**The Key Management Lifecycle**

*(NOTE: This blog post was updated on 3/31/08 to properly reflect the overlap of Rotation and Expiration. The original draft published incorrectly showed overlap between Rotation and Deployment, which, upon reflection, made no sense whatsoever.)*

In my past life, I was involved in the review and management of cryptographic services, including helping define key management processes and requirements. Now that I'm back into the consulting world, I'm finding that the topic of key management and encryption requirements is one of interest to a fairly broad, and rapidly expanding audience. Let's face it: the PCI DSS requirement for encrypting data at rest has served as the catalyst for deployment of numerous crypto systems, creating a secondary risk scenario related to improper management of those systems and related crypto materials (keys).

Toward that end, I've put together here an overview of how I view the key management lifecycle. While I do not claim to be an expert in crypto systems, by any means, I hope that you will find my thoughts on this matter to be of use. If nothing else, I hope that you can use it to analyze crypto systems within your environment and help ensure that the amount of risk related to these systems and associated key management processes is acceptable, or can be revised to bring them inline with accepted risk tolerances.

In general, I view the key management lifecycle as being comprised of eight major components, as represented in the following diagram. Each component represents a set of processes that must be addressed both in documentation and in practice.

[](http://www.secureconsulting.net/2008/03/21/KM-lifecycle-cropped.jpg)

***Creation***  
The first step in the key management lifecycle is to generate the key. Key creation must be conducted in a secure environment (hardened system), and may include the need to conform to requirements for separation of duties. In most cases, the key generated will be a symmetric key (a.k.a. “shared key”). The key should be of an adequate strength, and is reliant in part on the underlying ability to generate a random number. Key strength (generally measured in number of bits) is typically based on the project valid lifetime of the data being protected, combined with the time it takes to break (via brute forcing) keys within a given crypto scheme. For example, if your data is valid for 5 years, then you want a key that can withstand brute force attacks for longer than 5 years. In addition to choosing an appropriate key strength, one should also endeavor to choose an encryption algorithm that has been subjected to peer review and academic scrutiny. Proprietary encryption schemes should be avoided, bearing in mind the mantra "if they need to keep the algorithm secret, then it probably means that it cannot withstand scrutiny and is thus a bad technique."

Once your key is generated using known good libraries that have been properly reviewed, it may then be desirable to protect it by encrypting it with the public half of an asymmetric key pair (a.k.a. public key cryptography). In environments where the encryption key must be distributed to other systems, particularly via networks, it is recommended that this practice be followed. From the standpoint of separation of duties, an organization can have one team own and manage the key generation box, but bar them access to the encryption system itself, requiring them to instead encrypt the symmetric key with the public key before providing it to the deployment team.

***Backup***  
Before a new key is installed into production, it is imperative that a backup be made. The backup could simply be a matter of writing the key to external media (e.g., CD, DVD, USB drive) and storing it in a physical vault. Or, it may be useful to back it up using an existing traditional backup solution (local or networked).

In either case (though especially for traditional backups), it is highly recommended that a second asymmetric key pair be used to protect the symmetric key. Think of this second public key pair as your “escrow” key. As with the deployment key pair, it is recommended that the public key be used to encrypt the shared key, and then it becomes important that the private half of this key pair be protected and available for recovery operations.

Another consideration in performing backups is to apply the same disaster recovery plans that you would in any other business continuity planning process. The key should be stored in and retrievable from a location that can be accessed within the time requirements specified by the appropriate business owners. Given the potential damage due to catastrophic failure within the crypto system, it is extremely important that your BCP/DR team be briefed in on the system in order to help set and managed backup and recovery requirements and processes.

***Deployment***  
As already mentioned, it is highly recommended that the new symmetric key be encrypted by a public key prior to being delivered for deployment. This simple step provides a method for separation of duties in an environment where key management activities are not strictly contained within a single system (such as with encryption appliances). Some regulations may require this separation of duties to ensure that no single entity can create a key, deploy a key, and then arbitrarily access and decrypt data without authorization. Workflow practices should be documented and enforced in all cases to support this requirement.

