Understanding and Selecting a Key Management Solution

Version 1.0
Released: February 5, 2013

Securosis, L.L.C. http://securosis.com

Author's Note

The content in this report was developed independently of any sponsors. It is based on material originally posted on the Securosis blog but has been enhanced and professionally edited.

Special thanks to Chris Pepper for editing and content support.

Thanks to Carmen Zagazeta for design support. www.carmenzag.com

Licensed by Thales e-Security

Thales e-Security is a leading global provider of data protection solutions – delivering high assurance data encryption and key management solutions to the financial services, manufacturing, government, retail, healthcare, and technology sectors. The company has a 40-year track record of protecting sensitive corporate and government information across a wide range of technology areas including PKI, credential management, payment processing, network encryption, and many more. Thales e-Security solutions reduce the cost and complexity associated with the use of cryptography in today's traditional, virtualized, and cloud-based infrastructures, helping organizations reduce risk, demonstrate compliance, enhance agility, and pursue strategic goals with greater confidence. The company is represented in over 90 countries around the world. For more information, visit www.thales-esecurity.com.

Copyright

This report is licensed under the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 license.

http://creativecommons.org/licenses/by-nc-nd/3.0/us/

Table of Contents

Introduction 2

Why use a key manager? 3

How a key manager works 3

Key Manager Technical Features 5

Deployment options 5

Client access options 6

Key Generation, Encryption, and Cryptographic Functions 7

Physical Security and Hardening 8

Encryption Standards and Platform Support 8

System Maintenance and Deployment Features 9

Tokenization 9

The role of HSMs in key management 10

Management Features 11

Role Management and Separation of Duties 11

Key Grouping and Segregation 12

Auditing and Reporting 12

User Interface 13

Selection Process 14

Determine current project requirements 15

Write your draft RFP 16

Testing 16

Conclusion 17

Who We Are 18

About the Author 18

About Securosis 18

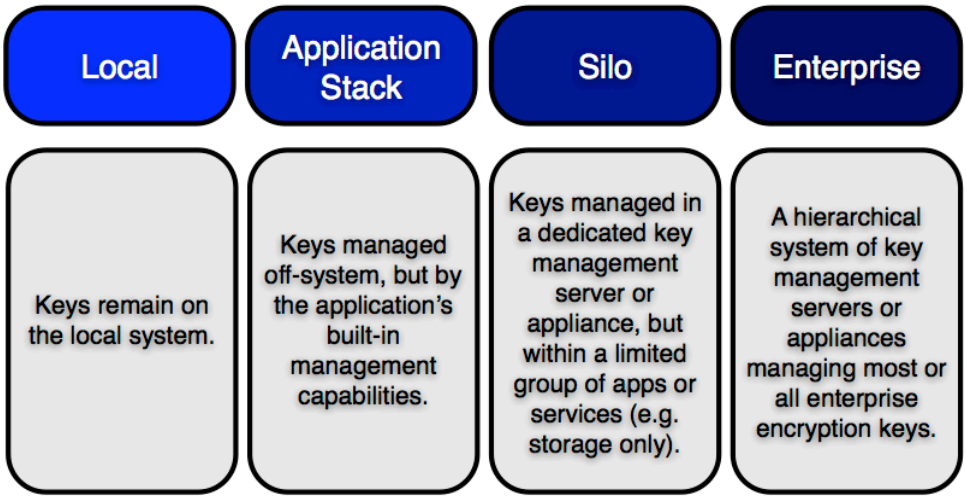
Introduction

Between new initiatives such as cloud computing, and new mandates driven by the continuous onslaught of compliance, managing encryption keys is evolving from something only big banks worry about into something which pops up at organizations of all sizes and shapes. Whether it is to protect customer data in a new web application, or to ensure that a lost backup tape doesn't force you to file a breach report, more and more organizations are encrypting more data in more places than ever before. And behind all of this is the ever-present shadow of managing all those keys.

In our Pragmatic Key Management for Data Encryption paper we highlighted some of the sins of the past that made key management painful, while also showing how new strategies and tools can cut through those roadblocks to make key management a much more (for lack of a better word) manageable process. In that paper we identified four strategies for data encryption key management:

- Manage keys locally.
- Manage keys within a single application stack with a built-in key management feature.
- Manage keys for a silo using an external key management service/server/appliance, separate from data and application stacks.
- Coordinate management of most or all keys across an enterprise with a centralized key management tool.

Key Management Strategies



We call these local, application stack, silo, and enterprise key management. The last two strategies introduce a dedicated tool for key management. This paper will dig in to explain the major features and functions of a key manager, what to look for, and how to pick one that suits your needs.

Why use a key manager?

Data encryption can be a tricky problem, especially at scale. Actually *all* cryptographic operations can be tricky; but we will limit ourselves to encrypting data rather than digital signing, certificate management, or other uses of cryptography. The more diverse your keys, the better your security and granularity, but the greater the complexity. While rudimentary key management is built into a variety of products – including full disk encryption, backup tools, and databases – at some point many security professionals find they need a little more power than what's embedded in the application stack. Drivers include:

- More robust reporting (especially for compliance).
- Better administrator monitoring and logging.
- Flexible options for key rotation and expiration.
- Management of keys across application components.
- Stronger security.

