old backups. But even that can be restricted.

Back Up (Escrow)

[NIST \(Section 8.3.1\)](#) requires that an archive should be kept for deactivated keys. The archive should “protect the archived material from unauthorized [disclosure,] modification, deletion, and insertion.” The encryption keys need “to be recoverable ... after the end of its cryptoperiod” and “the system shall be designed to allow reconstruction” of the keys should they need to be reactivated for use in decrypting the data that it once encrypted.

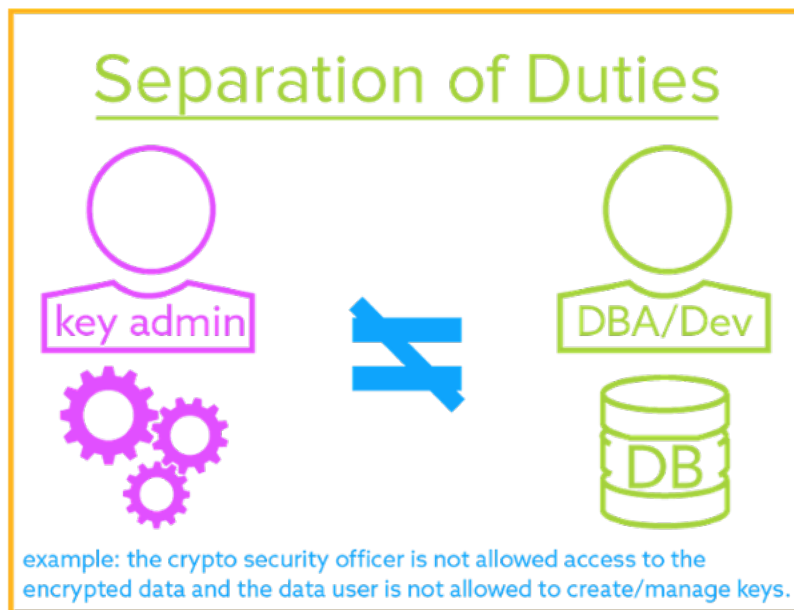
Key Deletion (Destruction)

If a key is no longer in use or if it has somehow been compromised, an administrator can choose to delete the key entirely from the key storage database of the encryption key manager. The key manager will remove it and all its instances, or just certain instances, completely and make the recovery of that key impossible (other than through a restore

from a backup image). This should be available as an option if sensitive data is compromised in its encrypted state. If the key is deleted, the compromised data will be completely secure and unrecoverable since it would be impossible to recreate the encryption key for that data.

[^Back to Top](#)

SEGREGATED ROLES IN KEY MANAGEMENT



Separation of Duties

In “[Recommendation for Key Management – Part 2](#)” NIST defines Separation of Duties as:

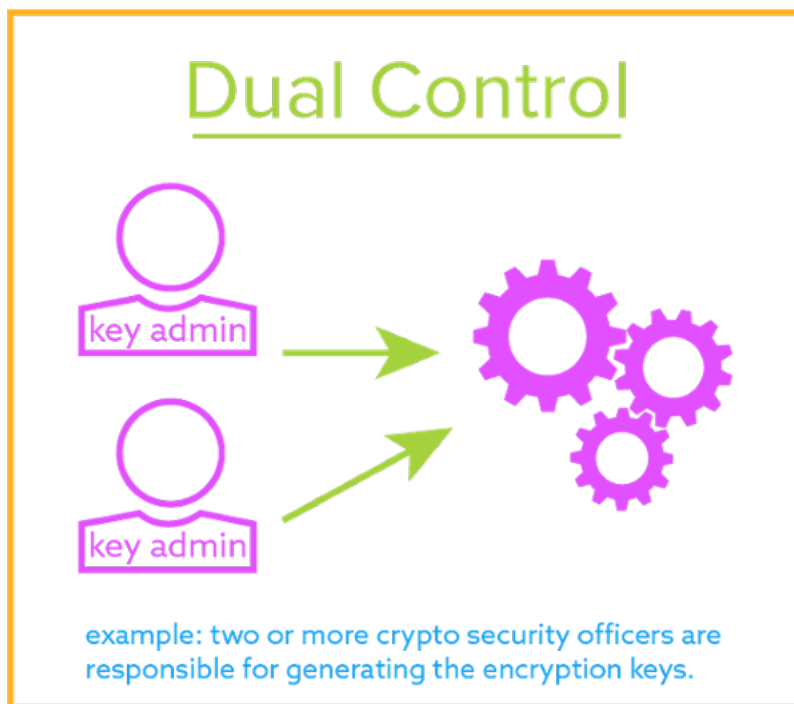
A security principle that divides critical functions among different staff members in an attempt

to ensure that no one individual has enough information or access privilege to perpetrate damaging fraud.

The practice of Separation of Duties reduces the potential for fraud or malfeasance by dividing related responsibilities for critical tasks between different individuals in an

organization. It is common in the financial and accounting procedures of most organizations. For example, the person who prints the checks at a company would not be the person who signs the checks. Similarly, the individual who signs checks would not reconcile the bank statements. A company would ensure that business critical duties are categorized into four types of functions: authorization, custody, record keeping, and reconciliation. In a perfect system, no one person should handle more than one type of function.

Regarding information security practices, the implementation of Separation of Duties is critical in the area of encryption key management. To prevent unwanted access to protected data, it is important that the person who manages encryption keys not have the ability to access protected data, and vice versa. This is no more difficult to accomplish in an information technology context than in a financial context, but is often overlooked or misunderstood in complex computer systems.



