2.2 Keys, Metadata, Trusted Associations, and Bindings

Khóa phải được liên kết với siêu dữ liệu chỉ định các đặc điểm, ràng buộc, cách sử dụng được chấp nhận và các thông số áp dụng cho khóa. Ví dụ: một khóa có thể được liên kết với siêu dữ liệu chỉ định loại khóa, cách tạo khóa, thời điểm tạo, số nhận dạng của chủ sở hữu, thuật toán dự định sử dụng và chu kỳ mã hóa của khóa. Mỗi đơn vị siêu dữ liệu được gọi là một phần tử siêu dữ liệu. Giống như các khóa, siêu dữ liệu cần được bảo vệ khỏi sự sửa đổi trái phép và có thể cần được bảo vệ khỏi bị tiết lộ; siêu dữ liệu cũng cần được xác thực đầy đủ nguồn của nó.

Phần tử siêu dữ liệu có thể được biết đến một cách ngầm định và do đó có thể không được ghi lại cụ thể cho một số khóa nhất định trong CKMS. Ví dụ: nếu tất cả các khóa trong một thiết bị là khóa AES-128, thì có thể không cần phần tử siêu dữ liệu ghi kích thước khóa. Tuy nhiên, trong nhiều hệ thống, cần phải phân biệt khóa này với khóa khác bằng cách sử dụng một hoặc nhiều phần tử siêu dữ liệu được ghi lại rõ ràng. Khung CKMS này tập trung vào các yếu tố siêu dữ liệu được ghi lại và quản lý rõ ràng bởi CKMS. Thuật ngữ “siêu dữ liệu” được sử dụng trong ngữ cảnh này (nghĩa là thuật ngữ “siêu dữ liệu” đề cập đến các phần tử siêu dữ liệu được ghi lại và quản lý một cách rõ ràng). Có thể có nhiều phần tử siêu dữ liệu cho một khóa nhất định. CKMS thường cần một liên kết đáng tin cậy, giữa một khóa và các phần tử siêu dữ liệu đã chọn, để thực hiện các chức năng quản lý chính. Ví dụ: bạn nên có một liên kết đáng tin cậy giữa khóa công khai tĩnh và số nhận dạng của chủ sở hữu. Khi được sử dụng cùng với quy trình đăng ký chủ sở hữu, liên kết đáng tin cậy cung cấp sự đảm bảo rằng chủ sở hữu được chỉ định bởi số nhận dạng đang hoặc đã sở hữu khóa cá nhân tương ứng. Các phần tử siêu dữ liệu có thể được tạo bởi cùng một thực thể tạo khóa hoặc chúng có thể được nhận từ một thực thể đáng tin cậy. Bất cứ khi nào siêu dữ liệu được nhận từ một thực thể đáng tin cậy (cho dù khóa được liên kết có được gửi đồng thời hay không) thì phải có sự liên kết đáng tin cậy giữa siêu dữ liệu và khóa được liên kết. Liên kết đáng tin cậy được duy trì trong quá trình phân phối có thể được thực thi bởi một ràng buộc mật mã (ràng buộc) của khóa và siêu dữ liệu (ví dụ: chữ ký điện tử được tính toán trên sự kết hợp của khóa và siêu dữ liệu) hoặc liên kết có thể được thực thi bởi một quy trình đáng tin cậy ( ví dụ: chuyển giao trực tiếp siêu dữ liệu từ một thực thể được biết đến và đáng tin cậy). CKMS thường cung cấp các chức năng xác minh và ràng buộc mật mã được sử dụng trong quá trình quản lý và phân phối khóa và siêu dữ liệu. Người nhận có được sự đảm bảo rằng khóa và siêu dữ liệu của nó được liên kết đúng cách, đến từ một nguồn cụ thể, không bị sửa đổi và được bảo vệ khỏi bị tiết lộ trái phép trong quá trình truyền tải. Sau khi nhận được siêu dữ liệu, mối liên kết giữa khóa và siêu dữ liệu phải được xác minh. Ràng buộc mật mã được xác minh bằng cách áp dụng chức năng xác minh mật mã thích hợp cho các phần tử siêu dữ liệu khóa và liên kết. Một liên kết đáng tin cậy không mật mã được xác minh bằng cách đánh giá quy trình đáng tin cậy (tức là sự tin cậy trong thực thể gửi và quy trình phân phối).

