

Analysis of the PCI-DSS 1.2 Encryption and Key Management Requirements

*Version: 1.0
October 18, 2008*

Copyrights & Notices

Copyright © 2001-2008, StrongAuth Inc. All rights reserved.

This document has been provided by StrongAuth, Inc. (StrongAuth) for the purpose of informing users of StrongAuth's products and services. With the exception of referenced material, StrongAuth reserves all rights, without waiver, election or other limitation to the full extent permitted by law, in and to this material and the information contained herein.

While this document may be freely distributed, it must be redistributed without any modifications. For all inquiries please contact info@strongauth.com.

THIS DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

1. Introduction

The Payment Card Industry (PCI) released version 1.2 of its Data Security Standard (DSS) on October 01, 2008 (https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml).

The PCI-DSS covers security requirements of many information technology components that companies processing and/or storing credit card numbers must comply with. However, this paper focuses on analyzing Sections 3.4 through 3.6 (Protect Cardholder Data) of the DSS, to provide an analysis on these requirements, from StrongAuth, Inc.'s (StrongAuth) perspective.

While StrongAuth is not a Qualified Security Assessor (QSA) and does not speak for the PCI Security Standards Council, given our decade-long experience in setting up key-management infrastructures for some of the largest companies in the world, it is hoped that this perspective will provide greater insight on how cryptographic technology can be used to protect sensitive customer data, while reducing risks in the key-management infrastructure.

2. Analysis

Data encryption and key-management remain a highly-specialized field of study in the field of information security, which in itself is a specialized field within information technology. While it is possible for any information technologist - with a little effort - to understand cryptographic concepts, learn an Application Programming Interface (API) and write encryption code, a thorough understanding of the vulnerabilities in encryption and key-management requires more than a casual review of relevant text on the subject.

Secondly, because every IT project deals with systems, networks, databases, web-servers, etc., most information technologists have a deep understanding of these components and how they work with each other. However, encryption and key-management is not a typical requirement of the vast majority of IT projects, and as a result, information technologists have not built up a deep understanding of this area. This can lead to weak designs, insecure procedures and poor implementations that leaves their data vulnerable to attacks. What is worse is that it provides the company with a false sense of security because of the cryptography involved.

The PCI-DSS provides very high-level guidelines with respect to encryption and key-management. Within those guidelines, it is possible to implement a range of designs that can be deemed secure or weak. While opinions will always vary on the topic of security, most information technologists and business people will not want to revisit their encryption designs and solutions after implementing it, if they can avoid it. However, the goal of this analysis is to not create a “military-strength” encryption and key-management infrastructure, but to create a secure environment that will not require reinvesting in this technology for many years to come. Some adjustments will always be necessary to accommodate for new information in this field, but that is all it should remain - adjustments, and not revamping.

The format of this paper presents the PCI-DSS requirement first, followed by our interpretation of the requirement and how the open-source StrongKey™ software (and the StrongAuth EKMI Appliance™, which includes StrongKey) addresses the requirement. An assumption made in this analysis is that the company in question needs to store the Primary Account Number (PAN); if there is no need to store the PAN, then there may be no need for StrongKey in your IT environment.

2.1. PCI-DSS Requirement 3.4

Render PAN, at minimum, unreadable anywhere it is stored (including on portable digital media, backup media, in logs) by using any of the following approaches:

- *One-way hashes based on strong cryptography*
- *Truncation*
- *Index tokens and pads (pads must be securely stored)*
- *Strong cryptography with associated key-management processes and procedures*

The MINIMUM account information that must be rendered unreadable is the PAN.

StrongAuth Analysis: If a business has no need to “view” the PAN as unencrypted data (also known as “plaintext”) after the initial transaction, and only needs to compare a PAN submitted by an application with a stored PAN (as when processing a refund transaction), a one-way cryptographic transformation of the PAN into a message digest (hash) is an optimal method of securing the sensitive information. The creation of message-digests depends only on a cryptographic algorithm

and has no need for cryptographic keys. Consequently, once PANs are converted to message-digests the vulnerability to the exposure of sensitive data is eliminated.