Dual Control

Again, NIST, in [Recommendation for Key Management – Part 2](#), defines Dual Control:

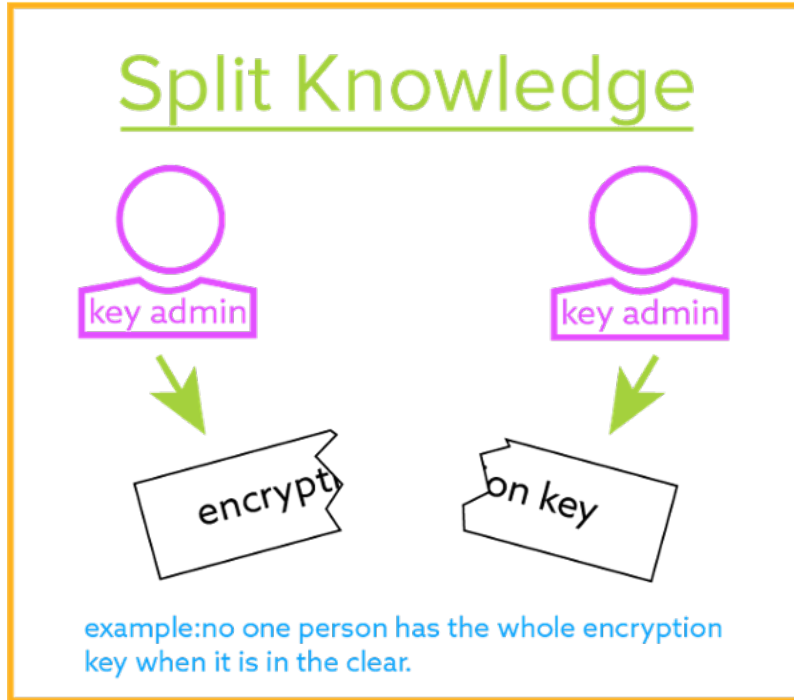
A process that uses two or more separate entities (usually persons) operating in concert to protect sensitive functions or information. No single entity is able to access or use the materials, e.g.,

cryptographic keys.

While Separation of Duties involves distributing different parts of a process to different people, Dual Control requires that at least two or more individuals control a single process.

In data security practice it is common to find requirements for Dual Control of encryption

key management functions. Because a key management system may be storing encryption keys for multiple applications and business entities, the protection of encryption keys is critically important.



Split Knowledge

The concept of Split Knowledge applies to any access or handling of unprotected cryptographic material like encryption keys or passphrases used to create encryption keys, and requires that no one person know the complete value of an encryption key. If passphrases are used to create encryption

keys, no one person should know the entire passphrase. Rather, two or more people should each know only a part of the pass phrase, and all of them would have to be present to create or recreate an encryption key.

[^Back to Top](#)

THE DOMAINS TO SECURE

ENCRYPTION KEYS

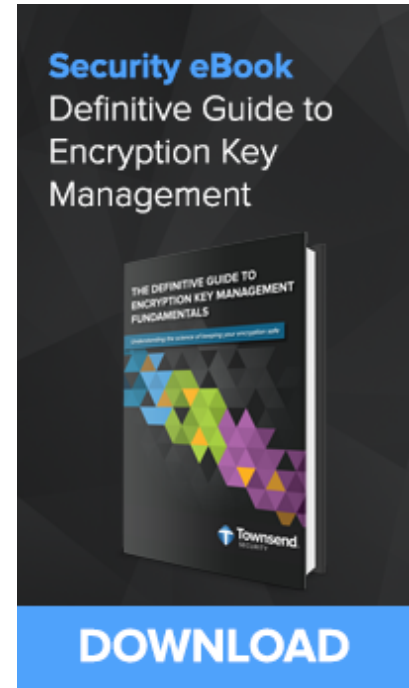
Physical Security

Many, when talking about securing a key manager, will naturally turn to securing the key manager itself with a hardware security module (HSM). While that is a necessary topic (and we will discuss it), we should first talk about securing the physical environment in which your key manager is housed.

In NIST's Special Publication 800-14, they offer this definition of physical security:

"Physical and environmental security controls" should be "implemented to protect the facility housing system resources, the system resources themselves, and the facilities used to support their operation."

[Click here to view this eBook offline](#)



An organization's physical security plan need to include things like:

- **Physical access controls:** limit access to critical systems, including locations of wiring connecting to the system, to as few people as possible.
- **Ports:** FIPS 140-2 notes that in the case sending plaintext encryption keys, physical ports should be dedicated for only that purpose, and all other use excluded for level 3 and 4 cryptographic modules.
- **Fire safety:** make sure all physical environments housing the system have adequate, and current, fire suppression systems.
- **Structural integrity:** ensure that all physical environments housing the system meet current earthquake, flooding, and snow load for roofing regulatory requirements.
- **Utilities failure:** systems such as electricity, air conditioning, and heating can malfunction. Ensure that each is functioning properly with back-ups in place, where necessary.
- **Interception of data:** ensure that all transmission of sensitive data is properly encrypted with public/private keys.

- **Mobile device management:** all devices that can remotely access the system should be cataloged and managed in a permissions database.