And sometimes, as with custom applications, there isn't any existing key management to lean on. In these cases it makes sense to start looking at a dedicated key manager. In terms of use cases, sweet spots we have found include:

- Backup encryption, due to a mix of longevity needs and very limited key management implementations in backup products themselves.
- Database encryption, because most database management systems only include the most rudimentary key management, rarely with the ability to centrally manage keys across different database instances or segregate keys from database administrators.
- Application encryption, which nearly always relies on a custom encryption implementation and, for security reasons, should separate key management from the application itself.
- Cloud encryption, due to the high volume of keys and variety of deployment scenarios.

This is just to provide some context – many of you reading this will already know you need a dedicated key manager. If you want more background on data encryption key management, and when to step up to a key manager, you should read our other paper first, then return to this one. The rest of this paper will cover technical features, management features, and how to choose between products.

How a key manager works

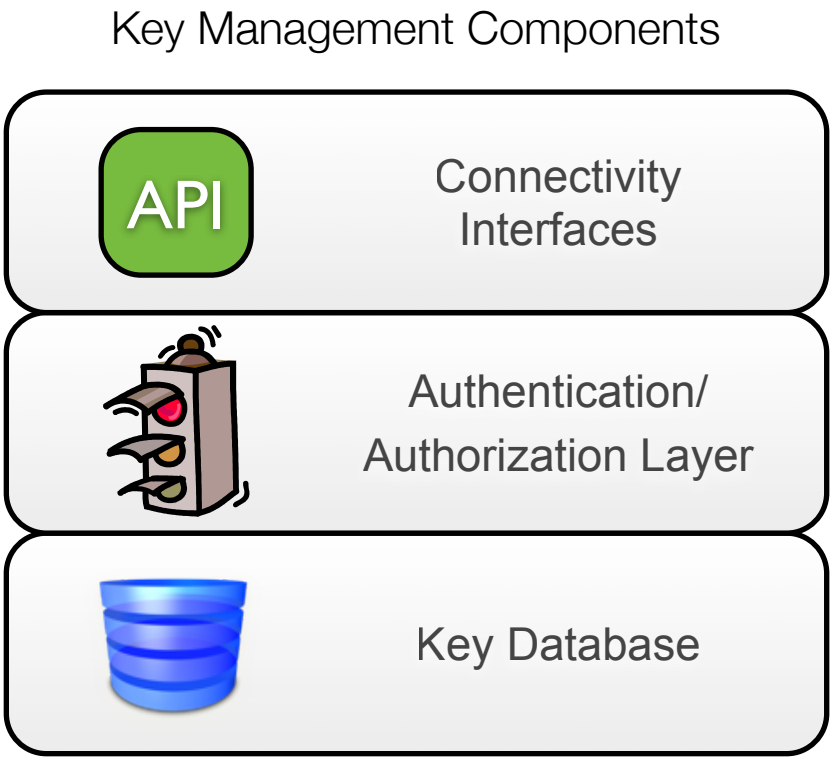
The core function of a key manager is fairly straightforward: it is essentially a database that tracks encryption keys and provides those keys for authorized requests.

At minimum this requires three features:

- A secure database for managing keys.
- An Authentication, Authorization, and Access control (AAA) model for restricting and managing access to keys.
- Network-based access methods to provide the keys to requesting systems and services.

It looks simple but the details can become quite complex. The key manager needs to scale to massive numbers of keys, tracking all the metadata and rights associated with each key. The tool also must support a variety of access methods — which may include client software, APIs, and network protocol. Security needs vary per key, which means the authorization and access control model needs to scale with the keys while still protecting all keys in the event of attack.

These complexities differentiate between products and will guide your eventual selection. They are also the reason you can't simply stand up your own database and call it a key manager — trust us, we have seen people try.



Key Manager Technical Features

Due to the different use cases for encryption tools, key management solutions have likewise developed along varied paths, reflecting their respective origins. Many evolved from Hardware Security Managers (HSMs), some were built from the ground up, and others are offshoots from key managers originally developed for a single purpose, such as full disk or email encryption.

Most key managers include a common set of base features but there are broad differences in implementation, support for different deployment scenarios, and additional features. We will focus on technical features, follow up with management features (including user interface), and conclude with product selection.

Deployment options

There are three deployment options for enterprise key managers:

- Hardware Appliance
- Software
- Virtual Appliance

Let's spend a moment on the differences between these approaches.

Hardware Appliance

The first key managers were almost all appliances – most frequently offshoots of Hardware Security Modules (HSMs). HSMs are dedicated hardware tools for the management and implementation of multiple cryptographic operations, and are in wide use (especially in financial services), so key management was a natural evolution. Hardware appliances have two main advantages:

- Specialized processors improve security and speed up cryptographic operations.
- Physical hardening provides tamper resistance.