The objective of the deployment phase is to install the new key into the encryption environment, but it is not charged with removing the old key from that environment. Specifically, it is advisable that the new key be deployed and tested for a pre-determined period of time to ensure that operations with the new key are successful before risking a data outage. With modern appliances, these concerns have been reduced, but from a lifecycle perspective, they are still worth bearing in mind. When working with crypto systems, one must tend toward caution, lest a key be lost, effectively removing access to important data. That is to say, errors with crypto systems can become quite costly.

***Monitoring***  
Monitoring may or may not deserve to be its own lifecycle phase as it could just as easily be an area of responsibility that is parallel to the entire key management lifecycle. There are three key aspects to monitoring that should be considered.  
1) It is important to monitor for unauthorized administrative access to crypto systems to ensure that unapproved key management operations are not performed. Any sort of unauthorized operation could have serious consequences for your system, and for your data.

2) Monitoring the performance of crypto systems is important. The performance of cryptographic calculations tends to be CPU-intensive, which means that your systems may be under significant load. Events such as flash popularity can result in a denial of service on its own, but when combined with an overloaded encryption service, the results could be far more serious, including data corruption or unavailability.

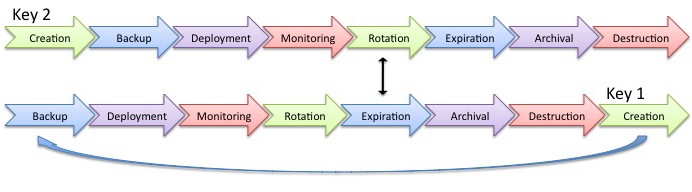
3) Monitoring the key in production is also important to ensure that the key has been created and deployed properly. If a corrupted key is deployed too quickly without proper vetting, the results could be catastrophic. Similarly, if a fault in the crypto system occurs, then the results could also interrupt service, with a negative impact to the business.

***Rotation***  
The concept of key rotation is related to key deployment, but they are not synonymous. In rotating keys, the goal is to bring a new encryption key into active use by the crypto system, and to also convert all stored, encrypted data to the new key. This process can be extremely time-consuming and CPU-intensive. However, assuming that all previous steps of the lifecycle have been followed, then this phase can be discretely focused on the conversion activity, and less on the activation of the new key for future encryption requests.

A word of caution: be mindful not to remove an old key from a production system until it can be proven that no data in production is still encrypted with the old key. Failing to perform due diligence, such as doing a manual query for the old key ID may result in data loss or a service outage.

Bearing this warning in mind, the diagram below demonstrates how the lifecycle of the first key will overlap the lifecycle of the new key. Note that Rotation and Expiration are essentially overlapping between the two keys. Also, it's worth noting that Rotation is maintained as a separate phase from Deployment because of its unique considerations, independent of installing a key into the production crypto system.

One additional thought to consider is that there may be good reasons for not performing a batch rollover of stored data from the old key to the new key. E.g., if the data is highly transient (accessed, written, or rewritten with very high frequency), then it may be adequate to instead flag a condition within the calling application such that the data will be automatically converted to the new key as part of the read/write activity. Using such a mechanism can reduce some of the load associated with key rotation.



***Expiration***  
The chosen strength of an encryption key will primarily take into consideration the length of time for which the data may be valid. In addition to key strength, it has also become best practice (and dictated by regulations like PCI DSS) that keys be expired and replaced on a timeframe shorter than the calculated life span of the key. The minimum time span that is advocated these days is one year for each key, and preferably shorter for keys protecting data of greater sensitivity.

The Expiration phase of key management represents the beginning of the deprecation period for the key. Key rotation should be completed prior to expiration, with all data encrypted with that key converted to the new key. The objective is to have the key replaced within the production system (but not removed) before it expires. In a sense, expiration represents a gating factor for planning, as much as it is a discrete phase in the lifecycle.

***Archival***  
The absolute last thing that you want to do when managing crypto systems is to destroy a key that still has data associated with it. Therefore, the second-to-last phase in the lifecycle is included, with a potentially open-ended mandate to not proceed to the final phase.  
Archival of expired, decommissioned keys should be based on a determination of whether or not data still exists somewhere in the data ecosystem that may be encrypted with the archived key. The data ecosystem extends beyond “live” data in production to backups that may exist in disaster recovery sites, as well as to all offline backups. If there is a requirement for data to be recoverable, then it is imperative that the keys be archived in parallel to that data.

