

Key management standards and best practices for embedded systems

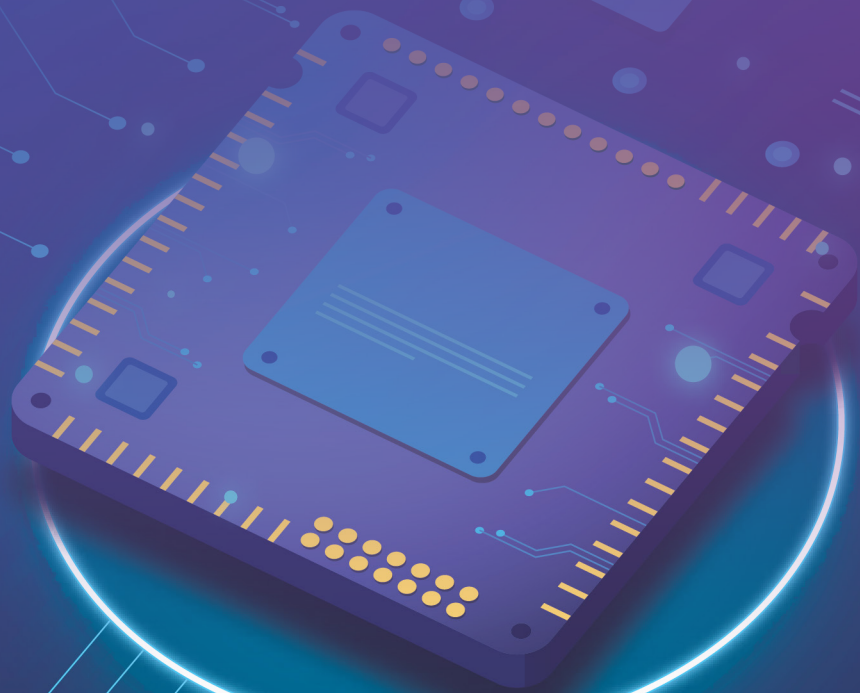


Table of content

Introduction	4
Cryptography and key management	4
Cryptographic keys and their types	5
Authentication	5
Encryption	5
Key wrapping	5
Random number generation	5
Key derivation	5
Transportation	6
Key agreement	6
Authorization	6
Key metadata trusted associations and bindings	7
Key management lifecycle (NIST SP - 800 57)	7
Key states	7
Pre-activation state	8
Activation state	8
Suspended state	8
Deactivated state	8
Compromised state	8
Destroyed state	8
Key management phases	8
Pre-operational phase	10
Operational phase	10
Post-operational phase	10
Key management standards and guidelines	12
Key management best practices	14
Embedded systems best practices for key management	15
Secure boot process	15

Strong encryption algorithms	15
Key hierarchies	15
Secure key storage	16
Key rotation	16
Use randomized keys	16
Authentication and authorization	16
Secure key transport	16
Backup and recovery	16
Logging and auditing	16
Firmware updates	16
Security standards	17
Testing and verification	17
Documentation and training	17
Incident response plan	17
Regular auditing	17
Continual improvement	17
Case study: Secure Boot	17
The process	18
Used cryptographic keys	18
Platform Key (PK)	18
Key Exchange Key (KEK)	18
Signature database	18
Role in key management	19
Key provisioning	19
Chain of trust	19
Authentication	19
Key generation	19
Conclusion	20
References	21
Author info	22

Introduction

Effective key management is a critical component of a robust security architecture. Cryptographic keys safeguard security by ensuring confidentiality, integrity and availability of data, while a key management system protects the key itself and ensures the use of 'active' keys throughout the product lifecycle for consistent data protection.

This whitepaper examines key management practices aligned with key management standards set by the National Institute of Standards and Technology (NIST), ISO, IETF and other standard bodies with a focus on embedded security. Additionally, it highlights best practices for embedded security architects to strengthen the security posture of embedded systems.

We introduce the basics of cryptography and key management principles and guidelines, providing insights into the protocols and principles used in key management.

Next, we review the standards established by various organizations in key management domains.

Furthermore, the HCLTech Product Security team has curated a set of best key management practices specifically tailored for embedded security architects. These practices encompass guidance on key generation, storage, distribution, provisioning, rotation and disposal while addressing the unique challenges associated with embedded security.

In summary, this whitepaper is a comprehensive guide for key management practices in embedded systems.

