

## **Assignment 6.1: Metadata for Keys**

Joshua A. Wortz

University of San Diego

CSOL-510

Prof. Hoffman

06/15/2026

## **Metadata for Keys**

While cryptographic algorithms provide the mathematical foundation for data confidentiality and integrity, their effectiveness relies entirely on how the underlying cryptographic keys are generated, distributed, and protected throughout their lifecycle (Barker, 2020). A critical component of robust key management is the use of key metadata. Information about a cryptographic key that specifies its characteristics, acceptable uses, and applicable parameters must be tightly associated with the key (Barker et al., 2015). This information is called the key's metadata, and each descriptive item is a metadata element (Barker, 2020). This paper will identify a comprehensive list of standard key metadata attributes, recommend system and cryptographic hardening features to protect this data, and apply these concepts to a real-world web traffic scenario.

### **Comprehensive List of Cryptographic Key Metadata**

Although metadata does not alter the mathematical function of a cryptographic algorithm, a Key Management System (KMS) must manage, govern, and protect keys securely. Different applications and key types require distinct metadata elements to govern their secure use. Based on the authoritative guidelines in the National Institute of Standards and Technology (NIST) Special Publications 800-152 and 800-57 Part 1, Table 1 details a comprehensive list of standard metadata elements utilized across operating systems and applications. In practical deployment, these metadata elements are embedded across diverse operational layers. For example, user account authentication relies on owner and key-identifier metadata in SSH `authorized_keys` files to map cryptographic tokens to specific identities. At the operating system and application layers, infrastructure such as the Windows Certificate Store or web server

configurations explicitly ingest metadata on algorithm, key length, and usage constraints to determine how underlying services can use a key.

**Table 1**

*Comprehensive List of Cryptographic Key Metadata Elements*

| Category                           | Metadata Element                  | Description                                                                                                                   |
|------------------------------------|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <b>Identifiers &amp; Ownership</b> | <b>Key Identifier &amp; Label</b> | Unique string or label used to identify, track, and retrieve the key.                                                         |
|                                    | <b>Owner Identifier</b>           | The person, device, or organization explicitly authorized to own and use the key.                                             |
|                                    | <b>Sponsor / Representative</b>   | The identity of the human responsible for managing the key if the owner is a non-human device or process.                     |
|                                    | <b>Location</b>                   | The physical or logical location of the device or process using the key.                                                      |
|                                    | <b>Associated / Parent Keys</b>   | Identifies related keys, such as the corresponding public/private key in a pair, or the parent key from which it was derived. |
|                                    | <b>Source</b>                     | The entity that generated or provided the keying material.                                                                    |
| <b>Cryptographic Properties</b>    | <b>Algorithm &amp; Key Length</b> | The specific cryptographic algorithm (e.g., AES, RSA) and key size (e.g., 256-bit) to be used.                                |
|                                    | <b>Key Type</b>                   | Specifies the key's function, such as a signing private key, symmetric encryption key, or master key.                         |

|                              |                                  |                                                                                                                             |
|------------------------------|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
|                              | <b>Key Format Specifier</b>      | Identifies the specific data format of the key.                                                                             |
| <b>Lifecycle &amp; State</b> | <b>Key Lifecycle State</b>       | The current status of the key such as pre-activation, active, suspended, or destroyed.                                      |
|                              | <b>Cryptoperiod</b>              | The defined lifespan (start and end dates) during which the key is authorized for active use.                               |
|                              | <b>Event Dates</b>               | Timestamps recording critical lifecycle events, including activation, compromise, revocation, and future destruction dates. |
|                              | <b>Revocation Reason</b>         | The justification is recorded if a key is suspended or revoked early (e.g., suspected compromise).                          |
|                              | <b>Key Usage Count (Counter)</b> | A counter used to track the number of times a key has been used or to detect playback attacks.                              |
| <b>Usage Constraints</b>     | <b>Appropriate Applications</b>  | Explicitly lists the applications permitted (or forbidden) to use the key, such as TLS, IPSec, or SSH.                      |
|                              | <b>Key Sensitivity</b>           | Defines the sensitivity level of the information the key is authorized to protect.                                          |
|                              | <b>Access Control List (ACL)</b> | A list determining which users or entities have rights to access or manage the key.                                         |
|                              | <b>Key Security Policy</b>       | An identifier linking the key to the specific organizational security policy governing its use.                             |
| <b>Security Protections</b>  | <b>Confidentiality Mechanism</b> | The identifier of the key-wrapping key and the                                                                              |

|                            |                                                                                                                                                                   |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            | algorithm used to encrypt the primary key and its metadata.                                                                                                       |
| <b>Integrity Mechanism</b> | The Message Authentication Code (MAC) or Digital Signature algorithm used to cryptographically bind the metadata to the key, ensuring it cannot be tampered with. |

### Recommended Hardening Features and Best Practices

To enforce best practices in key management and protect the integrity of keys and their associated metadata, organizations must implement robust hardening measures at the system and cryptographic levels.