Diagram

Description automatically generated

2.4 Framework Topics and Requirements

2.5 CKMS Design

Mục đích của thiết kế CKMS là mô tả cách một hệ thống có thể được xây dựng để cung cấp các khóa mật mã cho các thực thể sẽ sử dụng các khóa đó để bảo vệ dữ liệu nhạy cảm. Mô tả cấp cao của CKMS phải chỉ ra cách sử dụng của từng loại khóa, vị trí và cách các khóa được tạo ra, cách chúng được bảo vệ trong lưu trữ tại mỗi thực thể nơi chúng cư trú và trong quá trình phân phối, và các loại thực thể mà chúng được chuyển đến.

Thiết kế CKMS phải chỉ rõ tổng quan cấp cao về hệ thống CKMS mà

bao gồm:

a) Việc sử dụng từng loại khóa,

b) Các khóa được tạo ra ở đâu và như thế nào,

c) Các phần tử siêu dữ liệu được sử dụng trong một liên kết đáng tin cậy với mỗi loại khóa,

d) Cách khóa và / hoặc siêu dữ liệu được bảo vệ trong bộ nhớ tại mỗi thực thể nơi chúng

cư trú tại,

e) Cách các khóa và / hoặc siêu dữ liệu được bảo vệ trong quá trình phân phối, và

f) Các loại thực thể mà khóa và / hoặc siêu dữ liệu có thể được phân phối (ví dụ: người dùng, thiết bị người dùng, thiết bị mạng)