Now comes securing the cryptographic module itself. The Federal Information Processing Standards (FIPS) has identified four levels of increasing security in [FIPS 140-2](#) that can be applied to the module, each corresponding to the commensurate threat level:

- **Level 1:** “No specific physical security mechanisms are required in a Security Level 1 cryptographic module beyond the basic requirement for production-grade components. ... Security Level 1 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an unevaluated operating system.”
- **Level 2:** “enhances the physical security mechanisms of a Security Level 1 cryptographic module by adding the requirement for tamper-evidence, which includes the use of tamper-evident coatings or seals or for pick-resistant locks on removable covers or doors of the module.”
- **Level 3:** “attempts to prevent the intruder from gaining access to [Critical security parameters (CSPs)] held within the cryptographic module. ... The physical security mechanisms may include the use of strong enclosures and tamper detection/response circuitry that zeroizes all plaintext CSPs when the removable covers/doors of the cryptographic module are opened.”
- **Level 4:** “provides the highest level of security defined in this standard. At this security level, the physical security mechanisms provide a complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access. Penetration of the cryptographic module enclosure from any direction has a very high probability of being detected, resulting in the immediate zeroization of all plaintext CSPs.”

How an Encryption Key Manager is Validated

Every data security product available makes claims as to superior functionality or data protection. But when protecting sensitive data, organizations need to have assurance that a product's stated security claim is valid. This is certainly true when it comes to an encryption key manager. To address this, NIST has devised a system to validate cryptographic modules and ensure that they comply with FIPS 140-2 standards. Here are the steps an encryption key manager vendor must take to show full compliance:

1. First they will contract with an [accredited laboratory](#), who has successfully undergone the [National Voluntary Laboratory Accreditation Program \(NVLAP\)](#), to conduct “adequate testing and validation of the cryptographic module and its underlying cryptographic algorithms against established standards” to look for “weaknesses such as poor design or weak algorithms.”
2. Next, the accredited laboratory will conduct the [Cryptographic Algorithm Validation Program \(CAVP\)](#). With this testing, they will provide “validation testing of FIPS-approved and NIST-recommended cryptographic algorithms and their individual components.”
3. Once that testing is complete and the key manager has meet all standards, the lab will then move on to the [Cryptographic Module Validation Program \(CMVP\)](#) testing. The “laboratories use the [Derived Test Requirements \(DTR\)](#), [Implementation Guidance \(IG\)](#) and applicable CMVP programmatic guidance to test cryptographic modules against the applicable standards.”
4. Finally, once the encryption key manager has been shown to meet all FIPS 140-2 standards, the independent lab issues the FIPS 140-2 Validation Certificate and the cryptographic module is placed on the [FIPS 140-1 and FIPS 140-2 Vendor List](#).

Logical Access Security

The next arena in which you can protect your encryption keys is by logically separating the different cryptographic components housing the keys from the rest of the larger network. There are three main items to consider:

- **Interfaces:** In [FIPS 140-2](#), Section 4.2, it gives this criteria for needing to separate logical interfaces:
 - **Level 1 and 2:** the “logical interface(s) used for the input and output of plaintext cryptographic keys, cryptographic key components, authentication data, and CSPs may be shared physically and logically with other ports and interfaces of the cryptographic module.”
 - **Level 3 and 4:** “the logical interfaces used for the input and output of plaintext



cryptographic key components, authentication data, and CSPs shall be logically separated from all other interfaces using a trusted path.”

- **DEK from encrypted data:** In level 1 environments, where the encryption key manager is not in a physically separated HSM, the DEK(s) should be logically separated from the data that is encrypted. This effectively keeps the DEK(s) from being used to decrypt the data in case unauthorized users gain access to the sensitive material.
- **KEK from DEK:** Within the encryption key manager, the KEK(s) should be logically separated from the DEK(s). This ensures that though the database DEKs be compromised, they will be rendered unusable because the KEK is in a logically separate location from the DEKs.

User/Role Access

Once Physical Security and Logical Security are addressed, the final component is user roles and privileges. The core concept promulgated by NIST is the concept of least privilege: where you restrict “the access privileges of authorized personnel (e.g., program execution privileges, file modification privileges) to the minimum necessary to perform their jobs.”

NIST gives guidance, in Sections 5.3.5 of [Recommendation for Key Management – Part 2](#), on the access controls and privileges necessary to properly manage user access to the key management system.

- Document and implement which roles within the organization will be authorized to access the KMS and to what level.
- What functions will the role be able to execute on (i.e. generation, handling, distribution, storage, deletion).
- What means of authentication will be used (i.e. passwords, personal identification numbers, biometrics, and their expiration dates).

Beyond limiting access to the key management server, you should also limit access to the keys themselves based on user and group. The users and group access can be defined on a system level, or at the level of each key. When you create a key you can define the restrictions on user and group access. As an example: There is an AES encryption key available on the key management server used to protect an employee's personal data. It is restricted so that only members of the Human Resources group can use that key. So any

individual with "Human Resources" defined as their individual or group role can successfully request that key, all others are turned away.

High Availability and Business Continuity

Once you have physical security, logical security, and user roles in place, you must also consider business continuity. If an intruder does compromise your data or your production server(s) are taken offline for a variety of reasons, you must be able to bounce back in a relatively short time with pre-prescribed steps. Here are a couple definitions to start us off:

Business Continuity: As defined by [ISO 22301:2012 \(Section 3.3\)](#), it is the “capability of the organization to continue delivery of products or services at acceptable” levels after a “disruptive incident.”