Cryptography and key management

Cryptographic algorithms are essential for delivering security services in embedded systems. Below is the list of cryptographic algorithms used for these purposes.

Symmetric key algorithm is used to encrypt data in ciphertext in such a way that it cannot be decoded without the appropriate key. The same key is used for encryption and decryption. It ensures confidentiality, source and integrity authentication, key derivation and key wrapping.

Asymmetric key algorithm utilizes two related keys, where a cryptographic operation performed by one key can only be undone by the other key. One key is used for encryption and the other for decryption. This algorithm facilitates key agreement, key transport and source and integrity authentication.

Hash functions generate a condensed representation of the input and are designed to make it impractical to find two different messages that produce the same hash value. These are used to provide source, integrity authentication, digital signature generation, key derivation and establishment.

Random bit generators provide a stream of random bits essential for generating keys.

Cryptographic keys and their types

Cryptographic keys are essential for converting plaintext data into ciphertext data and vice-versa. The cryptographic system utilizes two main types of keys: symmetric and asymmetric.

Asymmetric keys are a set of public-private key pairs where data is encrypted with one key and can only be decrypted by the other key.

The symmetric key is one secret key used for both encryption and decryption of the data.

These keys can also be categorized as either static or ephemeral keys based on their validity.

Additionally, there is a usage-specific classification that names keys according to their usage, making it easier to implement the principle of 'one, key one use.'

Authentication

- **Private/public signature key:** The private key of an asymmetric-key pair is utilized to generate digital signatures for long-term use, while the public key is employed to verify digital signatures.
- **Symmetric authentication key:** A symmetric key used for authentication and encryption through authenticated encryption modes of operation [1].
- **Private/public authentication key:** An asymmetric-key pair used for identity authentication during the establishment of an authenticated communication session or authorizing specific actions.

Encryption

- **Symmetric data-encryption key:** A single symmetric key employed for both source authentication and encryption.

Key wrapping

- **Symmetric key-wrapping key:** A key-encrypting key used to encrypt/decrypt other keys.

Random number generation

- **Symmetric random number generation keys:** A key used to generate random numbers or bits.

Key derivation

- **Symmetric master key/key-derivation key:** A key used to derive other symmetric keys (e.g., data-encryption keys or key-wrapping keys). This key is also known as a key-derivation key.

Transportation

- **Private/public key-transport key:** An asymmetric-key pair used to decrypt and establish symmetric keys, such as key-wrapping keys, data-encryption keys or MAC keys. It may also encrypt other keying material, such as initialization vectors. While integrity protection is also provided, it is not the primary intention of this key.

Key agreement

- **Symmetric key-agreement key:** Symmetric keys used to establish other symmetric keys like key-wrapping keys, data-encryption keys or MAC keys as well as optionally encrypting other keying material (e.g., Initialization Vectors) through symmetric key-agreement algorithm.
- **Private/Public static key-agreement key:** Long-term keys from asymmetric key pairs used to establish symmetric keys, including key-wrapping keys, data-encryption keys or MAC keys and optionally, other keying material (e.g., Initialization Vectors).
- **Private/Public ephemeral key-agreement key:** Short-term private keys from asymmetric-key (public-key) pairs that are used only once to establish one or more symmetric keys (e.g., key-wrapping keys, data-encryption keys or MAC keys) and optionally, other keying material (e.g., Initialization Vectors).

Authorization

- **Symmetric authorization key:** A key used to grant privileges to an entity. The authorization key to both entities is responsible for monitoring and granting access privileges and by the entity seeking access to resources.
- **Private/Public authorization key:** An asymmetric-key pair is used to verify the owner's right to privileges, typically through the use of a digital signature.

Key metadata trusted associations and bindings

Each key is associated with specific data that pertains to its identity, capability and functionalities. Here is a brief overview of these key qualifiers:

- Key metadata is associated with key information that provides context about cryptographic keys including attributes such as key type, owner, key creation date, key expiry date and the algorithm used.
- Trusted associations define the entities authorized to use the keys for encryption and decryption, helping to prevent unauthorized access.
- Bindings are used to link keys with specific data, ensuring that the key is required to manipulate or access the data.

To illustrate the concept of metadata, associations and bindings, consider an SSL certificate. The data associated with the certificate – such as the domain name, creation date, key ID, etc. – represents the key metadata. The association between CA and the issued certificate constitutes the trusted association, while the binding of SSL public keys to domain names facilitates secure communication.