Cryptographic hashes are a secure means of storing “transformed” PANs, since the business application can always compare a PAN presented to the application with a stored PAN without having to decrypt the stored PAN. If businesses can perform their business function with just message digests, they should attempt to do so, since it reduces their exposure significantly. However, make sure that only approved digest algorithms are used to avoid surprises.

How StrongKey meets this requirement

StrongKey includes the ability to generate message-digests for any kind of data. It generates digests not only based on the National Institute of Standards and Technology (NIST) approved Secure Hashing Algorithm: SHA-1, but also on the new “Suite B” algorithms: SHA-256, SHA-384 and SHA-512. (StrongKey does not use the deprecated MD5 hashing algorithm at all).

However, if creating message-digests is the *only* cryptographic requirement in an application, StrongKey is not an optimal tool for the job. It is fairly simple for most programmers to generate message-digests within their applications and using StrongKey for merely generating message-digests would be excessive.

StrongKey is best suited for environments where there is a need to encrypt sensitive data – any data and not just PANs – for later decryption, and where cryptographic keys need to be managed independent of applications. This allows multiple applications to share encrypted data (also known as “ciphertext”) at will with each other – even over insecure networks – while each application can avail key-management services from StrongKey to decrypt the ciphertext independently.

2.2. PCI-DSS Requirement 3.4.1

If disk encryption is used (rather than file- or column-level database encryption), logical access must be managed independently of native operating system access control not be tied to user accounts.

StrongAuth Analysis: Disk encryption (encryption performed at the disk-drive layer either by an operating system driver supplied by the disk-drive manufacturer, or by the firmware on the disk-drive controller) can, potentially, mitigate risks associated with stolen or lost computing devices containing sensitive data. Without the pass-phrase to the disk-decryption tool, the thief may be unable to decrypt the contents of the hard disk. They may be able to take advantage of vulnerabilities in the implementation of the encryption system, but in theory, this is supposed to keep data secure.

*However, disk encryption **cannot** protect a company from on-line attacks on a system that is powered-up and operational.*

This is because, once a user has legitimately authenticated themselves to the machine's drive encryption software (even with a separate password or PIN), all data-blocks are automatically decrypted by the cryptographic software regardless of which application running under the authenticated user's ID, requests the data. Thus, if an attacker managed to execute malware on the machine, because the user has already authenticated himself/herself to the drive, the malware will be able to read data off the disks much as the legitimate user would.

Additionally, encrypting in the disk-drive – which is the lowest layer of an application's technology stack – leaves all layers above the disk-drive vulnerable to snooping by attackers. Given the complexity of today's applications, there are, potentially, numerous opportunities for attackers to snoop unencrypted data on a compromised machine.

While disk-drive encryption has some short-term risk-mitigation properties, StrongAuth believes that the strongest long-term data-protection comes from encrypting and decrypting data within the application-layer *by the application itself*. In this paradigm, data is protected no matter what the storage media, no matter which network the data traverses, and no matter how many software layers intervene between the application and storage media.

How StrongKey meets this requirement

The Symmetric Key Client Library (SKCL) in StrongKey can be used by disk-encryption driver's to acquire symmetric encryption keys from a centralized Symmetric Key Services (SKS) server, and the driver, in turn, can use these keys to encrypt data on storage-media. However, StrongKey itself does not provide any disk-drive encryption capability.

While the SKCL, besides sending and receiving messages to the SKS server and managing keys in the local cache **does** encrypt and decrypt data, this encryption and decryption capability is only supported outside the operating system kernel (where device drivers typically operate). The SKCL is designed to be integrated into applications towards minimizing the “attack-surface” of the application, and thus freeing the applications from the burden of managing cryptographic keys and cryptographic operations.

2.3. PCI-DSS Requirement 3.5

Protect cryptographic keys used for encryption of cardholder data against both disclosure and misuse.