Hot failover: In a network environment, a hot failover is switching to a backup server that is regularly updated from the production server and is ready, at any time, should the production server no longer be able to function normally for any length of time.

In the case of key management, each production key management server should be mirrored with a high availability server in a geographically separate location in case the production server is compromised and taken offline for any length of time. As an abbreviated list, here are some features to look for in key management solutions or what you will want to address if you build your own:

- Hardware - hot swappable RAID disk drives
- Hardware - dual redundant power supplies
- Hardware - independent network interfaces
- Active-Active secure key server mirroring
- Active-Passive secure key server mirroring
- Real time key mirroring
- Real time access policy mirroring
- Key manager integrity checking on startup
- Key retrieval integrity checking

[^Back to Top](#)

PLATFORMS FOR HOUSING THE KEY MANAGER

HSM

The hardware security module (HSM) has been discussed already in “Physical Security” mostly referred to as the “cryptographic module.” But, to summarize, a HSM is typically a server with different levels of security protection or “hardening” that prevents tampering or loss. These can be summarized as:

- **Tamper Evident:** adding tamper-evident coatings or seals on screws or locks on all removable covers or doors
- **Tamper Resistant:** adding “tamper detection/response circuitry” that wipes out all sensitive data such as DEKs and KEKs
- **Tamper Proof:** complete hardening of the module with tamper evident/resistant screws and locks along with the highest sensitivity to “tamper detection/response circuitry” that wipes out all sensitive data

Hosted HSM

With many organizations moving some or all of their operations to the cloud, the need for moving their security has also arisen. The good news, many key management providers have partnered with cloud hosting providers to rack up traditional HSMs in cloud environments. The same levels of “hardening” would still apply, as it is a traditional HSM in an offsite environment.

Virtual

Virtual instances of an encryption key manager offer a great deal more flexibility than their HSM counterparts. In many cases, a virtual key manager can be downloaded from a vendor in a matter of minutes and deployed in a virtual environment. An HSM, on the other hand, can take days or weeks being shipped to the site and then requires a physical installation. Further, virtual instances can be installed anywhere that supports the virtual platform that the key manager runs in, VMware, as an example.

The downside, of course, is that by its nature of being virtual with no set physical components, a virtual key manager's software can only be FIPS 140-2 compliant, but not validated. So, if your business need(s) or compliance regulation(s) require FIPS 140-2 validation, then a HSM is your only option.

That being said, the logical security that FIPS 140-2 compliant virtual key managers provide is normally more than enough for most organizational needs.

AWS, Microsoft Azure, and More: Dedicated or “as a Service”

Cloud providers, such as Amazon Web Services (AWS), Microsoft Azure (Azure), and more have marketplace offerings for encryption key management as well as their own key management as a service (KMaaS). AWS and Azure's KMaaS is typically multi-tenant, meaning more than one user's key(s) are present on the same key management instance. This can raise concerns for organizations that need dedicated services to mitigate security concerns of other users accessing the same key data stores.

To combat this issue, most cloud providers will also offer dedicated services. In their marketplaces, there are also independent vendors that provide dedicated services that typically come in two forms: Pay-Per-Usage and “bring your own license.” Townsend Security provides for both platforms and for both licensing models: [Alliance Key Manager for AWS](#) and [Alliance Key Manager for Azure](#). Both the AWS and the Azure instances are dedicated key managers in an IaaS virtual instance and also enjoy the flexibility of being the same key manager that is deployed as an [HSM](#), [Cloud HSM](#), and [VMware](#) instance so

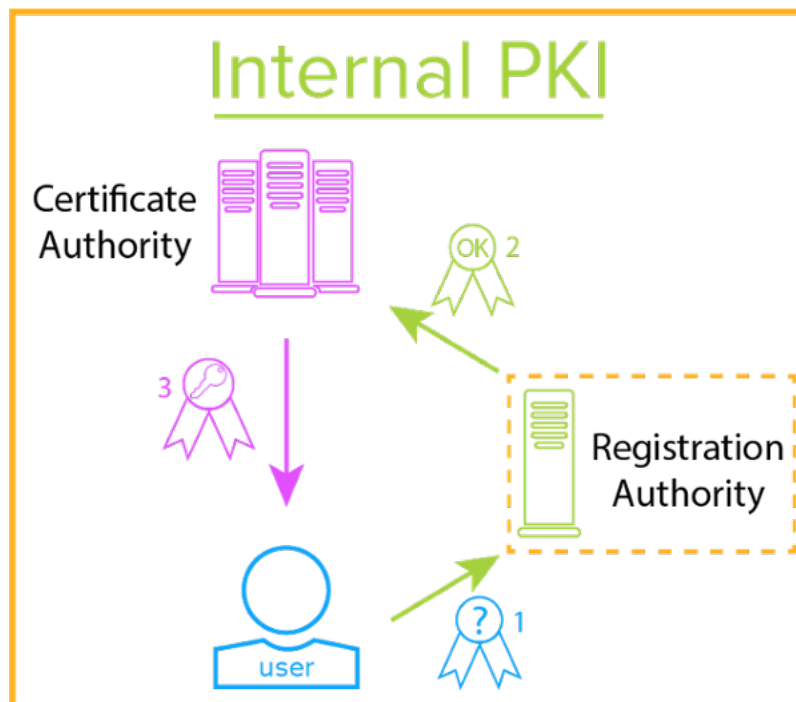


that your environment can scale past AWS and Azure, if needed. This is useful for organizations with existing (or future) physical data center(s), because having the same technology secure your data everywhere reduces complexity for your IT staff as they use and maintain it.