Some non-HSM-based key managers also started as hardware appliances, often due to customer demand for physical hardening.

These advantages are still important in many use cases, but over the past five to ten years the market segment of users *without* hardening requirements has expanded and matured. Key management itself doesn't necessarily require

encryption acceleration or hardware chains of trust. Physical hardening is still valuable, but not mandatory in many use cases.

Software

Enterprise key managers can also be deployed as software applications on your own hardware. This provides more flexibility in deployment options if you don't need additional physical security or encryption acceleration. Running the software on commodity hardware may also be cheaper.

Aside from cost savings, key management deployed as software can offer more flexibility – such as multiple back-end database options, or the ability to upgrade components without having to replace the entire server.

Of course software running on commodity server hardware is less locked down than a secure hardware appliance, but – especially on a properly configured dedicated server – it is more than sufficiently secure for many use cases.

Virtual Appliance

A virtual appliance is a pre-built virtual machine, and offers some deployment advantages over both hardware appliances and software.

Virtual appliances come pre-configured, so there is no need to install software components yourself. Their bundled operating systems are generally tightly locked down and tuned to support the key manager. Deployment is similar to a hardware appliance – you don't need to build or secure a server yourself, but as a virtual machine you can deploy it as flexibly as software with a suitable virtualization infrastructure.

This is a great option for distributed or cloud environments with adequate virtual infrastructure.

We will return to this choice in the selection process.

Client access options

Whatever deployment model you choose, you need some way of getting the keys where they need to be, when they need to be there, for cryptographic operations. Remember, in this report we are always talking about using an external key manager, which means a key exchange is always required.

Clients (whatever needs the key) usually need support for the following core functions for a complete key management lifecycle:

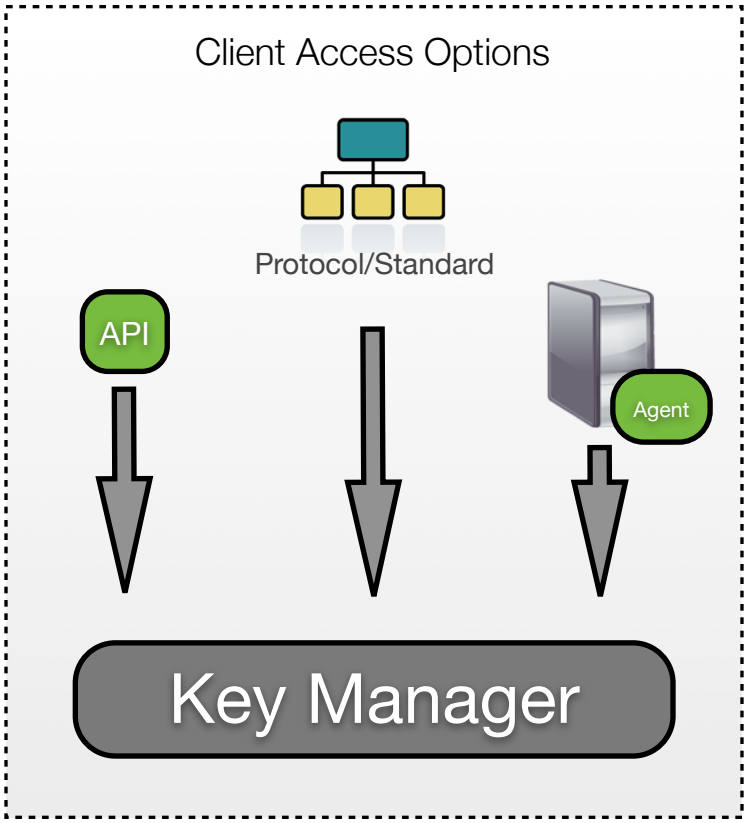
- Key generation
- Key exchange (gaining access to the key)
- Additional key lifecycle functions, such as expiring or rotating a key

Depending on what you are doing, you may allow or disallow these functions under different circumstances. For example you might allow key exchange for a particular application, but not allow that application any other management functions (such as key generation and rotation).

Access is managed in one of three ways, and many tools support more than one:

- **Software agent:** A dedicated agent handles the client's side of key functions. These are generally designed for specific use cases – such as supporting native full disk encryption, specific backup software, various database platforms, and so on. Some agents may also perform cryptographic functions such as wiping the key from memory after each use.

- **Application Programming Interfaces:** Many key managers are used to handle keys from custom applications. An API allows you to access key functions directly from application code. Keep in mind that APIs are not all equivalent – they vary widely in platform support, supported programming languages, simplicity or complexity of API calls, and functions accessible via the API.
- **Protocol & standards support:** Many key managers support a variety of proprietary and open protocols. Various encryption tools support their own protocols for key management, and (like a software agent) a key manager may include support – even for protocols from different vendors. Open protocols and standards are also emerging but not in wide use yet, and may be supported.



Key Generation, Encryption, and Cryptographic Functions

Due to their heritage, some key managers also offer cryptographic functions, such as:

- Key generation
- Encryption and decryption
- Key rotation
- Digital signing

Key generation and rotation options are fairly common because they are important parts of the key management lifecycle; encryption and decryption are less common.