StrongAuth Analysis: For protection from breaches, unencrypted data (plaintext) depend on controls implemented in the operating system, the database management system and/or the network layer. When plaintext is encrypted into ciphertext, the vulnerability of the system shifts from the plaintext to the data-encryption key (DEK) and the focus of data-protection shifts from secure data-management to secure key-management. However, since DEKs are typically used to encrypt many sets of data, and because of the tendency to treat ciphertext with less caution than plaintext, the risk of disclosing DEKs to unauthorized entities is greater. It is, therefore, necessary to treat encryption keys with greater care than with plaintext.

How StrongKey meets this requirement

StrongKey offers a robust key-management solution, using some of the strongest technology and practices honed by the applied-cryptography industry over the last decade. Strong cryptographic algorithms: Triple Data Encryption Standard (3DES) and Advanced Encryption System (AES); large key-sizes for DEKs: 192-bit 3DES, 128-256 bit AES; large key-sizes for keys that encrypt DEKs: 2048-bit RSA keys; Federal Information Processing Standard (FIPS) 140-2 certified Hardware Security Modules (HSM) and smartcards for generating and storing RSA key-pairs; digitally-signed and encrypted messages over the network: – these are just some of the protection mechanisms used by StrongKey to protect cryptographic keys within an Enterprise Key Management Infrastructure (EKMI).

While the security of any environment depends on more than just technology, StrongKey provides significant flexibility in choices of hardware, software and cryptographic components in protecting keys from disclosure and misuse.

2.4. PCI-DSS Requirement 3.5.1

Restrict access to cryptographic keys to the fewest number of custodians necessary.

StrongAuth Analysis: An optimally secure cryptographic environment is one where only the application that uses sensitive data can access the DEK to decrypt it. In such an environment, even Key Custodians and Security Officers of the EKMI would not have access to DEKs. This may appear illogical; but is, nonetheless, possible within the field of cryptography through “chains of control” of cryptographic keys.

In the past, when there were few applications that needed to encrypt data, most key-management procedures were manual, requiring intervention from human Key Custodians in most activities. However, on an internet-scale networked environment, such human-intervention for every key-management procedure is neither desirable nor feasible.

Using “chains of control”, it is possible to establish key-management infrastructures with Key Custodians, and delegate the responsibility for day-to-day controls of key-management to a trusted key-management service. Not only does this scale, but it reduces the cost of ownership of the key-management infrastructure without compromising on security.

How StrongKey meets the requirement

StrongKey uses a sophisticated “Chain of Control” mechanism, allowing Key Custodians (called EKMI Custodians within the StrongKey environment) to be divorced from the generation, transport and access of the DEK. Security Officers within an EKMI establish policies that direct which key(s) can be accessed by application(s) and exclude Key Custodians and Security Officers from gaining access to DEKs.

Once an EKMI has been established, all DEKs generated and accessed by applications are managed directly by StrongKey and do not require the involvement of EKMI Custodians. The custodians are required only when the StrongKey service needs to be started up during a system boot, or when a new Symmetric Key Services (SKS) server needs to be established. Security Officers are able to revoke and suspend DEKs, modify key-use and access-control policies without having access to the DEKs themselves. All policy changes are recorded with digital signatures for audit purposes so that they cannot be tampered.

2.5. PCI-DSS Requirement 3.5.2

Store cryptographic keys securely in the fewest possible locations and forms.

StrongAuth Analysis: Ideally, all cryptographic keys are stored centrally in a secure environment, and applications that need them request and get them only for needed durations. A network-based key-management service, modeled along the lines of Domain Name Service (DNS) or Dynamic Host Configuration Protocol (DHCP), etc., is an optimal design for such a capability.

Unfortunately, StrongAuth has witnessed many implementations where software developers have implemented “home-grown” key-management into their applications. Secure storage of cryptographic keys has included “hiding” keys in “hidden” files and/or directories, storing them in operating system registries, compiling them into applications and/or using proprietary algorithms to recreate a DEK at run-time from cryptographic parameters.

Today's attackers are familiar with all these tricks. For example, tools have been available for the last decade that show files accessed by a running application which includes their full path-names. Decompilers can reverse the contents of a binary program to a fairly accurate representation of their source-code,. Finally, there is no protection for proprietary algorithms that assemble keys at run-time, since a mere copy of the binary program will reassemble the DEK for the attacker without their having to know the algorithm.