**System-Level Hardening:** Hardening is the process of eliminating avenues of attack by patching vulnerabilities and disabling nonessential services (Barker et al., 2015). Standard system hardening features include disabling network ports not strictly required for system operations, removing non-essential software to minimize the attack surface, and running continuous malware checks (Barker et al., 2015).

**Cryptographic and Metadata Hardening:** Because metadata defines how a key can be used, unauthorized modification can lead to system compromise. All metadata must be cryptographically protected with integrity mechanisms (such as MACs or digital signatures) before transmission or storage, thereby verifying that the metadata has not been maliciously altered (Barker, 2020; Barker et al., 2015). Furthermore, keys should be stored in tamper-resistant, FIPS 140-validated Hardware Security Modules (HSMs) or Trusted Platform Modules

(TPMs) to prevent the keys from ever existing in plaintext outside of a secure cryptographic boundary (Barker, 2020).

**Access and Governance Controls:** An access control system ensures that every key and metadata management function is initiated only in response to a request from a verified, authorized entity (Barker et al., 2015). Organizations should enforce strict Role-Based Access Control (RBAC), utilize automated key rotation to minimize human error, and maintain immutable audit logs that record all successful and failed attempts to access keys or modify metadata. Operationalizing these access controls requires adherence to the principle of least privilege and a strict separation of duties. To prevent insider threats and configuration tampering, the administrative roles responsible for managing key metadata (such as modifying Access Control Lists or updating cryptoperiod parameters) must be cryptographically and organizationally isolated from the operational roles or system service accounts that merely use the keys for active encryption and decryption.

### **Real-World Application**

**Web Traffic (HTTPS/SSL):** Modern web traffic over HTTPS does not rely on a single cryptographic algorithm but instead uses a hybrid model to balance security and performance. This process begins with asymmetric public-key algorithms, such as RSA or ECDSA, to verify digital certificates and authenticate the web server's identity (Mammeri, 2024). To establish the handshake, the systems use key-agreement schemes such as Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) to negotiate a shared symmetric session key (Barker, 2020). Once this shared secret is established, symmetric algorithms such as the Advanced Encryption Standard (AES) are used for fast, bulk encryption of the actual data payload transmitted during the session (Mammeri, 2024).

Behind the scenes of the Transport Layer Security (TLS) handshake, cryptographic metadata governs the entire execution of this hybrid model. Before the asymmetric public key can be trusted to establish a session, the client evaluates the server's public-key certificate. The client checks the key's cryptoperiod metadata to ensure the key is currently active and not expired (Barker, 2020). It also evaluates usage constraints, verifying that the key metadata explicitly permits its use for the TLS application (Barker et al., 2015). Finally, the system confirms the integrity of the metadata itself by verifying the Certificate Authority's digital signature, guaranteeing that the key and its attributes have not been tampered with. If any of these metadata checks fail, the TLS protocol terminates the connection to prevent a potential compromise.

As web traffic adapts to emerging threats, this hybrid TLS architecture is currently evolving to support post-quantum cryptography; future TLS handshakes will negotiate hybrid key exchanges that combine traditional algorithms with post-quantum Key Encapsulation Mechanisms (like ML-KEM) to protect session keys against quantum adversaries (ETSI, 2024; Moody et al., 2024).

## **Conclusion**

Even the most mathematically secure cryptographic algorithms will fail if the keys they rely upon are mismanaged or poorly protected (Barker, 2020). Metadata transforms a meaningless string of random bits into a highly governed, strictly controlled cryptographic asset by establishing its lifespan, ownership, and permitted applications (Barker et al., 2015). By pairing rigorous metadata tracking with critical hardening measures, such as turning off vulnerable network ports, using hardware security modules, and enforcing strict access controls,

organizations can ensure the long-term confidentiality, integrity, and availability of their sensitive data across both internal networks and real-world web traffic.

## References

Barker, E. (2020). *Recommendation for key management: Part 1, General* (NIST Special Publication 800-57 Part 1 Rev. 5). National Institute of Standards and Technology.

<https://doi.org/10.6028/NIST.SP.800-57pt1r5>

Barker, E., Smid, M., & Branstad, D. (2015). *A profile for U.S. federal cryptographic key management systems* (NIST Special Publication 800-152). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-152>

ETSI. (2024). CYBER Security (CYBER); Quantum-Safe Cryptography (QSC); Deployment Considerations for Hybrid Schemes (ETSI TR 103 966 V1.1.1).

Mammeri, Z. (2024). Chapter 14: Key management. In *Cryptography: Algorithms, protocols, and standards for computer security* (1st ed.). John Wiley & Sons.

Moody, D., Perlner, R., Regenscheid, A., Robinson, A., & Cooper, D. (2024). Transition to Post-Quantum Cryptography Standards (NIST IR 8547 ipd). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.IR.8547.ipd>