The interdependence between these concepts ensures that cryptographic keys are used securely and in compliance with established policies and access controls.

Key management lifecycle

The key management lifecycle is all about managing different states of key and their transition throughout the product deployment lifecycle. Here, the lifecycle is explained with respect to NIST standard SP-800-57 [2].

Key states

A key traverses through various states throughout its lifecycle. The figure below explains the key state transition between various states –

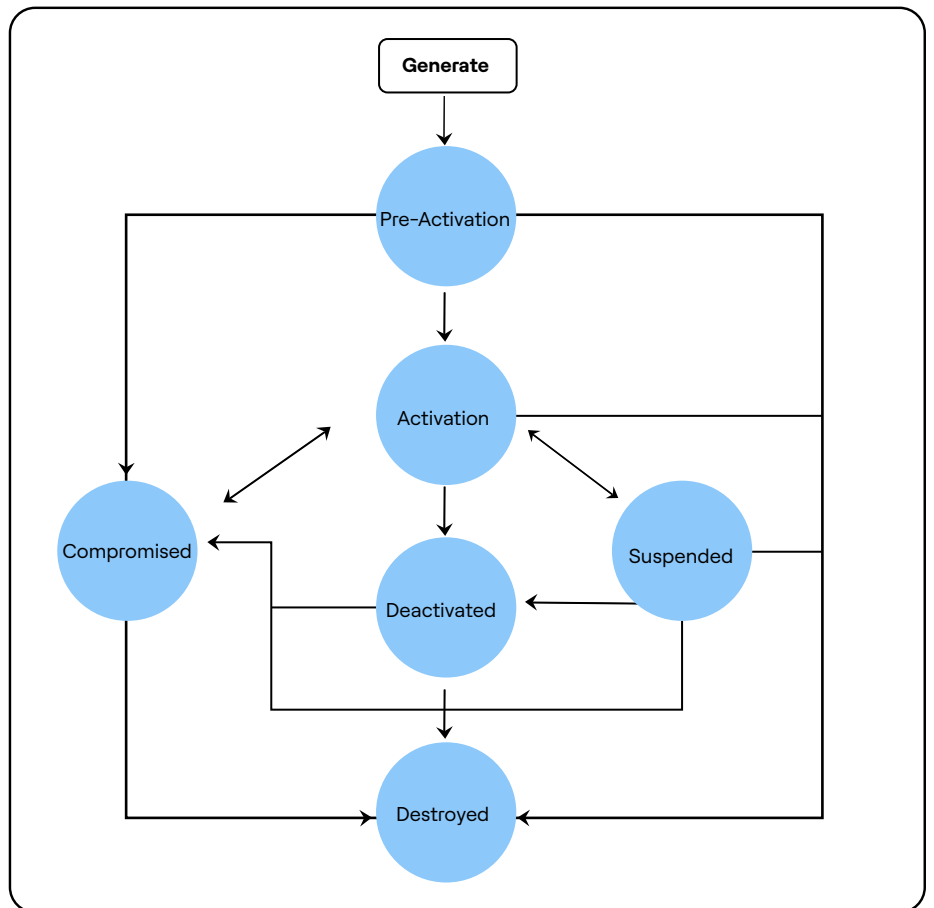


Figure 1 Key state transition diagram

Pre-activation state

The key enters the pre-activation state immediately after its generation. In this state, it cannot be used for encryption or decryption, but can be used for proof of possession or key confirmation.

If the key is exposed or no longer needed, it may transition to a compromised or destroyed state. If the key is valid and available, it may move to an active state and commence its cryptoperiod.

Activation state

In this state, the key is utilized for cryptographic operations such as encryption and decryption, effectively securing and processing the data securely. However, an active state key may transition to a compromised, suspended or deactivated state if it is leaked, remains unused for an extended period or has expired.

Suspended state

A key may enter a suspended state if it is suspected of being compromised or if it has not been used for a prolonged period. In the case of a suspected compromise, the keys are neither used for cryptographic protection (encryption/decryption) nor for any verification. Conversely, if a key is suspended due to inactivity, it may still be used for verification purposes.

A suspended key or key pair may transition to an active, deactivated or destroyed or compromised state, based on the reason for suspension.

Deactivated state

In this state, a key may be used for cryptographic verification but not for protection. If the reason for deactivation is a compromise, the key will not be used for verification either.

