# *The StrongAuth Key Appliance and the PCI-DSS 3.0 Requirements*

***Version: 1.0***

***June 1, 2014***

StrongAuth, Inc.
10846 Via San Marino
Cupertino CA 95014-6333
USA

June 2014

## Copyrights & Notices

## Introduction

The Payment Card Industry (PCI) released version 3.0 of its Data Security Standard (DSS) in November 2013 (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 only some sections of the DSS (pertaining to the StrongAuth KeyAppliance or SAKA), 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.

The format of this paper presents the PCI-DSS requirement first, followed by how the StrongAuth Key Appliance (SAKA) appliance 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 the SAKA in your environment.

## System Security

### 2.1 PCI-DSS Requirement 2.1

*Always change vendor-supplied defaults and remove or disable unnecessary default accounts before installing a system on the network. This applies to ALL default passwords, including but not limited to those used by operating systems, software that provides security services, application and system accounts, point-of-sale (POS) terminals, Simple Network Management Protocol (SNMP) community strings, etc.).*

| **How the SAKA meets this requirement** |
|---|
| This requirement must be fulfilled by customer sites. StrongAuth supplies instructions as part of the SAKA documentation on how to change all default passwords on the system. While the appliance is capable of supporting an SNMP agent, the default configuration does not make this agent accessible – the appliance's host-based firewall blocks SNMP access. However, if a customer site chooses to manage the appliance using SNMP, they are responsible for the security of the SNMP agent. |

#### 2.1.1 PCI-DSS Requirement 2.1.1

*For wireless environments connected to the cardholder data environment or transmitting cardholder data, change ALL wireless vendor defaults at installation, including but not limited to default wireless encryption keys, passwords, and SNMP community strings.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA does not use any wireless networks. If the customer-site has a wireless network that accesses the SAKA, this requirement must be fulfilled by the customer. |

### 2.2 PCI-DSS Requirement 2.2

*Develop configuration standards for all system components. Assure that these standards address all known security vulnerabilities and are consistent with industry-accepted system hardening standards.*

*Sources of industry-accepted system hardening standard may include, but are not limited to:*

- *Center for Internet Security (CIS)*
- *International Standards Organization (ISO)*
- *SysAdmin Audit Network Security (SANS) Institute*
- *National Institute of Standards and Technology (NIST)*

| **How the SAKA meets this requirement** |
|---|
| StrongAuth does not, currently, use any of the above-mentioned hardening standards; it bases its security configuration on the recommendations of the manufacturer of the operating system on the appliance. Since the operating system of the SAKA is a derivative of Red Hat Enterprise Linux (RHEL), StrongAuth uses the RHEL Security Guide as the basis for hardening the SAKA.<br><br>We recognize that the hardening standards mentioned in the DSS list will result in a secure system. We also believe that the operating system manufacturer is most likely to have the most up-to-date information on vulnerabilities and attack-vectors associated with their product, and will likely have the most current information on securing systems using their product. However, we are reviewing the different hardening standards mentioned in the DSS to ensure there are no gaps between the vendor's and the independent guidelines. |

### 2.2.1 PCI-DSS Requirement 2.2.1

*Implement only one primary function per server to prevent functions that require different security levels from co-existing on the same server. (For example, web servers, database servers, and DNS should be implemented on separate servers.)*

*Note: Where virtualization technologies are in use, implement only one primary function per virtual system component.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA performs only one function: the cryptographic functions associated with encryption and tokenization of sensitive data. In order to protect sensitive data within the appliance, it has built-in key-management functionality which uses a Common Criteria (CC) certified or Federal Information Processing Standards (FIPS) certified cryptographic hardware module (depending on the appliance model), a true random number generator (TRNG) and strong cryptographic algorithms. |

### 2.2.2 PCI-DSS Requirement 2.2.2

*Enable only necessary and secure services, protocols, daemons, etc., as required for the function of the system.*

| **How the SAKA meets the requirement** |
|---|
| The SAKA disables all unneeded services, protocols and daemons. It also removes unnecessary packages from the operating system. |

### 2.2.3 PCI-DSS Requirement 2.2.3

*Implement additional security features for any required services, protocols, or daemons that are considered to be insecure—for example, use secured technologies such as SSH, S-FTP, SSL, or IPSec VPN to protect insecure services such as NetBIOS, file-sharing, Telnet, FTP, etc.*

| **How the SAKA meets the requirement** |
|---|
| Only two ports are accessible by default to the general-purpose network: <br>• 22 for secure shell access (SSH) by authorized individuals; and <br>• 8181 for cryptographic web-services over Secure Socket Layer (SSL) or Transport Layer Security (TLS). |