How StrongKey meets this requirement

StrongKey generates and stores ALL DEKs centrally, and “escrows” them encrypted under the SKS server's 2048-bit RSA key even before it sends it to the requesting client. (This ensures that the company is never at risk of having ciphertext in the enterprise, but without a DEK to decrypt it with). The RSA keys are generated and stored on a hardware cryptographic module to protect against tampering or unauthorized access. The DEK is also encrypted under the requesting client's RSA public-key and transported over the network securely so that the payload cannot be used by anyone else other than the original, legitimate requester. The response from the SKS server to the client is also digitally signed to protect against tampering en-route and to prevent spoofing of SKS servers.

Based on Key-Caching policies established centrally, DEKs may be either cached on the client machine temporarily or never cached at all. If cached, the duration of the cache and the number of keys cached are defined by the centrally-defined Key-Cache-Policy. All keys in the cache are always encrypted and digitally-signed to protect them from attack.

2.6. PCI-DSS Requirement 3.6

Fully document and implement all key-management processes and procedures for cryptographic keys used for encryption of cardholder data, including the following (3.6.1 – 3.6.8):

StrongAuth Analysis: PCI-DSS references the NIST Key Management publication (SP 800-57) as one of many guidelines for managing cryptographic keys. While StrongAuth believes that the NIST guidelines are fundamentally sound, they were designed for an environment that assumed manual controls for managing keys in small, controlled environments that had few applications requiring encryption keys. They do not take into account the possibility that encryption keys will be necessary over very large networks for millions of devices and applications. They also do not incorporate technologies prevalent in the market today that allow companies to manage keys easily – and securely – over the network: Web Services Security (WSS), XML Signature, XML Encryption, Trusted Platform Modules (TPM) – these are just some of the technologies that make it possible to build internet-scale key-management infrastructures using secure and highly automated procedures.

How StrongKey meets this requirement

StrongKey relies on industry-standard security components – cryptographic hardware modules, Public Key Infrastructure (PKI), WSS – to create a secure Symmetric Key Management Systems (SKMS). Using “Chains of Control” to separate the establishment of an EKMI from its day-to-day operations, StrongKey embodies sound key-management principles while scaling to meet internet-level demands.

2.7. PCI-DSS Requirement 3.6.1

Generation of strong cryptographic keys.

StrongAuth Analysis: Data-encryption is only as good as the cryptographic keys generated to perform the cryptographic operations. Aside from using a trusted environment for cryptographic operations, a critical element of generating strong cryptographic keys is the use of a true Random Number Generator (RNG). Another is using approved algorithms such as the Triple DES and/or AES.

How StrongKey meets this requirement

StrongKey generates DEKs on FIPS-certified Hardware Security Modules (HSM), which typically provide a true RNG as a product feature. The RSA key-pairs of SKS servers are also generated and stored on the HSM using the true RNG. StrongKey supports the generation of 192-bit 3DES, 128-, 192- and 256-bit AES, and 2048-4096 bit RSA keys on various HSMs and smartcards within EKMI. (StrongAuth is currently working on incorporating the remaining NIST-approved Suite B protocols in the next release of StrongKey).

2.8. PCI-DSS Requirement 3.6.2

Secure cryptographic key distribution.

StrongAuth Analysis: In any large-scale networked environment, a centralized key-management service has many benefits: you centralize critical resources and policy information while minimizing resources that must be expended on client devices. Yet all client devices are capable of accessing the centralized services without knowing the service's intricacies,

However DEKs must be delivered far more securely than most other payloads. The key-distribution source must be authenticated to mitigate spoofing attacks on the client; and the integrity and confidentiality of the payload must always be protected during transit and storage if the payload is to be of any use to the company.

How StrongKey meets this requirement

StrongKey always encrypts the DEK under the requesting client's public-key and digitally signs the response before sending it to the client over the network. This use of "message-level" security enables companies to securely distribute DEKs over the *internet* - not just the intranet - **without the need for Virtual Private Networks (VPN) or Secure Socket Layer (SSL)**.