If a key compromise is proved, the key may transit to a compromised state. Otherwise, it may be moved to a destroyed state

Compromised state

A key is classified as being in a compromised state when it has been disclosed or determined by an unauthorized entity. In this case, the key cannot be used for cryptographic verification and/or protection. However, in rare cases, these keys may be allowed for verification if the verification timestamp is earlier than the compromise timestamp. Once the key's cryptoperiod expires, it will transition to the destroyed state.

Destroyed state

Here, the key is permanently unusable for any purpose. However, some metadata, such as key state transition history, key name, type and cryptoperiod will be retained for audit purposes.

Key management phases

Key management phases align with the product lifecycle phase, comprising pre-operational, operational, post-operational and destroyed phases. These phases ensure that cryptographic keys are generated, used and disposed of securely throughout the product lifecycle.

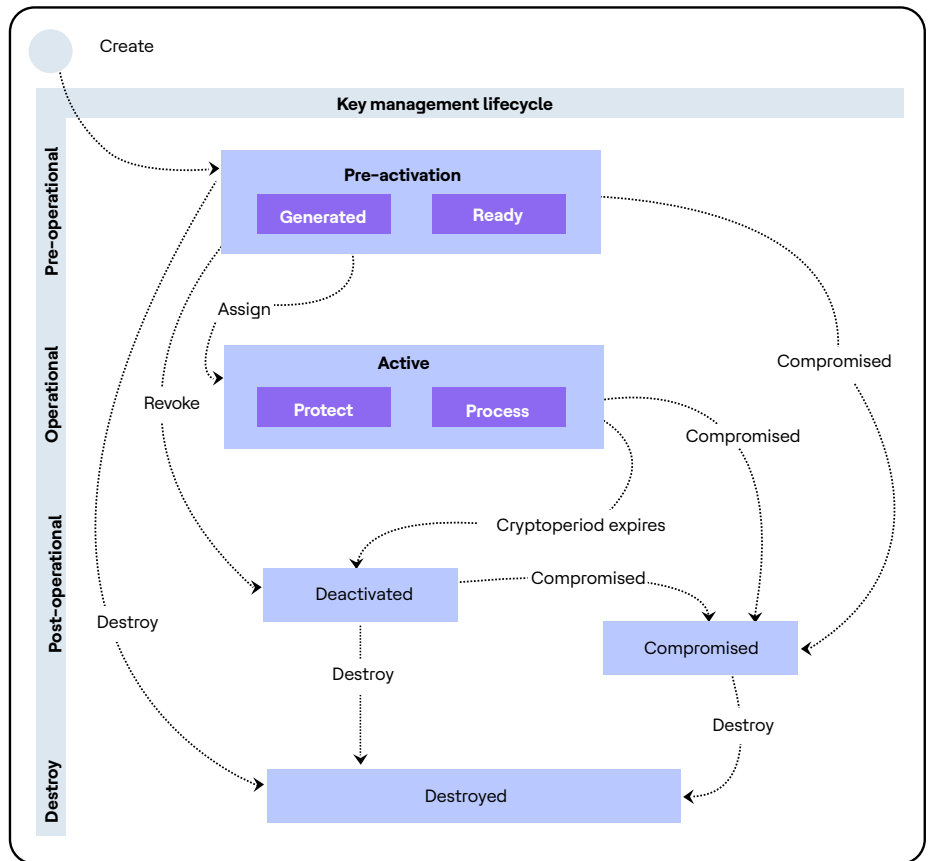


Figure 2 Key lifecycle with states and phases

Pre-operational phase

This phase focuses on the secure generation, distribution and provisioning of keys. During this phase, keys can be used for verification purposes but not for data protection. The following are the tasks that were performed during this phase. Please note this is the key generation phase. Hence, the key is not available for cryptographic protection.

Key generation

Keys are generated using a random bit generator, a KDF function or Hardware Security Module (HSM). It is essential that the keys are generated with sufficient entropy and randomness to withstand potential attacks.

Key distribution

Keys must be distributed securely to the system through a secure channel and protocol. Some distribution methods rely on verification keys that are permanently fused onto the device.

Key storage

Keys should be stored securely using trusted modules like TPM, OPTEE, etc.

Operational Phase

In this phase, keys are actively used to perform all cryptographic operations. During operational activities, keys may undergo processes such as rotation or revocation. While keys are used for cryptographic operations – such as encryption, decryption, signing or authentication – access control and storage considerations have already been handled in the pre-operational phase.