### 2.2.4 PCI-DSS Requirement 2.2.4

*Configure system security parameters to prevent misuse.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA installation and configuration process includes steps to secure the appliance. These steps can be reviewed by viewing the *install-saka.sh* script in the /usr/local/software/saka folder. Additionally, a *harden.sh* script is sent to customers to enable specific audit control parameters. Customers, upon consultation and concurrence from StrongAuth, may add additional controls to the appliance to enhance the security of the appliance, if desired. Customers are at liberty to add any control outside the appliance, to secure access to the appliance. |

### 2.2.5 PCI-DSS Requirement 2.2.5

*Remove all unnecessary functionality, such as scripts, drivers, features, subsystems, file systems and unnecessary web servers.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA removes unnecessary packages from the base operating system, and disables unneeded services and features. Customers are also discouraged from giving the appliance access to the internet unless they choose to enable the relay of transactions from the appliance to a targeted payment processor over SSL/TLS. StrongAuth supports the relay of PAN directly to payment gateways to minimize the need for decrypted PAN within the customer's cardholder data environment (CDE) |

## 2.3 PCI-DSS Requirement 2.3

*Encrypt all non-console administrative access using strong cryptography. Use technologies such as SSH, VPN, or SSL/TLS for web-based management and other non-console administrative access.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA only enables two ports for administrative access: port 22 for secure shell (SSH) access and port 8181 for access to administrative web-services using SAKA administrative tools: the Key Custodian SetPIN Tool and the Domain Administration Console Tool. These tools operate only over SSL/TLS and requires strong authentication with personal digital certificates and keys issued by the appliance to authorized individuals before administrative access is granted. No other port offers administrative access to the appliance in the default configuration. |

## 2.4 PCI-DSS Requirement 2.4

*Maintain an inventory of system components that are in scope for PCI DSS.*

| How the SAKA meets this requirement |
|---|
| This requirement must be fulfilled by customer sites and their hosting providers, if any. However, the SAKA is always within scope for PCI-DSS when used to protect PAN. |

### 2.5 PCI-DSS Requirement 2.5

*Ensure that security policies and operational procedures for managing vendor defaults and other security parameters are documented, in use, and known to all affected parties.*

| How the SAKA meets this requirement |
|---|
| This requirement must be fulfilled by customer sites and their hosting providers, if any. |

### 2.6 PCI-DSS Requirement 2.6

*Shared hosting providers must protect each entity's hosted environment and cardholder data. These providers must meet specific requirements as detailed in Appendix A: Additional PCI DSS Requirements for Shared Hosting Providers.*

| How the SAKA meets this requirement |
|---|
| This requirement must be fulfilled by customer sites and their hosting providers, if any. |

## Encryption and Key Management

### 3.4. PCI-DSS Requirement 3.4

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

- *One-way hashes based on strong cryptography (hash must be of the entire PAN)*
- *Truncation (hashing cannot be used to replace the truncated segment of PAN)*
- *Index tokens and pads (pads must be securely stored)*
- *Strong cryptography with associated key-management processes and procedures*

*Note: It is a relatively trivial effort for a malicious individual to reconstruct original PAN data if they have access to both the truncated and hashed version of a PAN. Where hashed and truncated versions of the same PAN are present in an entity's environment, additional controls should be in place to ensure that the hashed and truncated versions cannot be correlated to reconstruct the original PAN.*

| How the SAKA meets this requirement |
|---|
| The SAKA does not generate message-digests, but supports Hashed Message Authentication Codes (HMAC) to create unique identifiers internally within the SAKA. HMACs require symmetric cryptographic keys and these keys are generated and managed automatically by the appliance. The SAKA generates HMACs based on the HmacSHA256, HmacSHA384 and the HmacSHA512 algorithms.<br><br>Applications may choose to use the HMAC or a Psuedo-Number (PSN) – a substituted value which resembles a PAN - through *Tokenization*. Using the PSN as a token, applications can be effectively taken out of PCI-DSS scope for encryption and key-management controls, since all keys and PAN are stored only on the SAKA; neither the keys nor the sensitive data are stored on the client application. |