[^Back to Top](#)

COMMUNICATION PROTOCOLS

PKI



Public key infrastructure

(PKI): NIST defines PKI as an infrastructure that “binds public keys to entities, enables other entities to verify public key bindings, and provides the services needed for ongoing management of keys in a distributed system.” Put another way, it is a cryptographic infrastructure consisting of the software, hardware, roles, procedures, and policies needed to properly manage and

distribute public keys (such as a digital certificate) and private keys.

A very simple internal PKI installation (as shown in the graphic would flow like this:

1. A user requests a certificate.
2. The Registration Authority authenticates the user and the user's request, and once authenticated, sends the request to the Certificate Authority. (A Registration Authority is optional, the Certificate Authority can handle these requests, if necessary.)
3. The Certificate Authority receives the request and issues the certificate to the user.

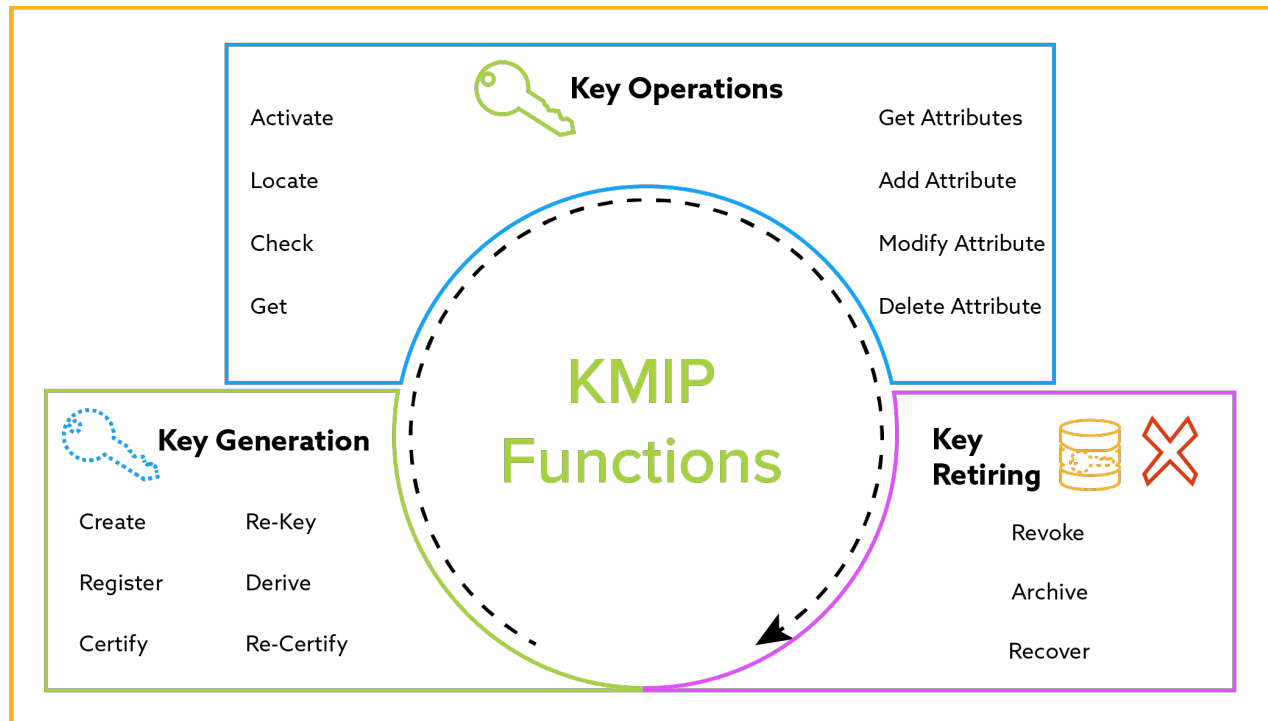
As defined by NIST in, "[Introduction to Public Key Technology and the Federal PKI Infrastructure](#)", the PKI environment consists of:

- **Certification Authority (CA):** NIST likens it "to a notary. The CA confirms the identities of parties sending and receiving electronic payments or other communications." It digitally signs and publishes the public key bound to a given user or machine and authenticates the identity of authorized users of each certificate.
- **Registration Authority (RA):** (or subordinate CA) NIST explains, it "is an entity that is trusted by the CA to register or vouch for the identity of users to a CA." It accepts requests for certificates, authenticates the user/machine making request, and issues the certificate for uses granted by the CA.
- **Central Directory:** Again, from NIST: it "is a database of active digital certificates for a CA system. The main business of the repository is to provide data that allows users to confirm the status of digital certificates for individuals and businesses that receive digitally signed messages."
- **Archive:** is a database of public keys and certificates. The archive should store sufficient information to determine if a digital signature on an "old" document should be trusted.
- **Public Key Certificate:** NIST requires one "for each identity, confirming that the identity has the appropriate credentials. A digital certificate typically includes the public key, information about the identity of the party holding the corresponding private key, the operational period for the certificate, and the CA's own digital signature."
- **Certificate Revocation List (CRL):** Simply put, a list of certificates that have been

revoked.