Key rotation

Key rotation ensures that keys are updated periodically to minimize the impact of compromised keys. This process involves generating new keys, updating the system with these new keys and securely disposing of the old keys.

Key backup and recovery

Procedures for key backup are established to ensure critical keys can be restored in the event of data loss or hardware failures.

Key usage monitoring

This involves monitoring and logging key usage to track any abnormal/suspicious activities related to cryptographic keys in case of a breach.

Post-operational phase

Once the key is no longer in use, the key management process transitions to archiving and de-provisioning the key. In this process, keys are archived to comply with regulatory requirements and facilitate potential forensic analysis in the event of security incidents.

Key de-provisioning

Once the key is not in use, it shall be marked as deactivated and the corresponding cryptographic functions should be updated to use the new key.

Key destruction

Secure destruction methods are used to destroy the keys, which includes cryptographically secure erasure or physical destruction of the storage media.

Documentation

Proper documentation of key destruction processes is essential for audit and compliance.

Key management standards and guidelines

Standards and guidelines offer numerous benefits for security architects designing embedded systems security. These benefits include enhanced security, interoperability, long-term resilience, scalability and consistency in the solutions provided by various organizations. Additionally, these standards contribute to community knowledge, helping both organizations and individuals comprehend and implement the complexities of key management while effectively safeguarding data. Here are some notable organizations that offer standards in key management areas:

National Institute of Standards and Technology (NIST)

NIST SP 800-57: Provides guidelines for cryptographic key management, including key generation, storage, distribution and decomposition. It is provided in three parts.

- A comprehensive guideline for cryptographic key management in information security
- Specific recommendations for effective key management within organizations
- Application-Specific Key Management Guidance tailored for specific applications (PKI, TLS, S/MIME etc.) and scenarios in cybersecurity

NIST SP 800-130: Offers a comprehensive guide to the framework consisting of policies, procedures, components and devices used to protect, manage and distribute cryptographic keys and associated metadata.

International Organization for Standardization (ISO)

ISO 27001: Annex A.10 of the ISO/IEC 27000 family of standards outlines key management requirements for information security management systems (ISMS) and policy on the use of cryptography controls.

ISO 11770: ISO/IEC 11770-2:2008 specifies a series of mechanisms for establishing shared secret keys using symmetric cryptography. Here, the focus is on key management principles and practices, including key generation, distribution and archival.

Internet Engineering Task Force (IETF)

RFC 4046: Multicast Security (MSEC) key management protocols to support a variety of application, transport and network layer security protocols.

RFC 4107: Guidelines to determine whether a given security system requires some form of automated key management or manual keying is sufficient.

RFC 4306: Describes Internet Key Exchange (IKE) version 2 protocol. IKE is used for performing mutual authentication and establishing and maintaining security associations (SAs) in IPsec.

RFC 4962: Provides architectural guidance to designers of AAA key management protocols

RFC 5275: Describes a mechanism to manage (i.e., set up, distribute and rekey) keys used with symmetric cryptographic algorithms.

Payment Card Industry Security Standards Council (PCI SSC)

PCI DSS: Sections 3.6 and 3.7 define the process for safeguarding cryptographic keys and implement processes for cryptographic key management.

GlobalPlatform

KMS Requirements Specifications v1.0: Defines the standards for the exchange and control of cryptographic keys between systems that interact with smart cards or generate keys for smart cards.

European Telecommunications Standards Institute (ETSI)

ETS 300 841: Describes authentication and key management methods for a privacy system used for narrowband audiovisual services.

Cloud Security Alliance (CSA)

CSA Cloud Control Matrix (CCM): Provides guidelines for cloud service providers (CSPs) and cloud customers, including key management practices in cloud environments.

Key management lifecycle best practices: Guideline to manage cryptographic keys effectively and securely throughout the lifecycle to protect digital assets and maintain regulatory compliance.

Health Information Trust Alliance (HITRUST)

HITRUST CSF: Addresses key management practices for securing healthcare information and systems.

Open Web Application Security Project (OWASP)

Key Management Cheat Sheet: Provides guidance for implementation of cryptographic key in terms of key lifecycle management, key storage, key agreement and key recovery.

Common Criteria (CC)

Common Criteria is an international standard (ISO/IEC 15408) for evaluating and certifying the security of IT products and systems, including cryptographic modules and key management.