### 3.4.1. 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 mechanisms (for example, by not using local user account databases). Decryption keys must not be associated with user accounts.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA does not use disk-based, file and/or column-level database encryption. The SAKA only supports application-level encryption through its web-services. Applications must make an explicit authorized call to encrypt/decrypt the PAN on the SAKA. By eliminating the encrypted data and keys from the application machine and databases, the SAKA minimizes the "attack-surface" of the application. |

**Note**: 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.

## 3.5. PCI-DSS Requirement 3.5

*Document and implement procedures to protect keys used to secure stored cardholder data against disclosure and misuse:*

*Note: This requirement applies to keys used to encrypt stored cardholder data, and also applies to key-encrypting keys used to protect data-encrypting keys—such key-encrypting keys must be at least as strong as the data-encrypting key.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA offers one of the most 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: the Advanced Encryption System (AES); large key-sizes for Data Encryption Keys (DEK): 128-256 bit AES; large key-sizes for Key Encryption Keys (KEK) that encrypt DEKs: 2048-bit RSA keys; Federal Information Processing Standard (FIPS) 140-3 certified Hardware Security Modules (HSM); Common Criteria EAL 4+ certified Trusted Platform Modules (TPM) and digitally-signed and encrypted messages over the network for Key Custodians and Encryption Domain Administrators: – these are just some of the protection mechanisms used by the SAKA to protect cryptographic keys.<br><br>While the security of any environment depends on more than just technology, the SAKA provides significant levels of integration in hardware, software and cryptographic components towards protecting keys from disclosure and misuse. |

### 3.5.1. PCI-DSS Requirement 3.5.1

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

| **How the SAKA meets the requirement** |
|---|
| The SAKA uses a sophisticated "Chain of Control" mechanism, allowing Key Custodians to be divorced from the generation, transport and access of the DEK. Security Officers establish policies that direct which users can request encryption, decryption or both services and can exclude Key Custodians from gaining access to DEKs.<br><br>Once an SAKA is installed, all DEKs generated and accessed by applications are managed directly by the SAKA and do not require the involvement of Key Custodians. The custodians are required only to activate the cryptographic hardware module when the SAKA service needs to be started up after a system reboot, when a new encryption domain needs to be created, or when an Encryption Domain Key is being migrated from one SAKA to another (for high-availability or disaster recovery). |

### 3.5.2. PCI-DSS Requirement 3.5.2

*Store secret and private keys used to encrypt/decrypt cardholder data in one (or more) of the following forms at all times:*

- *Encrypted with a key-encrypting key that is at least as strong as the data-encrypting key, and that is stored separately from the data-encrypting key*
- *Within a secure cryptographic device (such as a host security module (HSM) or PTS-approved point-of-interaction device)*
- *As at least two full-length key components or key shares, in accordance with an industry-accepted method*

*Note: It is not required that public keys be stored in one of these forms.*

| How the SAKA meets this requirement |
|---|
| The SAKA generates and stores ALL DEKs centrally, and "escrows" them encrypted under the Encryption Domain's 2048-bit RSA key-encrypting-key. The Encryption Domain's RSA keys are encrypted by another 2048-bit RSA key-pair generated and stored on a hardware cryptographic module to protect against tampering or unauthorized access. With the exception of the keys inside the hardware module, all keys are replicated to redundant SAKA appliances securely for disaster recovery. None of the keys are ever transported to requesting applications. None of the roles involved in the SAKA – Key Custodians, Encryption Domain Administrators or the applications – are ever given access to the DEKs directly when encrypting PANs. The SAKA is the only application/device that handles the DEKs, KEKs and hardware module key upon receiving authorized web-service requests. |

### 3.5.3. PCI-DSS Requirement 3.5.3

*Store cryptographic keys in the fewest possible locations.*

| How the SAKA meets the requirement |
|---|
| The SAKA stores the data-encryption-keys (DEK), key-encrypting-keys (KEK) and the hardware module key (HMK) only on the SAKA itself. It never releases these keys outside the SAKA, except for disaster recovery purposes to be migrated to another SAKA securely, under the protection of the Key Custodians and a SAKA-specific migration-key. Applications using the SAKA webservices never see these keys outside the appliance. |

## 3.6. 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:*

*Note: Numerous industry standards for key management are available from various resources including NIST, which can be found at http://csrc.nist.gov.*

| How the SAKA meets this requirement |
|---|
| The SAKA relies on industry-standard security components to create a secure Symmetric Key Management Systems (SKMS). Using "Chains of Control" to separate the establishment of an SKMS from its day-to-day operations, the SAKA embodies sound key-management principles while scaling to meet internet-level demands. |