- **PKI Users:** NIST defines them as, “organizations or individuals that use the PKI, but do not issue certificates. They rely on the other components of the PKI to obtain certificates, and to verify the certificates of other entities that they do business with.”

KMIP



Key Management Interoperability Protocol (KMIP): As defined by OASIS, KMIP is a communication “protocol used for the communication between clients and servers to perform certain management operations on objects stored and maintained by a key management system.” This protocol is a standardized way of managing encryption keys throughout the lifecycle of the key and is designed to facilitate “symmetric and asymmetric cryptographic keys, digital certificates, and templates used to simplify the creation of objects and control their use.”

Below is a curated list of what OASIS further defines in Section 4 as what the key management client can request of the key management server:

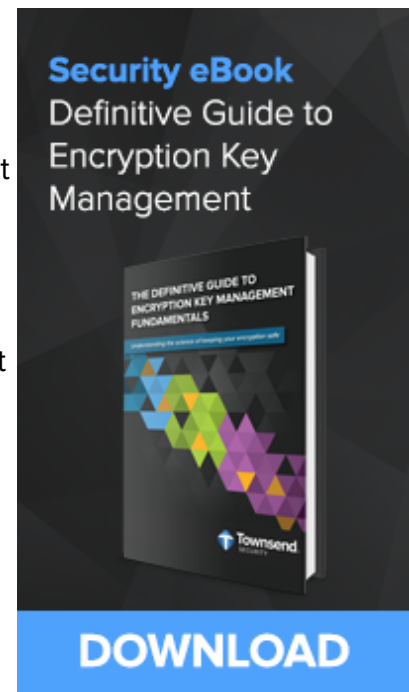
- **Create a Key or Key Pair:** “to generate a new symmetric key” or “new public/private key pair” and

[Click here to view this eBook offline](#)

register the “corresponding new Managed Cryptographic Objects.”

- **Register:** “to register a Managed Object,” typically keys, passwords, or other cryptographic materials, “that was created by the client or obtained by the client through some other means, allowing the server to manage the object.”
- **Re-Key or Re-key Key Pair:** “to generate a replacement key,” also called a key change, “for an existing symmetric key” or “key pair for an existing public/private key pair.”
- **Derive Key:** “to derive a Symmetric Key or Secret Data object from keys or Secret Data objects that are already known to the key management system.”
- **Certify or Re-certify:** “to generate a Certificate object for a public key” or “renew an existing certificate.”
- **Locate:** to “search for one or more Managed Objects, depending on the attributes specified in the request.”
- **Check:** to “check for the use of a Managed Object according to values specified in the request.”
- **Get or Get Attributes:** to return “the Managed Object specified by its Unique Identifier” or request “one or more attributes associated with a Managed Object.”
- **Add, Modify, or Delete Attribute:** to add, modify, delete an “attribute instance to be associated with a Managed Object and set its value.”
- **Activate:** “to activate a Managed Cryptographic Object.”
- **Revoke:** “to revoke a Managed Cryptographic Object or an Opaque Object.”
- **Destroy:** “that the key material for the specified Managed Object SHALL be destroyed.”
- **Archive:** “to specify that a Managed Object MAY be archived.”
- **Recover:** “to obtain access to a Managed Object that has been archived.”

For further reading on KMIP, try the [KMIP Usage Guide Version 1.2](#), Edited by Indra Fitzgerald and Judith Furlong.



[^Back to Top](#)

ENCRYPTION KEY MANAGEMENT IN MEETING COMPLIANCE

PCI DSS



[Standard \(PCI DSS\)](#) is a widely accepted set of regulations intended to secure credit, debit and cash card transactions and cardholder data. PCI DSS requires that merchants protect sensitive cardholder information from loss and use good security practices to detect and protect against security breaches.

In **Section 3.5 of PCI DSS**, organizations that process, store, or transmit cardholder data should, “document and implement procedures to protect keys used to secure stored cardholder data against disclosure and misuse.” This includes:

- maintaining “a documented description of the cryptographic architecture” used to protect the data
- restricting “access to cryptographic keys to the fewest number of custodians necessary”
- store encryption keys “in one (or more) of the following forms at all times:”
 - encrypt the data encryption key with a key encryption key



- within a secure cryptographic device

Likewise, **Section 3.6** requires that you “fully document and implement all key management processes and procedures for cryptographic keys used for encryption of cardholder data.” This includes securely:

- generating cryptographically strong encryption keys
- secure distribution of keys
- secure storage of keys
- establishment of cryptoperiods for all keys
- retiring and destroying the keys

HIPAA HITECH

HIPAA

The [Health Insurance Portability and Accountability Act \(HIPAA\)](#) and the [Health Information Technology for Economic and Clinical Health \(HITECH\) Act](#) both seek greater adoption and meaningful use of

health information technology. Both also lay out guidelines and regulations for proper data security around Electronic Protected Health Information (ePHI). Compliance with the HIPAA Security Rules and HIPAA Privacy Rules for ePHI requires the use of security technologies and best practices to demonstrate strong efforts towards complying with this federal regulation.

SOX

SOX

The [Sarbanes-Oxley \(SOX\) Act](#) was passed to protect investors from the possibility of fraudulent accounting activities by corporations. The Sarbanes-Oxley Act (SOX) mandated strict reforms to improve financial disclosures from

corporations and prevent accounting fraud. Sections 302, 304, and 404 of the Sarbanes-

Oxley Act mandate that organizations build, maintain, and annually report on the data security and internal controls used safeguard their sensitive data from misuse and fraud.