TCG Group

TCG Storage Specifications and Key Management: defines the process of locally managing the authentication keys for SEDs (Self Encrypting Drives) by exploiting the standard interface defined in the TCG Storage.

OASIS Group

KMIP: Define a protocol for communication between encryption systems and enterprise applications, including email, databases and storage devices.

These organizations and standards provide valuable guidance and requirements for organizations across various industries to establish robust and secure key management practices, ensuring the confidentiality and integrity of sensitive data and communications.

Key management best practices

Most of the standards in key management areas cover enterprise applications. They do not consider the constraints of embedded devices, though NIST has provided an elaborate list of practices to be followed for key management. Shown below are several of the standards we reviewed to list best practices for key management in embedded systems.

Key management area	Standards	Description
Key management guidelines	SP 800-57 Part 1 Revision 5 - General	A comprehensive guideline for cryptographic key management in information security.
	SP 800-57 Part 2, Revision 1, Best Practices for Key Management Organizations	Specific recommendations for effective key management within organizations.
	SP 800-57 Part 3, Application-Specific Key Management Guidance	Application-Specific Key Management Guidance tailored to specific applications (PKI, TLS, S/MIME, etc.) and scenarios in cybersecurity.
Key management transitions	SP 800-131A Revision 2, Transitioning the Use of Cryptographic Algorithms and Key Length SP 800-71, Key	Provides guidance to transition to more strong keys and secure algorithms while safeguarding sensitive data. Provides security
Key establishment	Establishment Using Symmetric Block Ciphers (DRAFT)	considerations for the use of symmetric-key algorithms for key establishment.
	SP 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography	Provides guidance for key establishment using discrete logarithmic (ECC) cryptography.
	SP 800-56B Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography	Provides guidance for key establishment using integer factorization (RSA) cryptography.
	SP 800-56C Revision 2 Recommendation for Key Derivation Methods in Key-Establishment Schemes	Guidance on key derivation method using a shared secret.
	NIST Special Publication 800-107 Revision 1 Recommendation for Applications Using Approved Hash Algorithms	Recommendation for Applications Using Approved Hash Algorithms.

Key generation	SP 800-133 Revision 2, Recommendation for Cryptographic Key Generation (June 2020)	Guidance on key generation methods.
	SP 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping	Guidance on approved methods for authenticated encryption and authenticated decryption.
	SP 800-135, Transitions: Recommendation for Existing Application-Specific Key Derivation Functions	Guidance on security requirements for existing application-specific key derivation functions, for example, TLS, PKI and more.
Cryptographic Key Management Systems (CKMS)	SP 800-130, A Framework for Designing Cryptographic Key Management Systems	Framework consisting of policies, procedures, components and devices used to protect, manage and distribute cryptographic keys and associated metadata.
	NIST Special Publication 800-175B Revision 1, Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms	Guideline for Using Cryptographic Standards in the Federal Government Cryptographic Mechanisms.

While these standards outline expectations from a key management system, HCLTech's Cybersecurity team has identified several best practices for embedded systems.

Embedded systems best practices for key management

The following best practices, curated by HCLTech's Product Security team, provide a robust foundation for securing embedded devices, protecting sensitive data and mitigating risks associated with key management in an increasingly interconnected world.

Secure boot process

Implement a secure boot process to ensure that only trusted firmware and software can run on the device. This approach helps protect the cryptographic keys from unauthorized access or tampering.

Strong encryption algorithms

Use approved, strong encryption algorithms for key storage and communication with lower memory and computation requirements, such as ECDSA and ECC algorithms. Refer to FIPS 140-3 for a list of approved algorithms.

Key hierarchies

Establish a hierarchical key management system to minimize the exposure of high-value keys and enable key rotation. Root keys should be used to derive and manage other keys. Different keys should be designated for specific purposes (e.g., encryption, authentication, signing) to reduce the impact of a compromise on other security aspects.

Secure key storage

Store cryptographic keys securely within Hardware Security Modules (HSMs), Trusted Execution Environments (TEEs) or secure elements. If sufficient storage is unavailable, keys can be wrapped and stored in the mass storage.

Key rotation

Define and follow a policy for key rotation. Regularly rotating keys helps reduce risks associated with long-term key exposure, allowing for prompt rotation as soon as a compromise is detected.

Use randomized keys

The security of the keys relies on the secrecy and randomness of the keying material. Generate keys using a strong entropy-based hardware random number generator and avoid using predictable or hardcoded keys.