### 3.6.1. PCI-DSS Requirement 3.6.1

*Generation of strong cryptographic keys.*

| How the SAKA meets this requirement |
|---|
| The SAKA generates DEKs using content from FIPS-certified Hardware Security Modules (HSM) or Common Criteria EAL4+ approved Trusted Platform Modules (TPM), both of which provide a True Random Number Generator (TRNG) as a standard feature. The SAKA only generates 128-, 192- and 256-bit AES symmetric keys for DEKs and HMACs, and 2048-bit RSA keys for KEKs. |

### 3.6.2. PCI-DSS Requirement 3.6.2

*Secure cryptographic key distribution.*

**How the SAKA meets this requirement**

The SAKA does not hand out the DEK, KEK and HMK to any requesting application. All keys are generated, stored and used within the appliance and only replicated to another appliance securely. However, even if the encrypted keys fall into the wrong hands, without the cryptographic hardware module on the specific appliance, it is computationally infeasible to decrypt them. This is because the Root 2048-bit key that can decrypt the chain is unavailable outside the cryptographic hardware module, which is a standard component of the SAKA.

### 3.6.3. PCI-DSS Requirement 3.6.3

*Secure cryptographic key storage.*

**How the SAKA meets this requirement**

The SAKA' 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 Encryption Domain's key-encrypting-key, and because the private key (which can decrypt the encrypted keys) of the root key (HMK) is stored inside cryptographic hardware modules (which in turn are protected by multiple Key Custodians), access to the decrypted DEK is non-trivial to very difficult for unauthorized entities.

The standard use of certified cryptographic hardware modules in the SAKA ensures that in the event the hardware module detects attacks, the modules can either lock-up (requiring a Security Officer to reset the module), zero out (erase) key-material inside the key-store and/or introduce increasing delays in responses when subjected to dictionary and/or rainbow-table attacks.

### 3.6.4. PCI-DSS Requirement 3.6.4

*Cryptographic key changes for keys that have reached the end of their crypto-period (for example, after a defined period of time has passed and/or after a certain amount of cipher-text has been produced by a given key), as defined by the associated application vendor or key owner, and based on industry best practices and guidelines (for example, NIST Special Publication 800-57).*

**How the SAKA meets this requirement**

The SAKA includes the ability to automatically rotate DEKs without shutting down cryptographic services or applications that rely on them. Since keys never leave the SAKA, and applications store a "token" to uniquely identify a PAN, the SAKA can rotate keys at will, or periodically, without changing the token, thus isolating applications from this house-keeping duty.

The default setting of the SAKA is to rotate DEKs once a year; however, a site can reconfigure its SAKA in minutes to change this setting to rotate as frequently as necessary – including once a week if business requirements permit.

### 3.6.5. PCI-DSS Requirement 3.6.5

*Retirement or replacement (for example, archiving, destruction, and/or revocation) of keys as deemed necessary when the integrity of the key has been weakened (for example, departure of an employee with knowledge of a clear-text key), or keys are suspected of being compromised.*

*Note: If retired or replaced cryptographic keys need to be retained, these keys must be securely archived (for example, by using a key encryption key). Archived cryptographic keys should only be used for decryption/verification purposes.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA does not provide DEKs to anyone; it uses them internally within the application to provide cryptographic web-services to authorized applications on the network. Any storage of DEKs (or KEKs) is only in an encrypted state. As such, knowledge of a clear-text key is highly unlikely in the normal use of the SAKA.<br><br>However, in the unlikely event that a zero-day attack renders a DEK in the clear-text, using the same methodology described in the earlier section (for key-rollover), the SAKA can be used to replace compromised DEKs at will at any time without affecting the applications and their database contents.<br><br>The SAKA also provides software "jobs" to delete encrypted PANs when they are past their data-retention period (which is customizable). Once all encrypted PANs are deleted from the SAKA, site personnel have the ability to permanently delete DEKs from the SAKA appliance, if desired. |

### 3.6.6. PCI-DSS Requirement 3.6.6

*If manual clear-text cryptographic key management operations are used, these operations must be managed using split knowledge and dual control (for example, requiring two or three people, each knowing only their own key component, to reconstruct the whole key).*