This contrasts with media-level security (such as SSL of IPsec) that decrypts data well before the payload reaches the application, potentially, allowing an attacker to "wedge" themselves between the encrypting media-layer and the application layer to glean sensitive information. Message-level security, as used by StrongKey, keeps the information secure all the way from the source application-layer to the destination application-layer. The same controls are used to store DEKs securely in the SKS database and the key-cache on the client.

2.9. PCI-DSS Requirement 3.6.3

Secure cryptographic key storage.

StrongAuth Analysis: A key-management infrastructure is only as secure as the protection afforded to its key-store(s). It does little good to use powerful cryptographic algorithms and large key-sizes if the keys themselves have little protection from unauthorized access.

Unfortunately, many people with little knowledge of cryptographic practices assume that "hiding" the DEK in hidden files/directories is secure. Or compiling them into applications hides them from attackers. Or using proprietary algorithms to assemble components of the key at run-time will escape the notice of attackers. Sadly, the professional attackers of today are far more capable than many corporate software developers. StrongAuth has witnessed some extremely sophisticated use of cryptography in publicly known attacks - a level of sophistication yet to be embraced by corporate IT environments or even commercial software developers.

How StrongKey meets this requirement

StrongKey's use of the "Chain of Control" concept allows DEKs to be treated like ordinary files on the operating system or records within a database. However, because DEKs are always encrypted under an SKS server's or the requesting client's public-key, and because the private keys (which can decrypt the encrypted DEKs) of the SKS server and clients are stored on cryptographic hardware modules (which in turn are protected either by smartcards and pass-phrases), access to the decrypted DEK is non-trivial to very difficult for unauthorized entities.

Additionally, StrongKey's use of FIPS-certified cryptographic modules ensures that in the event the hardware module detects attacks on its key-store, the modules can lock-up (requiring a Security Officer to reset the module) and/or zero out (erase) key-material inside the key-store. Most client platforms include this capability by way of the Trusted Platform Module (TPM), but are not utilized in most IT environments.

2.10. PCI-DSS Requirement 3.6.4

Periodic cryptographic key changes

- *As deemed necessary and recommended by the associated application (for example, re-keying); preferably automatically*
- *At least annually*

StrongAuth Analysis: Using a single, or few, DEKs to encrypt all or most sensitive data, and/or using the same DEKs for long durations brings additional risk to the system. Depending on the implementation of the cryptography modules, the controls around the application and its use of the DEKs as well as the procedures for managing the DEK, it is conceivable that risk to the system grows as either more time passes or as more plaintext gets encrypted with the single (or few) keys. (The

principle is similar to having only a single (or few) common stocks in a financial portfolio; while one may get lucky occasionally, statistical evidence shows that a portfolio with a single or few stocks in it is more likely to lose in the long-term in comparison to a portfolio with an array of financial instruments which spreads – and reduces – the risk to the portfolio).

How StrongKey meets this requirement
<p>StrongKey introduces such an innovative approach to addressing this requirement. Since the generation, storage and recovery of DEKs are managed automatically by the SKS server (once the policies and access control rules are defined), getting a new DEK to a client is no different from retrieving an existing DEK from the SKS server. Since StrongKey can manage trillions of DEKs (actually 2^{64} DEKs within a single SKS server, hardware permitting), it is possible for a company to use a unique DEK for encrypting every transaction, if desired. Although practically speaking, companies are likely to define policies that use a new DEK for every N transactions, or every N hours or days. This is the equivalent of spreading the risk of a breach across many thousands or millions of keys. The compromise of any one (or few keys) on a client device does not compromise the SKS server or any other client device's keys and/or data.</p> <p>With such capability, it becomes feasible for applications to use as many keys as required. If an application chooses to re-encrypt data when it rolls over a DEK, it can be accomplished without shutting down the application. A separate thread in the application periodically scans the database and determines which data-records need to be re-encrypted with a new key. It then acquires the old and a new key from the SKS server, decrypts the ciphertext with the old key and re-encrypts it with the new key. As long as the application is thread-safe, another thread accessing the re-encrypted record will be unaware that the key was rolled-over – it will just request the “new” key and decrypt the ciphertext as if it were still encrypted with the “old” key.</p>

2.11. PCI-DSS Requirement 3.6.5

Retirement or replacement of old or suspected compromised cryptographic keys.