6.2 Key Metadata

6.2.1 Metadata Elements

1. Key Label: A key label is a text string that provides a human-readable, and perhaps machine-readable, set of descriptors for the key. Examples of key labels
2. Key Identifier: This element is used by the CKMS to select a specific key from a collection of keys. A key identifier is generally unique in a security domain. For public and private keys, a key identifier can be a hash value or portion of the hash value of the public key or can be assigned by the CKMS.
3. Owner Identifier: This element specifies the identifier (or identifiers) of the entity (or entities) that owns (or own) the key.
4. Key Lifecycle State: A key lifecycle state is one of a set of finite states that describe the current permitted conditions of a cryptographic key (see Section 6.3).
5. Key Format Specifier: This element is used to specify the format for the key. This can be accomplished by reference to the structure using object identifiers. For example, an RSA public key consists of the modulus and a public exponent. The format specifier should specify the sequence in which these two values are stored and the format in which each value is encoded. The Internet Engineering Task Force (IETF) has defined an object identifier for storing different forms of public keys, such as DSA, DH, RSA, EC, RSAPSS, and RSAOAEP keys. The object identifiers and related public key structures are defined in the following Internet RFCs: [RFC 3279], [RFC 4055], and [RFC 5480].
6. Product used to create the Key: This element specifies which cryptographic product was used to create or generate the key.
7. Cryptographic Algorithm using the Key: This element specifies the cryptographic algorithm that is intended to use the key. Examples include DSA, ECDSA, RSA, AES, TDEA, and HMAC-SHA1.
8. ) Schemes or Modes of Operation: This element defines the applicable schemes or modes of operation for performing a cryptographic function using a key. For asymmetric algorithms, it may specify the operation of discrete logarithm algorithms in a mathematical finite field, binary field, or Elliptic Curve (EC) field. For symmetric algorithms, this field may define the mode(s) of operation that can be used by the block cipher algorithm when using the key. Examples of modes of operation are Electronic Code Book (ECB), Cipher Block Chaining (CBC), Output Feedback Mode (OFB), and Counter with Cipher Block ChainingMessage Authentication Mode (CCM). For more information, see [SP 800-38A] through [SP 800-38F].
9. Parameters for the Key: This element specifies the parameters, if applicable, for a key. For example, a DSA key has the following domain parameters: large prime (p), small prime (q), and generator (g).
10. Length of the Key: This element specifies the length of the key in bits (or bytes). Examples include 2048 bits for an RSA modulus, and 256 bits for an elliptic curve key
11. Security Strength of the Key/Algorithm Pair: This element is a number indicating the amount of work (that is, the base 2 logarithm of the number of operations) that is required to break (i.e., cryptanalyze) the cryptographic algorithm. For example, for a TDEA key of 168 bits (not including parity bits), the security strength is specified as 112 bits; for a 2048-bit RSA modulus, the security strength is specified as 112 bits. The security strength of a key/algorithm pair may be reduced if a previously unknown attack is discovered.
12. Key Type5 : This element identifies the key type. Key types were discussed in Section 6.1.
13. Appropriate Applications for the Key: This element specifies applications for which the key may be used. Examples include Kerberos, Signed E-Mail, Trusted Time Stamp, Code Signing, File Encryption, and IPSEC.
14. Key Security Policy Identifier: This element identifies the security policy applicable to the key or key type. A Key Security Policy is a set of security controls that are used to protect the key or key type during the lifecycle of the key from generation to destruction (see Section 6.7 and [RFC 3647]). A Key Security Policy is typically represented by an object identifier registered by the CKMS organization. The Key Security Policy for individual keys or key types is part of, and should be consistent with, the CKMS Security Policy.
15. Key Access Control List (ACL)6 : An access control list identifies the entities that can access and/or use the keys as constrained by the key and metadata management functions (see Section 6.7). This Framework does not specify the access control list structure. The following are examples of such structures: a Microsoft Windows file/folder access control list consisting of zero or more access control entries, a Sun File System access control list, and while not a list, the Unix protection bits. In cases where interoperability is desired, the following items may require standardization: the syntax and semantics of the separators among access control entries, the ordering of entity and “access modes” within an access control entry, the entity identifier, and the designation of bits for different “access modes”. If required for interoperability, these items should be included in an appropriately detailed design specification
16. Key Usage Count: This element indicates the number of times that the key has been used.
17. Parent Key: This element points to the key from which the key associated with this metadata is derived. For example, a new key (i.e., the child key) could have been derived from a TLS master secret (i.e., the parent key) with its metadata.

This element may have two sub-elements:

1. Key Identifier: The identifier for the parent key (see item b) above).
2. Nature of the Relationship: This element identifies how the parent key is related to the child key. An example of the relationship is a mathematical function that was used to create the child key using the parent key as one of the inputs. The relationship might be indicated by the identification of the mathematical function.
3. Key Sensitivity: This element specifies the sensitivity or importance of the key. It could relate to a risk level (e.g., Low, Moderate, or High) or a classification level (e.g., Confidential, Secret, or Top Secre
4. Key Protections7 : This element specifies the integrity, confidentiality, and source authentication protections applied to the key. A public key certificate is an example of key protection whereby the CA’s digital signature provides both the integrity protection and source authentication (see [X.509]). A symmetric key and its hash value encrypted together is an example of confidentiality and integrity protection. When a key and its metadata are received from an external entity, the protections should be verified before the key and metadata are operationally used. Generally, a single cryptographic function (e.g., HMAC or digital signature) is used to provide both integrity protection and source authentication. This element may have several sub-elements:
   1. i. The mechanism used for integrity protection (e.g., hash value, MAC, or digital signature),
   2. ii. The mechanism used for confidentiality protection (e.g., key wrapping or key transport),
   3. iii. The mechanism used for source authentication (e.g., MAC or digital signature), and
   4. iv. An indication of the protections that are enforced by a particular noncryptographic trusted process.