*Note: Examples of manual key management operations include, but are not limited to: key generation, transmission, loading, storage and destruction.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA does not allow any user, administrator or Key Custodian to see or handle clear-text keys. Notwithstanding this security feature, the SAKA meets uses a "Chain of Control" concept to secure the activation of the appliance's Root key.<br><br>The installation of the SAKA appliances requires a minimum of two (for HSMs) or three (for TPMs) key-custodians who are issued strong credentials (2048-bit RSA key-pairs). These credentials must be used to digitally sign commands that activate the cryptographic hardware module within the SAKA. Until **all** key-custodians activate the cryptographic hardware module, cryptographic services will not be enabled even if the database and application server are running.<br><br>This is because only the combined authorization of all Key Custodians activates the Root key in the cryptographic hardware module, which in turn decrypts the Encryption Domain Key (the KEK). The decryption of the Encryption Domain Key enables the decryption of all DEKs and HMAC keys within that specific Encryption Domain. Without the activation of the Root key, it is computationally infeasible to use any of the other keys on the appliance. |

### 3.6.7. PCI-DSS Requirement 3.6.7

*Prevention of unauthorized substitution of cryptographic keys.*

| **How the SAKA meets this requirement** |
|---|
| The SAKA automatically generates, encrypts and manages all DEKs and HMAC keys. In order to substitute keys inside the SAKA, attackers must not only compromise the operating system security and database security, but must also compromise the cryptographic hardware module to access its key-pair to encrypt the DEK and substitute it in the database. While there is no such things as perfect security, this degree of compromise requires a significant level of access to the SAKA appliance that is normally detected though other PCI-DSS controls. |

### 3.6.8. PCI-DSS Requirement 3.6.8

*Requirement for cryptographic key custodians to formally acknowledge that they understand and accept their key-custodian responsibilities.*

| **How the SAKA meets this requirement** |
|---|
| This is a procedural requirement that is addressed outside the realm of technology. |

## Secure Systems and Applications

The PCI-DSS requires that applications and systems storing and/or processing PAN be secure. Since StrongAuth itself does not store any PAN, it does not need to be PCI-DSS compliant. However, the SAKA, being a component of a customer site's infrastructure that stores and processes PANs facilitates a part of the process. This section of the document provides information to help a customer document how the SAKA helps them stay compliant to section 6.0 of the PCI-DSS.

Note: StrongAuth does not provide any guarantees about making a customer's site PCI-DSS compliant. While StrongAuth's products and services can help a customer achieve PCI-DSS compliance, the customer is ultimately, and solely, responsible for ensuring PCI-DSS compliance of their systems.

### 6.1. PCI-DSS Requirement 6.1

*Establish a process to identify security vulnerabilities, using reputable outside sources for security vulnerability information, and assign a risk ranking (for example, as “high,” “medium,” or “low”) to newly discovered security vulnerabilities.*

*Note: Risk rankings should be based on industry best practices as well as consideration of potential impact. For example, criteria for ranking vulnerabilities may include consideration of the CVSS base score, and/or the classification by the vendor, and/or type of systems affected.*

*Methods for evaluating vulnerabilities and assigning risk ratings will vary based on an organization’s environment and risk-assessment strategy. Risk rankings should, at a minimum, identify all vulnerabilities considered to be a “high risk” to the environment. In addition to the risk ranking, vulnerabilities may be considered “critical” if they pose an imminent threat to the environment, impact critical systems, and/or would result in a potential compromise if not addressed. Examples of critical systems may include security systems, public-facing devices and systems, databases, and other systems that store, process, or transmit cardholder data.*

| **How the SAKA meets this requirement** |
|---|
| StrongAuth subscribes to alert services to receive information about vulnerabilities to the underlying components of the SAKA appliance. In addition, StrongAuth personnel periodically check vendor websites for information regarding updates to their products. Where applicable and necessary, StrongAuth will notify SAKA customers of updates to the appliance and how the customer site may install them.<br><br>It should, however, be noted that the SAKA is recommended to be in the deepest part of the CDE, in a sub-network dedicated to the SAKA appliances, with access control rules that forbid any non-PCI related system from even reaching the SAKA. The additional controls on the network switch and the built-in firewall of the SAKA reduce the risk of vulnerabilities on the SAKA. |