If you are considering key managers that also perform cryptographic functions, you need to consider additional requirements, such as:

- How are keys generated and seeded?
- What kinds of keys and cryptographic functions are supported? (Take a look at the Standards section a bit later).
- Performance: How many cryptographic operations of different types can be performed per second?

But key generation isn't necessarily required – assuming you only plan to use the tool to manage existing keys, perhaps in combination with separate HSMs.

Physical Security and Hardening

Key managers deployed as hardware appliances tend to include extensive physical security, hardening, and tamper resistance. Many of them are designed to meet government and financial industry standards.

> "*The 140 series of Federal Information Processing Standards* ***(FIPS)*** *are U.S. government computer security standards that specify requirements for cryptography modules.* ***FIPS Level 1 does not require physical security****, and can apply to software.* ***Levels 2-4 must meet hardware requirements****.*"

These products come in sealed enclosures designed to detect attempts to open or modify them. They only include the external ports needed for core functions, without (for example) USB ports that could be used to insert malware.

Most include one or more smart card ports for physical keys to support certain administrative functions. For example, they can require two or three administrator keys to allow access to more sensitive parts of the system (yes, this means physically walking up to the key manager and inserting cards, even if administration is otherwise through a remote web interface).

Together these features help ensure the key manager isn't tampered with, and that its data is still secure even if the manager is physically stolen.

But physical hardening isn't always necessary – or we wouldn't have software and virtual machine options. Those alternatives are still highly secure — the choice comes down to which deployment scenarios you need to support. Software and virtual appliances also include extensive security features – just nothing tied to the enclosure or specialized hardware.

Anyone claiming physical security of an appliance should meet the FIPS 140–2 standard specified by the United States National Institute of Standards and Technology (NIST), or the regional equivalent. This includes requirements for both the software and hardware security of encryption tools.

Encryption Standards and Platform Support

As the saying goes, the wonderful thing about standards is there are so many to choose from. This is especially true in the world of encryption, which lives and breathes a wide array of standards.

An enterprise key manager needs to handle keys from every major encryption algorithm, plus all the communications and exchange standards (and proprietary methods) to actually manage keys outside the system or service where they are stored. As a database system, actually storing the keys for different encryption standards is easy. But supporting all the various ways of managing keys externally, for both open and proprietary products, is far more complex. When you add in requirements to generate, rotate, or change keys, life gets even harder.

Here are some possible standards and platform features:

- Support for storing keys for all major cryptographic standards.
- Support for key communications standards and platforms to exchange keys, which may include a mix of proprietary implementations (*e.g.,* a specific database platform) and open standards (*e.g.,* the evolving Key Management Interoperability Protocol [KMIP]).
- Support for generating keys for common cryptographic standards.
- Support for rotating keys in common applications.

It comes down to having a key manager that supports the kinds of keys you need, on the types of systems that use them.

System Maintenance and Deployment Features

As enterprise tools, key managers need to support a basic set of core maintenance features and configuration options:

Backup and Restore

Losing an encryption key is often worse than losing the data. When you lose the key, you effectively lose access to every version of the data that has ever been protected. And we try to avoid unencrypted copies of encrypted data, so you are likely to lose every version of the data, permanently.

Enterprise key managers need to handle backup and restoration in an extremely secure manner. Usually this means encrypting the entire key database (including all access control & authorization rules) before backup. Additionally, backups are generally encrypted with multiple or split keys, which require more than one administrator to access or restore. Various products use different strategies to manage secure incremental backups so you can back up the system regularly without destroying system performance.

High Availability and Load Balancing

Some key managers are only deployed in a limited fashion, but in general these tools need to be available all the time, every time, sometimes to large volumes of traffic.

Enterprise key managers should support both high availability and load balancing options to ensure they can meet demand.

Another important high-availability option is key replication. This is the process of synchronizing keys across multiple key managers, sometimes between geographically separated data centers. Replication is always tricky and needs to scale effectively to avoid either loss of a key or conflicts in case of a problem during rekeying or new key issuance.

Hierarchical Deployments

There are many situations in which you might use multiple key managers to handle keys for different application stacks or business-unit silos. Hierarchical deployment support enables you to create a “manager of managers” to enforce consistent policies across individual-system boundaries and throughout distributed environments.

For example you might use multiple key managers in multiple data centers to generate new keys, but have those managers report back to a central master manager for auditing and reporting.

Tokenization

Tokenization is an encryption alternative that replaces a sensitive value with a random one, known as a token. The tokenization tool tracks which tokens correspond to which sensitive values and, when the right conditions are met, can use that information to recover the protected data.

The key difference between tokenization and encryption is that an encrypted value can always be decrypted with the key, whereas a token can only be converted to the original data by the tokenization tool itself. Additionally, tokens can be created to match the format of the original data, making them easier to substitute in legacy systems with format constraints.