StrongAuth Analysis: Businesses with an information retention policy, typically, destroy information records when they are no longer required by policy or by law. If the information was encrypted, it does not make any sense to keep the DEKs when the associated ciphertext has been destroyed. Additionally, a compromised DEK, or a DEK suspected of being compromised, presents a risk to the company: not only must the compromised DEK be replaced, but ciphertext encrypted by the compromised DEK must be re-encrypted as soon as possible with a new DEK so as to avoid a breach to the system.

How StrongKey meets this requirement
<p>Using the same methodology described in the earlier section (for key-rollover), StrongKey can be used to replace compromised DEKs. Retiring DEKs only involves a simple administrative action on the StrongKey Console by an authorized Security Officer. If there is no ciphertext that uses the retired key, the application isn't even involved in this administrative action.</p>

2.12. PCI-DSS Requirement 3.6.6

Split knowledge and establishment of dual control of cryptographic keys

StrongAuth Analysis: In the cryptographic community, it is a standard practice to ensure that a single individual does not possess full control of a cryptographic key. This is, typically, accomplished by breaking up the cryptographic key into multiple parts, and ensuring that each part is protected and managed by a different Key Custodian. The split part may or may not be encrypted using Password-Based-Encryption (PBE) and/or may require two-factor authentication (using a smartcard) for the key-management application to assemble the key into its whole again.

How StrongKey meets this requirement

StrongKey meets this requirement through its “Chain of Control” concept. Initially, the SKS servers are established with their own 2048-bit RSA asymmetric key-pairs on the HSM. Authentication to the HSM requires presenting multiple smartcards and/or pass-phrases to the HSM so the private key of the RSA key-pair can be accessed by StrongKey to start the key-management service. Once operational, StrongKey manages the life-cycle of DEKs automatically, based on policies defined by the Security Officer. However, without the activated private-key on the HSM (which requires split-key knowledge and M of N control), StrongKey cannot access **anything** within its database.

On the client-side, the use of a smartcard, TPM or HSM enables the same chain of control over the client's private key (which then has the ability to decrypt the DEK sent over by the SKS server). This use of cryptographic “Chain of Control” provides for the same level of security as if the DEK itself were split into parts and managed by multiple Key Custodians. But, it allows the key-management infrastructure to scale to the enterprise, cost-effectively and securely. (This model is not really new; it has existed for nearly two decades in the Secure Multipurpose Internet Mail Extensions (S/MIME) and the Secure Socket Layer (SSL) protocols. StrongAuth has merely, extended this model to Symmetric Key Management Systems).

2.13. PCI-DSS Requirement 3.6.7

Prevention of unauthorized substitution of cryptographic keys.

StrongAuth Analysis: A form of attack on a cryptographic system, is to substitute the known DEK with one that was generated by the attacker. Since keys are random binary data, unless there is additional meta-data and a secondary protection mechanism that was established at the time of the DEK's creation, it is very difficult to distinguish one DEK from another. Once an attacker has managed to substitute – or introduce - the known DEK with his/her own generated DEK, he/she can collect ciphertext leisurely for decryption, while custodians remain unaware of the compromise.

How StrongKey meets this requirement

StrongKey uses digital signatures at every stage of the DEK's life-cycle to protect against such an attack. When the DEK is generated inside the HSM, it is immediately encrypted under the SKS server's RSA public key. The encrypted DEK, with its meta-data, is then digitally signed using the SKS server's RSA private key. The digitally signed object is now stored in the SKS server's database. Upon reading this object from the database, StrongKey verifies the digital signature before attempting to use the object. If the signature cannot be verified, StrongKey refuses to use the object (and logs/emails a security alert). Since the private-key of the SKS server is inside an HSM and access to it controlled through multiple EKMI Custodians, an attacker on the SKS server is unlikely to introduce/substitute DEKs on the SKS server.