### 6.2. PCI-DSS Requirement 6.2

*Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor-supplied security patches. Install critical security patches within one month of release. Note: Critical security patches should be identified according to the risk ranking process defined in Requirement 6.1.*

| **How the SAKA meets this requirement** |
|---|
| StrongAuth subscribes to alert services to receive information about vulnerabilities to the underlying components of the SAKA appliance. In addition, StrongAuth personnel periodically check vendor websites for information regarding updates to their products. Where applicable and necessary, StrongAuth will notify SAKA customers of updates to the appliance and how the customer site may install them.<br><br>It should, however, be noted that the SAKA is recommended to be in the deepest part of the CDE, in a sub-network dedicated to the SAKA appliances, with access control rules that forbid any non-PCI related system from even reaching the SAKA. The additional controls on the network switch and the built-in firewall of the SAKA reduce the risk of vulnerabilities on the SAKA. |

## *6.3. PCI-DSS Requirement 6.3*

*Develop internal and external software applications (including web-based administrative access to applications) securely, as follows:*

- *In accordance with PCI DSS (for example, secure authentication and logging)*
- *Based on industry standards and/or best practices.*
- *Incorporating information security throughout the software-development life cycle*

*Note: this applies to all software developed internally as well as bespoke or custom software developed by a third party.*

| **PCI-DSS Requirement** | **How the SAKA meets this requirement** |
|---|---|
| *6.3.1 Remove development, test and/or custom application accounts, user IDs, and passwords before applications become active or are released to customers.* | This requirement must be fulfilled by customer sites for SAKA appliances deployed at their site. All Production appliances shipped by StrongAuth must be installed and configured at the customer site with Production credentials and keys. |

| **PCI-DSS Requirement** | **How the SAKA meets this requirement** |
|---|---|
| *6.3.2 Review of custom code prior to release to production or customers in order to identify any potential coding vulnerability (using either manual or automated processes) to include at least the following.*<br><br>*Code changes are reviewed by individuals other than the originating code author, and by individuals knowledgeable about code-review techniques and secure coding practices.*<br><br>*• Code reviews ensure code is developed according to secure coding guidelines*<br><br>*• Appropriate corrections are implemented prior to release.*<br><br>*• Code-review results are reviewed and approved by management prior to release.*<br><br>*Note: This requirement for code reviews applies to all custom code (both internal and public-facing), as part of the system development life cycle.*<br><br>*Code reviews can be conducted by knowledgeable internal personnel or third parties. Public-facing web applications are also subject to additional controls, to address ongoing threats and vulnerabilities after implementation, as defined at PCI DSS Requirement 6.6.* | The SAKA does not use any custom code – all code delivered on the appliance is delivered as standard software. If customers modify the standard SAKA software, this requirement must be fulfilled by them.<br><br>While the SAKA software license permits customers to make custom modifications to the SAKA software, sites under support contracts with StrongAuth should not modify the standard SAKA code to avail support services.<br><br>In the event customer sites need specific capability not present in the SAKA currently, they are encouraged to contact StrongAuth to discuss the possibility of StrongAuth making such capability a standard feature of the appliance; most of the features in the SAKA are based on features requested by customers. |

## 6.4. PCI-DSS Requirement 6.4

*Follow change control processes and procedures for all changes to system components. The processes must include the following:*

| **PCI-DSS Requirement** | **How the SAKA meets this requirement** |
|---|---|
| *6.4.1 Separate development/test and production environments* | While this requirement must be implemented by customer sites, StrongAuth strongly recommends the acquisition of a separate SAKA for the customer's development environment. Since the root-key of the SAKA in the cryptographic hardware module is a completely separate key, it is impossible to take encrypted data, keys and key-encryption-keys from any production SAKA to a development SAKA for processing. This guarantees separation of environments at the cryptographic level. |
| *6.4.2 Separation of duties between development/test and production environments* | This requirement must be fulfilled by customer sites. |

| **PCI-DSS Requirement** | **How the SAKA meets this requirement** |
|---|---|
| *6.4.3 Production data (live PANs) are not used for testing or development*<br><br>*6.4.4 Removal of test data and accounts before production systems become active* | This requirement must be fulfilled by customer sites. The SAKA does not distinguish between live PANs or test data.<br><br>This requirement must be fulfilled by customer sites. |
| *6.4.5 Change control procedures for the implementation of security patches and software modifications. Procedures must include the following:* | |
| *6.4.5.1 Documentation of impact* | For the impact on PANs, this requirement must be fulfilled by customer sites.<br><br>StrongAuth evaluates the potential impact of changes to SAKA components, to ensure that the security of the appliance is not compromised. |
| *6.4.5.2 Document change approval by authorized parties* | For changes to system components in the customer's infrastructure, this requirement must be fulfilled by customers.<br><br>Any changes to SAKA components are always signed off by StrongAuth management before implementation. |
| *6.4.5.3 Functionality testing to verify that the change does not adversely impact the security of the system* | For customer infrastructure, this requirement must be fulfilled by customers.<br><br>All changes to SAKA components are tested before release to customers. |
| *6.4.5.4 Back-out procedures* | For customer infrastructure, this requirement must be fulfilled by customers.<br><br>SAKA software is maintained in a software repository under the control of strong-authentication. Any change that jeopardizes the integrity of the SAKA appliance can be backed-out to a previous release at any time. |