We cover tokenization products in depth in Understanding and Selecting a Tokenization Solution. Some key managers also include tokenization features, while some tokenization products include key management. If you are interested in tokenization we recommend that paper.

The role of HSMs in key management

All Hardware Security Modules – HSMs for short – supports a wide range of cryptographic functions such as key generation, signing operations, certificate management, key storage, and root certificate security. HSMs are comprised of specialized hardware that provides fast and efficient cryptographic operations and protection from physical attacks. The hardware is deployed as a network appliance, or a special card within a server, to offload these computationally expensive operations.

As we mentioned, many key managers started life as HSMs, and many current HSMs support most or all necessary key management functions. All HSMs include some level of key management, as well as additional features, but not all HSMs are intended for use as enterprise key managers – some are only designed to manage their own keys. Some HSMs don't store keys at all – they create and use keys stored elsewhere to perform encryption services.

An enterprise key manager is focused primarily on managing encryption keys. Additional cryptographic functions aren't required for these products.

There is considerable overlap between different products in this space, and many different options depending on what you need. We are focusing on key management features and functions, but they may show up in a single-purpose enterprise key management tool, an HSM, and other encryption tools with expanded key management support.

Management Features

It is one thing to collect, secure, and track a wide range of keys; but doing so in a useful and manageable manner demonstrates the differences between key management products.

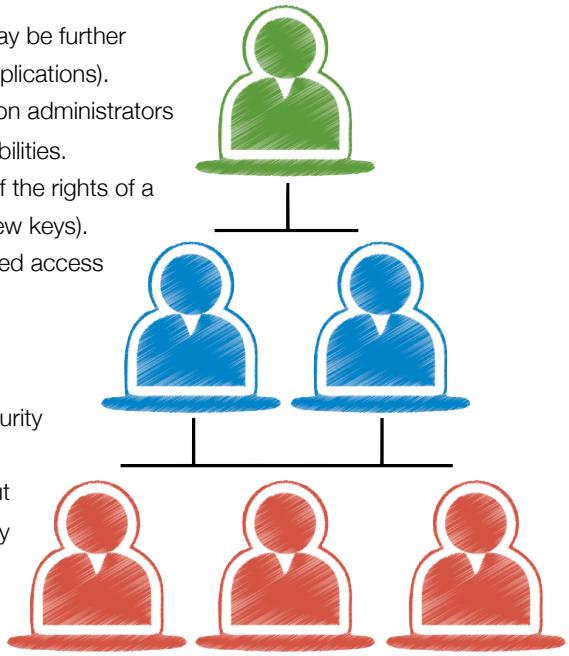
Managing disparate keys from distributed applications and systems — for multiple business units, technical silos, and IT management teams — is more than a little complicated. It involves careful segregation and management of keys; multiple administrative roles; abilities to organize and group keys, users, systems, & administrators; appropriate reporting; and an effective user interface to tie it all together.

Role Management and Separation of Duties

If you are managing more than a single set of keys for a single application or system you need a robust role-based access control system (RBAC) – not only for client access, but for the administrators managing the system. It must support ironclad separation of duties with multiple levels of access and administration.

An enterprise key manager should support multiple roles, especially multiple administrative roles. Regular users never directly access the key manager, but system and application administrators, auditors, and security personnel may all need some level of access at different points in the key management lifecycle. For instance:

- A systems-admin role for administration of the key manager itself, with no access to the actual keys.
- Limited administrator roles that allow access to subsets of administrative functions such as backup and restore, creating new key groups, and so on.
- An audit and reporting role for viewing reports and audit logs. This may be further partitioned to allow access only to certain audit logs (*e.g.,* specific applications).
- System/application manager roles for individual system and application administrators who need to generate and manage keys for their respective responsibilities.
- Sub-application manager roles which only have access to a subset of the rights of a system or application manager (*e.g.,* create new keys only but not view keys).
- System/application roles for the actual technical components that need access to keys.



Any of these roles may need access to a subset of functionality, and be restricted to key groups or individual keys. For example a database security administrator for a particular system gains full access to create and manage keys only for the databases associated with those systems, but not to manage audit logs, and no ability to create or access keys for any other applications or systems.

Ideally you can build an entitlement matrix where you select a

particular role and then assign it to one or mor users and groups of keys. You might assign the “application manager” role to “user bob” for group “CRM keys”.

Split administrative rights

There almost always comes a time where administrators need deeper access to perform highly-sensitive functions or even directly access keys. Restoring from backup, replication, rotating keys, revoking keys, and accessing keys directly, are all functions with major security implications and which you may not want to trust to a single administrator.

Most key managers allow you to require multiple administrators to approve such functions, to limit the ability of any one administrator to compromise security. This is especially important when working with the master keys for the key manager, which are needed for tasks such as replication and restoration from backup.

Such functions involving the master keys are often handled through a split key. Key splitting provides each administrator with a portion of a key, some or all of which are required to complete operations. This is often called “*m* of *n*” because you need *m* sub-keys from a total of *n* in existence to perform an operation (*e.g.,* 3 of 5 administrative keys).

These keys or certificates can be stored on a smart card or similar security device for better security.