Authentication and authorization

Implement strong authentication mechanisms to ensure that only authorized users or applications can access and use keys. Role-based access control can further restrict access, allowing users to use keys for a limited duration.

Secure key transport

Use secure channels like TLS/SSL, for key exchange and data transmission. Ensure that the keys are securely transported to the device during provisioning, avoiding insecure channels for exchanges.

Backup and recovery

Implement key backup and recovery processes to safeguard device against failure or loss. These processes maintain the same level of security as the key storage itself.

Logging and auditing

Any access to key or key management operations should be logged and regularly reviewed for suspicious activities.

Firmware updates

Ensure that firmware updates are securely delivered and installed without compromising the security of stored keys.

Secure key transport

Design security protocols in accordance with established security standards and best practices such as Federal Information Processing Standards (FIPS) for cryptographic modules, as well as industry-specific security guidelines such as HIPAA, FDPR, PCI-DSS and others.

Testing and verification

Periodically conduct security testing, including penetration testing and vulnerability assessments, to identify and mitigate potential weaknesses in the key management system, especially following major updates or feature enhancements.

Documentation and training

Thoroughly document key management procedures and provide training for developers and administrators. All personnel involved in the key management process should understand their responsibilities and adhere to the guidelines relevant to their roles.

Incident response plan

Develop an comprehensive incident response plan to address security breaches or key compromises. This plan should outline procedures for key revocation and recovery and may include out-of-bound communication to inform stakeholders of key compromises.

Regular auditing

Regularly audit the key management system to ensure it is functioning correctly and securely. Implement monitoring and alerting to detect anomalies.

Continual improvement

Key management is an ongoing process. Continually assess and improve the security of key management practices to adapt to evolving threats and technologies.

Case study: Secure Boot

One of the primary applications of key management is Secure Boot, a critical embedded systems use case that highlights the importance of key management, key distribution and provisioning. Here is the description of the Secure Boot process and its significance in ensuring secure key management.

The process

Secure Boot guarantees that an embedded system starts up only with trusted firmware. It leverages digital signatures to authenticate the source and integrity of the code being loaded. This mechanism effectively prevents the execution or loading of malicious code and can detect or prevent attacks that involves rootkits.

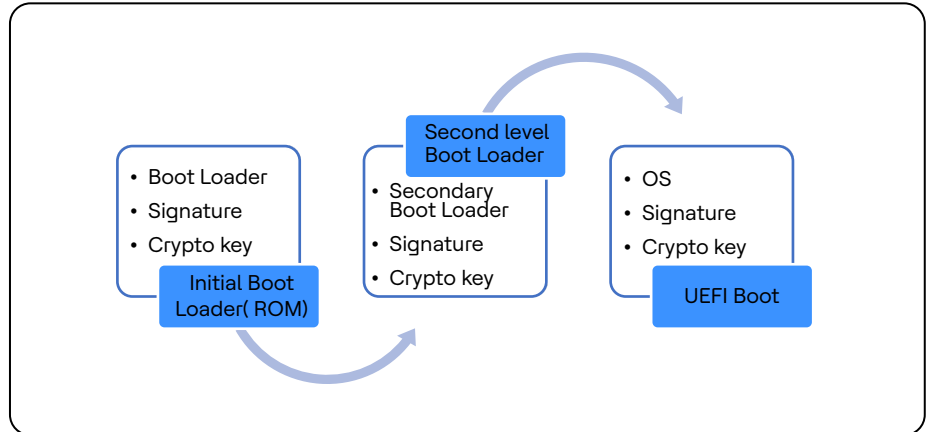


Figure 3 Secure Boot process

Used cryptographic keys

Secure Boot utilizes cryptographic keys to verify the bootloader and firmware. The following keys are integral to this process:

Platform Key (PK)

This is a public key embedded in the device's firmware by the manufacturer, which serves as the root of trust and is stored in immutable memory such as fuses.

Key Exchange Key (KEK)

This is also public key responsible for verifying the initial boot loader and the signature database. It establishes the root of trust through PK.

Signature database

It contains a collection of public keys used to verify the authenticity and integrity of operating systems, as well as other firmware and software components loaded during the Secure Boot process.

Role in key management

Key provisioning

The Secure Boot processes involve distributing public keys and certificates necessary for signature verification. These keys are securely provisioned to devices through secure channels, ensuring that only software components signed by authorized entities are allowed to execute.