## 6.5. PCI-DSS Requirement 6.5

Address common coding vulnerabilities in software-development processes as follows:

- *Train developers in secure coding techniques, including how to avoid common coding vulnerabilities, and understanding how sensitive data is handled in memory.*
- *Develop applications based on secure coding guidelines.*

*Note: The vulnerabilities listed at 6.5.1 through 6.5.10 were current with industry best practices when this version of PCI DSS was published. However, as industry best practices for vulnerability management are updated (for example, the OWASP Guide, SANS CWE Top 25, CERT Secure Coding, etc.), the current best practices must be used for these requirements.*

| **PCI-DSS Requirement** | **How the SAKA meets this requirement** |
|---|---|
| *6.5.1 Injection flaws, particularly SQL injection. Also consider OS Command Injection, LDAP and XPath injection flaws as well as other injection flaws.* | The SAKA servlet does not use Structured Query Language (SQL) to communicate with its internal database (IDB) – it uses Enterprise Java Beans (EJB), which in turn use Java Persistence API (JPA) to interact with the IDB. As a result, SQL-injection attacks do not work on the SAKA messaging protocol.<br><br>The same holds true for OS Command Injection, LDAP and XPath injection – the design of the SAKA web-services prevent it from being susceptible to such attacks. |
| *6.5.2 Buffer overflow* | The SAKA appliance parses all parameters for valid values and ensures that they are in conformance with expected values and sizes. |
| *6.5.3 Insecure cryptographic storage* | The SAKA appliance uses either a FIPS 140-2 certified, or a CC EAL4+ certified cryptographic hardware module to store the root key of the SAKA key-hierarchy. All key-encrypting-keys under the root-key are encrypted when stored on the hard-disk of the appliance. All data-encrypting-keys under the key-encrypting-keys are encrypted when stored on the hard-disk of the appliance. |
| *6.5.4 Insecure communications* | The SAKA appliances generates its own Secure Socket Layer (SSL) certificates for secure communication with clients. Not only does this provide message confidentiality and integrity on the network, but it has the added benefit that computers that do not know of the SAKA's SSL certificate will receive error messages when communicating with the appliance.<br><br>Furthermore, the SAKA can be equipped to use SSL client-authentication so that client systems must also possess an X509 digital certificate when communicating with the SAKA. However, the client and appliance's SSL certificates must come from a Public Key Infrastructure (PKI) to enable this level of security. |
| *6.5.5 Improper error handling* | The SAKA appliance logs all application level errors with a unique Error ID (EID) and message. Additionally, errors received at the operating system, database and/or application server are logged by each component individually, thus allowing customer sites to tailor their Security Incident Event Management (SIEM) tools to handle errors appropriately. |
| *6.5.6 All "High" vulnerabilities identified in the vulnerability identification process (as defined in PCI-DSS Requirement 6.2)*<br><br>*Note: This requirement is considered a best practice until June 30, 2012, after which it becomes a requirement.* | StrongAuth evaluates all risks against the possibility of corrupting and/or leaking sensitive data on the SAKA. Upon identifying such vulnerabilities, StrongAuth builds a patch (if necessary) and notifies SAKA customers. |