Cloud Security Alliance



While the [Cloud Security Alliance](#) is not a governmental agency able to levy fines for non-compliance of their standards, it is an not-for-profit organization of cloud vendors, users, and security experts whose mission is “To promote the use of

best practices for providing security assurance within Cloud Computing, and provide education on the uses of Cloud Computing to help secure all other forms of computing.”

They currently have over 80,000 members and growing. So conforming to their standards is in the best interest of many companies worldwide.

As a part of this mission the organization has published a document, “[Security Guidance For Critical Areas of Focus In Cloud Computing](#),” to help vendors and customers achieve more secure applications in cloud environments. The published guidance is now in its third edition and is available from the organization’s web site. The guidance provides recommendations for encryption key management in the section “Domain 11 – Encryption and Key Management”.

Domain 11 - Encryption & Key Management

Here are the three main points that the CSA stresses for encryption key management:

- **Secure key stores.** Key stores must themselves be protected, just as any other sensitive data. They must be protected in storage, in transit, and in backup. Improper key storage could lead to the compromise of all encrypted data.
- **Access to key stores.** Access to key stores must be limited to the entities that specifically need the individual keys. There should also be policies governing the key stores, which use separation of roles to help control access; an entity that uses a given key should not be the entity that stores that key.
- **Key backup and recoverability.** Loss of keys inevitably means loss of the data that those keys protect. While this is an effective way to destroy data, accidental loss of

keys protecting mission critical data would be devastating to a business, so secure backup and recovery solutions must be implemented.

Here also is a curated list of their requirements for encryption and key management:

- “In order to maintain best practices ... the organization should manage their keys in the custody of their own enterprise or that of a credible service.”
- “Keys used in existing encryption technology ... should be managed by central, internal to the enterprise, key storage technology.”
- “Manage keys used by the cryptographic processes using binding cryptographic operations.”
- “Binding cryptographic operations and key management to corporate identity systems will provide the organization with the most flexible integration.”

EU GDPR



The new [European Union General Data Protection Regulation \(EU GDPR\)](#) has now passed both the EU Council and Parliament and replaces the earlier Data Protection Directive (Directive 94/46/EC). In Provision 83 it states:

In order to maintain security and to prevent processing in infringement of this Regulation, the controller or

processor should evaluate the risks inherent in the processing and implement measures to mitigate those risks, such as encryption.

Article 32 also calls for “the pseudonymisation and encryption of personal data.” If an organization does so, Article 34 states that the strict data breach disclosure laws of Article 33 will not be enforced if,

the controller has implemented appropriate technical and organisational protection measures, and those measures were applied to the personal data affected by the personal data breach, in particular those that render the

personal data unintelligible to any person who is not authorised to access it, such as encryption.

The GDPR places a high priority on protecting data at rest with encryption. Since encryption key management is part of an overall encryption strategy, it should be considered part in parcel with complying with EU law.

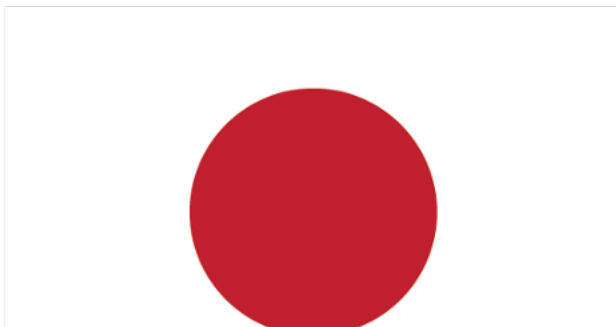
CAP 486



Hong Kong's [CAP 486 Personal Data \(Privacy\) Ordinance](#) requires that all practical steps will be taken to ensure that personally identifiable information, held by a data user, are protected against unauthorized or accidental access. Such considerations should be the kind of data stored and the harm that could result if any of those things should occur; the physical location where the data is

stored; and any security measures incorporated into any equipment in which the data is stored.

APPI



Japan's [Act on the Protection of Personal Information](#) contains policies that are guidelines, but not laws, governing the protection of personal information. It requires that businesses handling personal information should take all necessary and

proper measures for the prevention of leakage, loss, or damage.

PA 1988 & PA 2000



Australia's [Privacy Act of 1988](#) and the [Privacy Amendment Act of 2000](#) govern data security for the Down Under. In it, businesses must take all reasonable steps to protect personally identifiable information in its databases from abuse or theft. An organization must also destroy or permanently de-identify

personal information if it is no longer needed.

[^Back to Top](#)

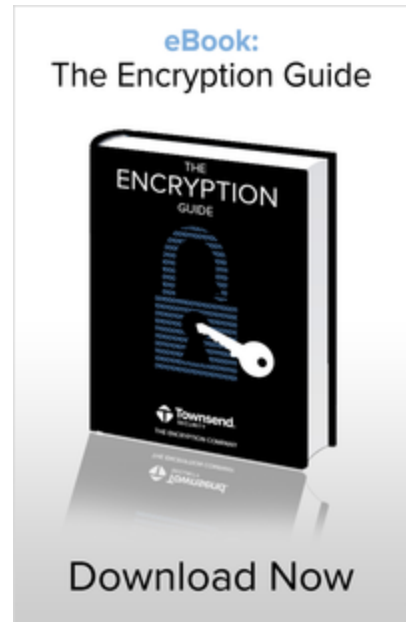
BONUS CONTENT

A Brief History - the Need for

Encryption Key Management

Encryption has been around for millennia. Some of the earliest mentions of it come from the [Arthashastra](#), a treatise on Imperial Indian governance written c2nd century BCE. In it, it describes giving messages to state spies in "[secret writing](#)". Later, and in arguably the most famous form of ancient encryption, Julius Caesar sent messages to his battle front generals in code. Known as the [Caesar Cipher](#), it is a:

"substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on."