Here are a couple tips to keep in mind when archiving a key:  
1) Document and index the key and its associated data in such a way that, should you need to recover the data with an archived key, you can do so in as effective and efficient a manner as possible. Documentation of the key should also include the time period during which the key was in use to help narrow down the search for an appropriate key in a recovery scenario.  
2) Ensure that the archived copy of the key has itself been secured. As was recommended in the Creation and Deployment phases, it may be useful to encrypt the symmetric key with the public half of an asymmetric key pair.  
3) Include recovery of encrypted data using archived keys as part of your routing BCP/DR testing procedures.

It is worth noting that some cryptographic appliances automatically archive expired keys in a secure fashion and may not ever move to the final phase, Destruction. Overall, this is a business risk decision that should be considered on a case-by-case basis. It may be rightfully concluded that encryptions keys will never advance from Archival to Destruction because the risk of such a change, and the associated permanence of the loss, may outweigh the risk of exposing the archived key.

***Destruction***  
The life of a key will end when it is destroyed. Key destruction should follow secure deletion procedures so as to ensure that it is properly obliterated. Be warned: key destruction should not be taken lightly, and should only occur after an adequately long Archival phase, and after at least two reviews have been completed to ensure that loss of the key will not correspond to loss of data.

***Links of Interest***  
If you found this post interesting, then you may also find the following links to be of interest.

The Future of Encryption  
[http://www.net-security.org/article.php?id=1113>/a>](http://www.net-security.org/article.php?id=1113)

[General, lightweight reference for encryption. Also see Schneier's seminal work, Applied Cryptography, and the other book, Practical Cryptography.](http://www.net-security.org/article.php?id=1113)[http://en.wikipedia.org/wiki/Encryption](https://en.wikipedia.org/wiki/Encryption)

Payment Card Industry Data Security Standard v1.1  
<https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf>

The Risks of Key Recovery, Key Escrow, & Trusted Third Party Encryption (1998)  
[http://www.cdt.org/crypto/risks98/](https://www.cdt.org/crypto/risks98/)

Analysis: Enterprise Key Management (MAY 1, 2007)  
<http://www.darkreading.com/document.asp?doc_id=126002>

Encryption key management worries loom (November 26, 2007)  
[http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9048838](https://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9048838)

Introduction To Database Encryption  
[http://securosis.com/2008/02/12/introduction-to-database-encryption/](https://securosis.com/2008/02/12/introduction-to-database-encryption/)

Cold Boot Attacks Against Disk Encryption  
[http://www.schneier.com/blog/archives/2008/02/cold\_boot\_attac.html](https://www.schneier.com/blog/archives/2008/02/cold_boot_attac.html)

CERIAS Weblogs » Confusion of Separation of Privilege and Least Privilege  
[http://www.cerias.purdue.edu/weblogs/pmeunier/infosec-education/post-139/confusion-of-separation-of-privilege-and-least-privilege/](https://www.cerias.purdue.edu/weblogs/pmeunier/infosec-education/post-139/confusion-of-separation-of-privilege-and-least-privilege/)

FIPS References  
<http://csrc.nist.gov/groups/STM/index.html>  
<http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>  
[http://en.wikipedia.org/wiki/FIPS\_140](https://en.wikipedia.org/wiki/FIPS_140)  
[http://en.wikipedia.org/wiki/FIPS\_140-2](https://en.wikipedia.org/wiki/FIPS_140-2)  
[http://en.wikipedia.org/wiki/Tamper\_resistance](https://en.wikipedia.org/wiki/Tamper_resistance)

**Tags:**

* [encryption](http://www.secureconsulting.net/MT/mt-search.cgi?blog_id=12&tag=encryption&limit=20)
* [infosec](http://www.secureconsulting.net/MT/mt-search.cgi?blog_id=12&tag=infosec&limit=20)
* [key](http://www.secureconsulting.net/MT/mt-search.cgi?blog_id=12&tag=key&limit=20)
* [key management](http://www.secureconsulting.net/MT/mt-search.cgi?blog_id=12&tag=key%20management&limit=20)
* [management](http://www.secureconsulting.net/MT/mt-search.cgi?blog_id=12&tag=management&limit=20)

Posted by [Ben Tomhave](http://www.secureconsulting.net/) on March 21, 2008 7:19 PM | [Permalink](http://www.secureconsulting.net/2008/03/the_key_management_lifecycle_1.html)