| **PCI-DSS Requirement** | **How the SAKA meets this requirement** |
|---|---|
| *6.5.7 Cross-site scripting (XSS)* | The SAKA does not use Representational State Transfer (REST)-based messages or HTML (that are typically used to carry out cross-site scripting attacks) between client applications and the appliance; it only uses Simple Object Access Protocol (SOAP)-based messages, accepts and returns only XML. As such, the SAKA appliance is, in theory, immune to XSS attacks. |
| *6.5.8 Improper Access Control (such as insecure direct object references, failure to restrict URL access and directory traversal)* | The SAKA appliance does not return any internal object references. The design of the appliance's key-management software forces a separation of the web-tier (servicing web-service requests) from the EJB-tier which processes the request. All objects passed/returned between the web and EJB tiers are passed by value – not as references.<br><br>The SAKA uses a single URL to provide all services from the appliance. The URL only accepts SOAP messages and neither redirects the calling application to any other URL nor displays any HTML. As such, the use of any other URL on the SAKA appliance would result in an error message to the caller. |
| *6.5.9 Cross-site request forgery (CSRF)* | Just as the SAKA avoids falling prey to XSS attacks through the use of SOAP-based calls and XML objects (as opposed to REST-based calls and HTML), it similarly avoids falling prey to CSRF attacks. |
| *6.5.10 Broken authentication and session management.*<br><br>*Note: Requirement 6.5.10 is a best practice until June 30, 2015, after which it becomes a requirement.* | The SAKA does not use application-level session management - it authenticates and authorizes every web-service request independent of other requests. Underlying SSL/TLS libraries of the customer's application may choose to hold onto the SSL/TLS sessions should they choose to; however, each web-service request always carries authentication credentials to determine the request's authorization of the service. |

## 6.6. PCI-DSS Requirement 6.6

*For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure these applications are protected against known attacks by either of the following methods:*

- *Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes Note: This assessment is not the same as the vulnerability scans performed for Requirement 11.2.*
- *Installing an automated technical solution that detects and prevents web-based attacks (for example, a web-application firewall) in front of public-facing web applications, to continually check all traffic.*

| **How the SAKA meets this requirement** |
|---|
| In general, the SAKA is never deployed as a public-facing web-service. However, if a customer site chooses to do so, it is strongly recommended that the following practices be used to secure access to the SAKA:<br>• Use a reverse-proxy in the demilitarized zone (DMZ) to redirect requests to the SAKA sitting in a secure sub-network rather than place the SAKA in the DMZ;<br>• Setup firewall rules on the internet facing firewall to only allow authorized Transmission Control Protocol/Internet Protocol (TCP/IP) addresses to the reverse-proxy;<br>• Configure the firewall rules on the SAKA to only allow the reverse-proxy to communicate with the SAKA;<br>• Configure SSL/TLS with ClientAuth for connections from external TCP/IP addresses to communicate with the reverse-proxy;<br>• Configure SSL/TLS with ClientAuth for connections from the reverse-proxy to communicate with the SAKA; and<br>• Hardening the reverse-proxy machine. |

### 6.7. PCI-DSS Requirement 6.7

Ensure that security policies and operational procedures for developing and maintaining secure systems and applications are documented, in use, and known to all affected parties.

| **How the SAKA meets this requirement** |
|---|
| This requirement must be fulfilled by customer sites and their hosting providers, if any. |

## Conclusion

The PCI-DSS 3.0 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.

## About StrongAuth, Inc.

StrongAuth, Inc. is a privately held California corporation, founded in July 2001, whose focus is strong-authentication and enterprise key management. StrongAuth has solved complex business problems for its customers by creating path-breaking software to address the problems:

- **StrongAuth KeyAppliance** – The industry's first appliance consisting of encryption, tokenization, key-management and a cryptographic hardware module software integrated to provide the lowest-cost solution for compliance to data-security regulations such as PCI-DSS, 201 CMR 17.00, EU Directive, HIPAA, SB-1386, etc.

- **StrongAuth CryptoDocument Appliance** – An appliance to orchestrate the protection of data-objects and documents across an enterprise based on a standardized *Data Protection Infrastructure (http://www.infoq.com/articles/cloud-data-encryption-infrastructure).*

- **StrongKey CryptoEngine** – An open-source encryption software product that encrypts files of any size and of any type and stores them automatically on public/private clouds, SAN, NAS or local storage, while escrowing the encryption keys on the StrongAuth KeyAppliance. Using the StrongKey CryptoEngine, a site may now offload its data-storage to the cloud environment, while maintaining control of the encryption keys (thus allowing them to prove compliance to security regulations while still using public clouds for storage). The StrongKey CryptoEngine is an integral component of the *Data Protection Infrastructure* and works in concert with the StrongAuth CryptoDocument Appliance;

- **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;
- **PKI Appliance** – An integrated hardware, software and services solution to deploy a Public Key Infrastructure at dramatically reduced costs and time. StrongAuth eliminates the complexity of designing and building a PKI using its reference PKI architecture, honed after 10+ years of building PKIs for some of the largest and smallest companies in the world in diverse industries.
- **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.

With customers in the 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 technological infrastructures. More information on StrongAuth can be found at www.strongauth.com or by contacting us at info@strongauth.com.