Unfortunately for Caesar, and fortunately for his opponents, once the cipher is known, all messages can be easily read. Thus rendering the cipher useless. There needed to be a better way.

Fast forward to the electronic age. In the 1921 Edward Hebern patented the [Hebern Electric Super Code Cipher Machine](#). It was the first to code the message with a secret key embedded in a detachable rotor. In [recently declassified documents](#), the NSA showed that the machine enciphered the message by having the operator type the message in and the ciphertext would appear in a light-board, one letter at a time.

But since the encryption key was limited by the use of one rotor, consisting of 26 circuit points, it was ultimately broken by cryptanalysis, specifically [letter frequencies](#).

The real leap forward was the Enigma Machine of World War II, developed by the Germans in the 1920s. It used three rotors and was thought unbreakable since the Germans, during the war, changed the rotors once a day, "giving 159 million million million possible settings to choose from," estimates [Bletchley Park](#).

But, the Enigma machine was compromised by the Poles in 1932 using [mathematical techniques](#). Later, this early work was used to read encrypted messages during World War

It by, among others, [Alan Turing](#) (at Bletchley Park) and the use of the then latest data crunching computers.

Sending messages securely had come a long way from simple substitution ciphers. Keys were now being used - but they could be cracked using the brute force of the latest computers. Enter: Data Encryption Standard.

First published as the FIPS 46 standard in 1977, in 1987 the US Government, under the Computer Security Act, mandated that the National Institute of Standards and Technology (NIST) issue the [Data Encryption Standard \(DES\)](#) in which it “specifies two FIPS approved cryptographic algorithms.” It also mandated that the “DES key consists of 64 binary digits ("0"s or "1"s) of which 56 bits are randomly generated and used directly by the algorithm. The other 8 bits, which are not used by the algorithm, may be used for error detection.”

DES was considered very secure at the time. But in little more than a decade, and as computers became exponentially faster, DES keys quickly became vulnerable to brute force attacks.

Two options were proposed to address the issue around the same time. The first, introduced in 1997, was [Triple Data Encryption Algorithm \(TDEA\)](#) or as it is more commonly known: Triple Data Encryption Standard (3DES). As NIST describes the cryptographic technique:

[3DES] encrypts each block three times with the DES algorithm, using either two or three different 56-bit keys. This approach yields effective key lengths of 112 or 168 bits

But 3DES, when using only 112 bits, is still vulnerable to attacks such as [chosen-plaintext attacks](#). Also, since 3DES is a multi-step encryption process using two or three encryption keys, a stronger, more efficient method was needed.

In 1997 NIST started a process to identify a replacement for DES. NIST invited cryptography and data security specialists from around the world to participate in the discussion and selection process. Five encryption algorithms were adopted for study. Through a process of consensus the encryption algorithm proposed by the Belgian cryptographers Joan Daeman and Vincent Rijmen was selected. Prior to selection Daeman and Rijmen used the name Rijndael (derived from their names) for the algorithm.

After adoption the encryption algorithm was given the name Advanced Encryption Standard (AES) which is in common use today.

In 2000 NIST formally adopted the AES encryption algorithm and published it as a federal standard under the designation [FIPS-197](#). AES encryption uses a single key as a part of the encryption process. The key can be 128 bits (16 bytes), 192 bits (24 bytes), or 256 bits (32 bytes) in length. Given that the fastest computer would take billions of years to run through every permutation of a 256-bit key, AES is considered an extremely secure encryption standard.

This brings us to today. AES is a very sophisticated encryption standard with an encryption key and can withstand the onslaught of the fastest computers. It's only vulnerability? The encryption keys falling into the wrong hands. That is why, after you have deployed your encryption, your best line of defense is a robust encryption key management strategy.

How Long Would It Take to Run Every Possible Combination of a 256-bit AES Key



Here is the total possible combinations for 256-bit AES keys:

110,

000,000,000,000,000,
000,000,000,000,000,
000,000,000,000,000,
000,000,000,000,000,
000,000,000,000,000



The fastest super computer, however, can only process this many operations per second:

93,

000,000,000,000,000

So...



It would take about this many years to run through every possible combination of an 256-bit AES key:

37,5

00,000,000,000,000,
000,000,000,000,000,
000,000,000,000,000,
000,000

In Comparison:



The sun will enter it's Red Giant phase, expanding in size to engulf Mercury, Venus, and possibly Earth in about this many years:

5,4

00,000,000

Conclusion:

Hackers Don't Break Encryption, They Find Your Keys

Sources:
<https://en.wikipedia.org/wiki/Supercomputer> | http://www.eetimes.com/document.asp?doc_id=1279619 | <http://www.universetoday.com/12648/will-earth-survive-when-the-sun-becomes-a-red-giant/>



[Click here to view this eBook offline](#)



CONNECT +1.800.357.1019