Key Grouping and Segregation

Role management covers users and their access to the system, while key groups and segregation manage the objects (keys) themselves. No one assigns roles to individual keys – you assign keys to groups, and then parcel out rights from there as described above.

Assigning keys and collections of keys to groups allows you to group keys not only by system or application (such as a single database server), but for entire collections or even business units (such as all the databases in accounting). These groups are then segregated from each other, and rights are assigned per group. Ideally groups are hierarchical so you can group all application keys, then subset application keys by application group, and then by individual application.

Auditing and Reporting

In our compliance-driven security society, it isn’t enough to merely audit activity. You need fine-grained auditing with customized reports for different compliance and security needs.

Type of activity to audit include:

- All access to keys
- All administrative functions of the key manager
- All key operations – including generating and rotating keys

A key manager is about as security-sensitive as it gets, so everything that happens to it should be auditable. That doesn’t mean you will want to track every time a key is sent to an authorized application, but you should have the ability when you need it.

Reporting

Raw audit logs aren’t overly useful on a day to day basis, but a good reporting infrastructure helps keep the auditors off your back while highlighting potential security issues.

Key managers may include a variety of pre-configured reports and support creation of custom reports. For example you might generate a report of all administrator access (not application access) to a particular key group, or one covering all administrative activity in the system.

Reports may run on a preset schedule, emailing regular activity summaries to appropriate stakeholders.

User Interface

In the early days of key management everything was handled using command line interfaces. Most current systems implement graphical user interfaces (often browser-based) to improve usability. There are massive differences in look and feel across products, and a GUI that fits the workflow of your staff can save a great deal of time and expense.

Common user interface components include:

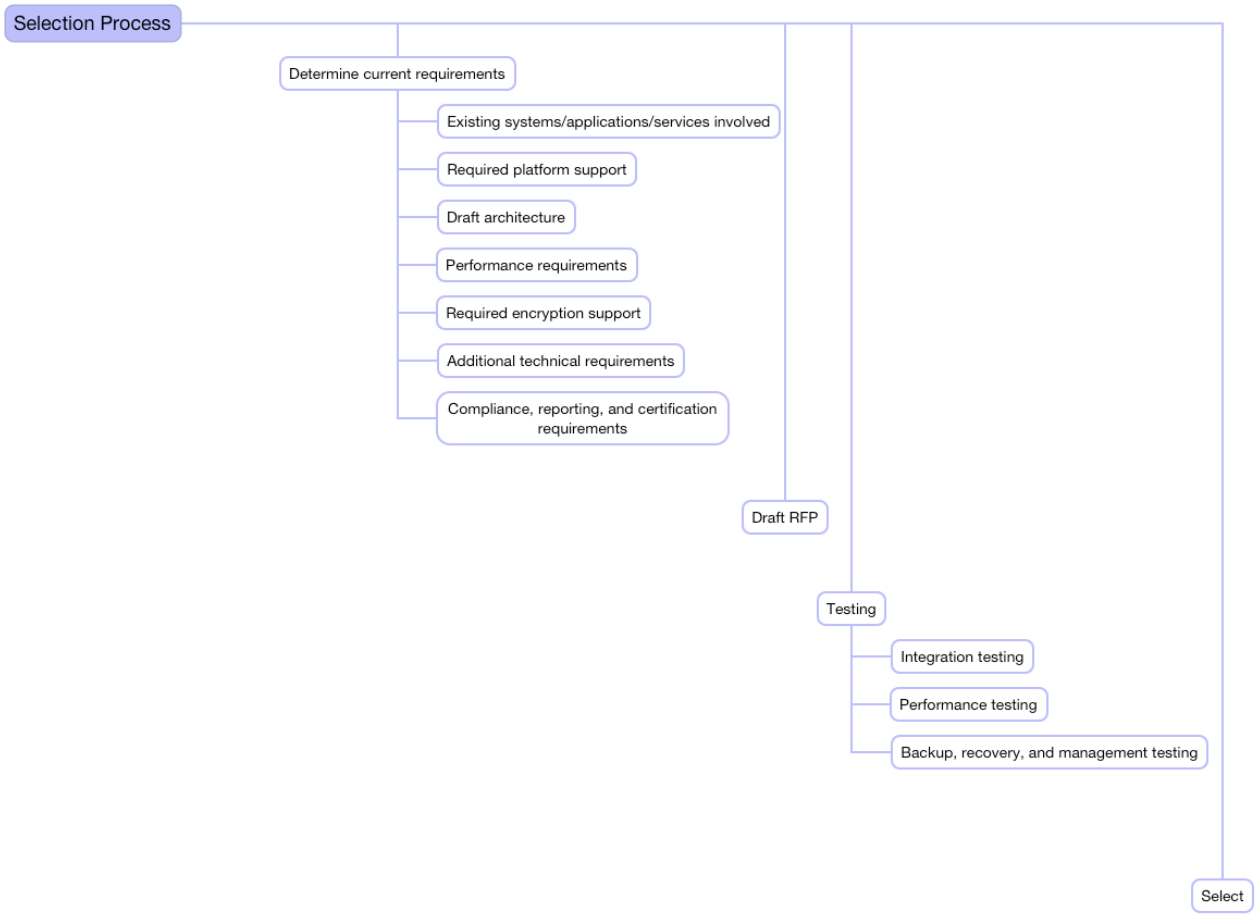
- A system administration section for managing the key manager configuration, backup/restore, and other system settings.
- A user management section for defining users, roles, and entitlements.
- A key management section for creating key collections and groups, and defining access to objects.
- A system/application manager section to allow system and application managers access to the key manager functions under their control.
- A section for managing which systems and applications access keys, and how they are configured and authenticated.
- Filtering of user interface elements so people only see what they have access to.