Every DEK sent over to a client, is encrypted under the requesting client's RSA public key and digitally signed by the SKS server's RSA private key. The SKCL, upon receiving the response from the SKS server, first verifies the digital signature to make sure the response is from a trusted SKS server and the payload has not been tampered en route. When the signature is verified successfully, the SKCL decrypts the payload for use. All DEKs are stored encrypted and digitally-signed in the client's key-cache; every key retrieved from the client key-cache is checked to ensure the digital signature can be verified before the payload is decrypted for use.

Since this integrity protection is enabled at the creation of the DEK, and is maintained throughout the DEK's life-cycle, it is very difficult for an attacker to substitute a DEK without compromising the controls around the private keys of the clients and/or SKS servers. Since those private keys are in cryptographic hardware modules using split-key knowledge and M of N controls, the attack becomes nearly impossible in a well-managed EKMI.

2.14. PCI-DSS Requirement 3.6.8

Requirement for cryptographic key custodians to sign a form stating that they understand and accept their key-custodian responsibilities

StrongAuth Analysis: Since the intricacies of cryptography and key-management are relatively new concepts to most IT people, Key Custodians should be given comprehensive training before they accept their responsibilities. In addition, users of applications on client-devices effectively become Key Custodians on their own devices. As such, they should also be trained on the proper protection of cryptographic keys on their devices. Not providing such training is indicative of a lack of seriousness on the part of the company implementing key-management.

How StrongKey meets this requirement

This is a procedural requirement that is addressed outside the realm of technology.

3. Conclusion

The PCI-DSS 1.2 requirements for encryption and key-management are an effective risk-mitigation strategy when all parts of the cryptographic system are designed, implemented and operated correctly. However, the lack of detail in the standard leaves much room for interpretation. While it is possible for general information technology professionals to learn and understand cryptography and key-management, it is best not to second-guess the intent of the DSS when choosing a design and implementation. Lack of knowledge, or misinterpretation of a requirement is not construed as a defense even if one has passed the the PCI audit. Attackers are not interested in your audit compliance - they are only interested in vulnerabilities in your system.; if there is one, the professional attackers will find it.

4. About StrongAuth, Inc.

StrongAuth, Inc. is a privately held California corporation, founded in July 2001, whose focus is Enterprise Key Management, PKI-based Identity Management and Compliance Workflow Management. Over these years, StrongAuth has solved complex risk-management problems for its customers by creating the following path-breaking solutions:

- **EKMI Appliance™** – An integrated appliance that includes hardware, software and services to setup an Enterprise Key Management Infrastructure for managing cryptographic keys across the enterprise. In addition to managing data-encryption keys using StrongKey, the EKMI Appliance also provides the capability to issue strong-authentication credentials to humans, applications and devices to enable consolidated Identity Management.
- **StrongKey™** – An open-source symmetric encryption key-management software product that creates a new paradigm for how encryption keys may be managed across the enterprise, in an application-independent manner. The protocol within StrongKey is the basis for a potential standard for the Symmetric Key Services Markup Language (SKSML) by the OASIS EKMI Technical Committee;
- **CSRTool™** – An open-source utility for generating RSA/ECDSA cryptographic key-pairs and combining them with associated digital certificates to create secure portable containers, allowing them to be transported to applications/servers.
- **SENSIDIEM™** – A Breach-Disclosure compliance workflow application that allows executives responsible for information protection to know their sensitive assets in the company, their stakeholders and controls, with a tool to track breaches through the investigations (with storage of forensic data) to their eventual disclosure (if required by law).

With customers in the Bio-Technology, Retail, Pharmaceutical, Financial, Technology, Consulting and Transportation sectors, StrongAuth has helped some of the largest companies in the world with the architecture, implementation and operations of complex key-management infrastructures.

StrongAuth actively participates on various technology forums, with the goal of advancing standards in the fields that it specializes in. StrongAuth employees are active in OASIS and currently hold the Chair position in the Enterprise Key Management Infrastructure Technical Committee. More information on StrongAuth can be found at www.strongauth.com or by contacting us at info@strongauth.com.