Chain of trust

The Secure Boot process establishes a chain of trust through PK and KEK. The PK verifies the KEK, which, subsequently verifies the bootloader and kernel images.

Authentication

Secure boot ensured the integrity and authenticity of installed firmware components by verifying the digital signatures.

Key generation

PK and KEK are used to generate various application-specific keys and storage keys using KDF functions.

In summary, Secure Boot processes depend on cryptographic keys to verify software authenticity, preventing unauthorized code execution. Proper key distribution and management ensure that the correct keys are used for validating software components. This enables establishing a secure chain of trust from the root of trust to the operating system.

Conclusion

This paper highlights the critical role of key management in safeguarding system data. It outlines how cryptographic keys contribute to achieving confidentiality, integrity, authentication and data protection, emphasizing the importance of securing these keys throughout various stages of the system lifecycle. Effective key management is essential for compliance with regulations such as GDPR, HIPAA and PCI DSS, which apply across multiple sectors.

The National Institute of Standards and Technology (NIST) has played a significant role in establishing the standards and guidelines for key management practices, particularly in cybersecurity. In developing these standards, NIST ensures interoperability across device applications and industries while also addressing the long-term viability of cryptographic algorithms and practices in the face of evolving threats.

This paper also presents best practices for key management in embedded devices and includes a case study of Secure Boot.

In summary, key management serves as the backbone of embedded device security. Neglecting key management can expose devices and the data they process to significant risks. Therefore, prioritizing key management practices, such as key generation, storage, distribution, rotation and revocation, is essential for strengthening the security posture of embedded devices and ensuring their resilience against evolving threats.

References

1. Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms
2. Recommendations for Key Management-Parts 1,2 &3
3. <https://csrc.nist.gov/Projects/Key-Management/Key-Management-Guidelines>
4. <https://datatracker.ietf.org/doc/html/rfc4046>
5. <https://datatracker.ietf.org/doc/html/rfc4071>
6. <https://datatracker.ietf.org/doc/html/rfc4962>
7. <https://datatracker.ietf.org/doc/html/rfc4306>
8. <https://datatracker.ietf.org/doc/html/rfc4107>
9. <https://datatracker.ietf.org/doc/html/rfc5275>
10. <https://iso-docs.com/blogs/iso-27001-standard/iso-27001-annex-a-10-cryptography>
11. <https://www.iso.org/standard/46370.html>
12. <https://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=39d0c648-0a66-4f46-b343-018dc7d3f19c>
13. <https://trustedcomputinggroup.org/resource/tcg-storage-specifications-and-key-management/>
14. https://cheatsheetseries.owasp.org/cheatsheets/Key_Management_Cheat_Sheet.html
15. <https://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=39d0c648-0a66-4f46-b343-018dc7d3f19c>
16. <https://trustedcomputinggroup.org/resource/tcg-storage-specifications-and-key-management/>
17. [https://ldapwiki.com/wiki/Wiki.jsp?page=Ephemeral%20Key#:~:text=Public%20Ephemeral%20Key%20agreement%20keys,\(e.g.%2C%20Initialization%20Vectors\).](https://ldapwiki.com/wiki/Wiki.jsp?page=Ephemeral%20Key#:~:text=Public%20Ephemeral%20Key%20agreement%20keys,(e.g.%2C%20Initialization%20Vectors).)

Author info

Shivani Aggarwal



Shivani is a Solution Architect and Embedded Security evangelist working on security architecture in IOT & Embedded products. She has wide experience in Threat Assessment & Modeling, security controls implementations and Risk management. She has more than 18 years of experience in complete SDLC Implementation including Requirement gathering & Risk Assessment, System Design and Threat Modeling, development of secure code, testing and secure deployment. She has worked in various domain including consumer electronics, Medical, Automotive Industrial etc. She is experienced in developing secure solutions using PKI Infrastructure, Secure boot, Secure Execution environment, OS hardening, Network and Communication Security.

HCLTech | Supercharging Progress™

HCLTech is a global technology company, home to 222,000+ people across 60 countries, delivering industry-leading capabilities centered around Digital, Engineering and Cloud powered by a broad portfolio of technology services and software. The company generated consolidated revenues of \$12.3 billion over the 12 months ended December 2022. To learn how we can supercharge progress for you, visit hcltech.com.

hcltech.com