Depending on how you use the key manager you might never allow access to anyone other than security administrators – with everyone else submitting requests or managing keys through an API, client, or command line interface. But if other people need access you will want to ensure the user interface supports whatever limited access you want to grant with an appropriate interface.

Selection Process

Now that you have a better understanding of major key manager features and options we can spend some time outlining the selection process. This largely comes down to understanding your current technical and business requirements (including any pesky compliance requirements), and trying to plan ahead for future needs.

Yes, this is all incredibly obvious, but with so many different options out there – from HSMs with key management to cloud key management services – it is too easy to get distracted by the bells and whistles and miss important requirements.



Determine current project requirements

Nobody buys a key manager for fun. It isn't like you find yourself with some spare budget and go, "Hey, I think I want a key manager!" There is always a current project driving requirements, so that is the best place to start.

We will talk about potential future requirements, but never let them interfere with your immediate needs. We have seen projects get diverted or derailed as the selection team tries to plan for every contingency, then buys a product that barely meets current needs, which quickly causes major technical frustration.

1. **List existing systems, applications, and services involved:** The best place to start is a list of the different systems, application stacks, and services that need key management. This could be as simple as "Oracle databases" or as complex as a list of different databases, programming languages used in applications, directory servers, backup systems, and off-the-shelf application stacks.

2. **Determine required platform support:** With your list of systems, applications, and services in hand, determine exactly what platform support you need. This includes which programming languages need API support, any packaged applications requiring support (including specific versions), and perhaps even operating systems and hardware. This should include encryption algorithms. Be as specific as possible because you will send this list to prospective vendors to ensure their product can support your platforms.

3. **Map out draft architectures:** In addition to knowing which technical platforms to support, you also need an idea of how you intend to connect them to the key manager. There is a big difference between connecting a single database to a key manager inside a same data center and supporting a few hundred cloud instances connected back to an on-premise key manager in a hybrid cloud scenario. If you plan to use one or more key managers in an enterprise deployment, with multiple different platforms, across different locations, you need to be sure you can get all the appropriate bits connected, and you might need features such as multiple network interfaces.

4. **Calculate performance requirements:** It can be difficult to accurately calculate performance needs before you start testing, but do your best. The two essential estimates are the number of key operations per second and your network requirements (both speed and number of concurrent network connections/sockets). If you plan to use a key manager that also performs cryptographic operations, include those performance requirements.

5. **List any additional encryption support required:** We have focused almost exclusively on key management, but as we mentioned some key managers also support a variety of other crypto operations. If you plan to use the tool for encryption, decryption, signing, certificate management, or other functions, make a list and include any detailed requirements such as the ones above (platform support, performance, etc.).

6. **Determine compliance, administration, reporting, and certification requirements**: This is the laundry list of compliance requirements (such as which operations require auditing), administrative and systems management features (backup, administrator separation of duties, etc.), reporting needs (such as pre-built PCI reports), and certifications (nearly always FIPS-140). We have detailed these throughout this paper, so you can use it as a guide when you build your checklist.

7. **List additional technical requirements**: By now you will have a list of your core requirements but there are likely additional technical features on your required or optional list.

And last but not least spend some time planning for the future. Check with other business units to see if they have key management needs or are already using a key manager. Talk to developers to see if they might need encryption and key

management for upcoming projects. Review any existing key management, especially in application stacks, that might benefit from a more robust solution.

You don't want to list every potential scenario to the point where no product can possibly meet all your requirements, but it is worth taking a step back before you put together an RFP to see if you can take a more strategic approach.

Write your draft RFP

Try to align your RFP with the requirements collected above. There is a common tendency to ask for every feature you have ever seen in product demos, but this frequently results in bad RFP responses which make it even harder to match vendor responses against your priorities. (That is a polite way of saying that an expansive cookie-cutter RFP request is likely to result in a cookie-cutter response, rather than one tailored to your actual needs).

Enough of the soapbox – let's get into testing.

Testing

Integrating a key manager into your operations will either be incredibly easy or incredibly tricky, depending on everything from network topography to applications involved to overall architecture. For any project of real scale it is absolutely essential to test compatibility and performance before you buy.

Integration Testing

The single most crucial piece to test is whether the key manager will actually work with whatever application or systems you need to connect with.

Ideally you will test this in your own environment before you buy, but that isn't always possible. Not all vendors can provide test systems or licenses to all potential customers, and in many situations you will need support services or even custom programming to integrate the key manager. Here are some suggestions and expectations for how to test and minimize your risk:

- If you are deploying the tool to manage the keys of a common application or platform that the vendor commonly supports, such as a major backup system, odds are it will work in your environment. Ask the vendor how many production deployments they know of with your version, and for a reference, and you might not need to test yourself.
- References are also your friends for less-direct use cases. Ideally reach out to your personal network for someone doing something similar instead of relying on vendor references, but they can also do in a pinch. Focus your questions on the complexity of integration, hidden obstacles, and amount of effort involved in the project. Ideally talk to someone at the engineering level, not the CISO who approved the budget and didn't actually manage the implementation.
- Be wary of selecting a product you can't test if you cannot find more than a couple references from people using it in circumstances like yours.
- The vendor may be able to provide a virtual machine or software for testing, even if you plan on using hardware. This is generally fine for integration testing.
- To test, perform a basic integration on a non-production system and practice the workflow. Depending on the nature of the project you should test some or all of key management, exchange, generation, rotation, and any other features you expect to use, including any sub-admin UI.
- For larger and more complex projects, it may make sense to buy a single license for more rigorous testing and a full proof of concept. You may even want to purchase low-end licenses for multiple products to compare them. Clearly this is only an option for larger organizations and projects.

For basic projects integration testing doesn't need to be overly complex. You simply need to make sure the tool will handle keys, for the operations you need, on the applications and systems within your project scope. Larger and more complex projects need much deeper testing.

Stress/Performance Testing

Don't ever trust a spec sheet. Key manager performance varies based on the kinds of operations you perform, any cryptographic operations (encryption, key generation, etc.), and network load. As before, direct testing isn't always possible or necessary, but any testing is best focused on the following areas:

- The number of key exchanges, of the type you require, per second. This isn't necessary for deployments where you aren't performing that many operations (including some backup scenarios).
- The number of key generation operations per second.
- The number of any other required crypto operations (beyond key management) per second, if those are within scope for your project (*e.g.,* encryption).
- Network bandwidth.
- Number of concurrent network sockets supported (this may be the limiting factor rather than bandwidth).

Make sure you perform this testing on the same platform you plan to use for deployment, as operations may not scale linearly. For example, if you are deploying software or a virtual appliance, put it on the same hardware you expect to use later. If you are deploying in the cloud make sure you know what physical server you are on (for private clouds – this doesn't apply to public), and what instance flavor (size) you are using.

Backup, Recovery, and Management Testing

Perform a key backup and restore, making sure you understand the workflow. If, for example, you plan to require *m* of *n* administrative keys for recovery, test that. Also walk through the management interface, including audit and reporting, to make sure it fits your needs and expected workflow.

Conclusion

Hopefully this has helped you gain a better understanding of the role of enterprise key managers, the major features to look for, and how to evaluate and select one (or more).

Increasing privacy concerns, regulations, use of distributed and cloud computing, and even BYOD, are all driving increased use of encryption in more diverse organizations than we have seen before. Sometimes the encryption implementations do a fine job of handling encryption keys themselves, but we see plenty of use cases and specific implementations where it makes far more sense to shift to an external key manager.

Don't forget to read our Pragmatic Key Management for Data Encryption paper for an overview of key management strategies and more use cases, then use this paper for guidance if you decide to use an external key manager.

Who We Are

About the Author

Rich Mogull, Analyst and CEO

Rich has twenty years of experience in information security, physical security, and risk management. He specializes in data security, application security, emerging security technologies, and security management. Prior to founding Securosis, Rich was a Research Vice President at Gartner on the security team where he also served as research co-chair for the Gartner Security Summit. Prior to his seven years at Gartner, Rich worked as an independent consultant, web application developer, software development manager at the University of Colorado, and systems and network administrator. Rich is the Security Editor of TidBITS, a monthly columnist for Dark Reading, and a frequent contributor to publications ranging from Information Security Magazine to Macworld. He is a frequent industry speaker at events including the RSA Security Conference and DefCon, and has spoken on every continent except Antarctica (where he's happy to speak for free — assuming travel is covered).

About Securosis

Securosis, L.L.C. is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services.

We provide services in four main areas:

- Publishing and speaking: Including independent objective white papers, webcasts, and in-person presentations.
- Strategic consulting for end users: Including product selection assistance, technology and architecture strategy, education, security management evaluations, and risk assessments.
- Strategic consulting for vendors: Including market and product analysis and strategy, technology guidance, product evaluations, and merger and acquisition assessments.
- Investor consulting: Technical due diligence including product and market evaluations, available in conjunction with deep product assessments with our research partners.

Our clients range from stealth startups to some of the best known technology vendors and end users. Clients include large financial institutions, institutional investors, mid-sized enterprises, and major security vendors.

Securosis has partnered with security testing labs to provide unique product evaluations that combine in-depth technical analysis with high-level product, architecture, and market analysis.