Storage Security: Encryption and Key Management

Version 2.0

September 5, 2023

Eric A. Hibbard, CISSP, FIP, CISA

Table of Contents

SNIA

List of Tables

List of Figures

Executive Summary

Encryption and key management capabilities have become pervasive within storage systems and ecosystems. To realize the full value of these capabilities, storage and security personnel need to understand these technologies, as they relate to storage, as well issues and considerations affecting their use. When implemented correctly, storage-based encryption and key management can serve as an important element of a defense in depth architecture.

Understanding the relevant cryptographic mechanisms and their uses in protecting data transferred to and stored in storage systems is critical to protecting sensitive and high-value data. This technical paper provides a solid foundation for storage and security professionals as it covers a broad range of relevant concepts and offers guidance while highlighting important issues and considerations.

1 Introduction

Storage security capabilities and practices have seen significant advances since their initial introductions. Storage systems (e.g. hard disk drives, solid state drives, storage arrays, and file servers) and storage ecosystems (e.g. storage devices and systems, storage networks, and storage management software) are able to protect data in a variety of ways, including the use of modern cryptography.

The application of encryption and accompanying key management to storage systems and storage ecosystems can afford very different protections depending on how and where it is applied. For example, encrypting files or shares within a Network Attached Storage (NAS) file system can offer user-specific protections that are not possible with encryption at the drive level if it stores files for multiple users. Thus, it is important to understand the threats to be mitigated when considering encryption.

This technical paper provides background information on encryption and key management, summarizes relevant security options, and offers additional guidance that can help in securing storage.

1.1 Confidentiality and Secrecy

Confidentiality is the property whereby information is available to authorized parties on demand, but never available to unauthorized parties lacking the relevant key(s). Secrecy is a term that is often used synonymously with confidentiality. Cryptographic mechanisms are one of the strongest ways to provide confidentiality and other security services in applications and protocols for data storage.[1]

Confidentiality is often achieved by using encryption to render the information unintelligible to unauthorized entities lacking the relevant key(s). The information is rendered intelligible to authorized entities by decryption. For encryption to provide confidentiality, the cryptographic algorithm must be designed and implemented so that an unauthorized party cannot determine the secret or private keys[2] associated with the encryption or be able to derive the plaintext directly without use of any keys.

---

[1] A general introduction to cryptography is *Cryptography Decrypted* by H. X. Mel and Doris Baker (Addison-Wesley:2000 ISBN 978-0201616477)

[2] Secret keys are used in symmetric encryption while private keys are used in asymmetric encryption.

1.2 Cryptographic Overview

The primary purpose of encryption (or *encipherment*) systems is to protect the confidentiality of stored or transmitted data. Encryption algorithms achieve this by transforming plaintext into ciphertext, from which it is computationally infeasible to find any information about the content of the plaintext unless the decryption key is also known.

It is important to note that encryption may not always, by itself, protect the integrity or the origin of data. In many cases it is possible, without knowledge of the key, to modify encrypted text with predictable effects on the recovered plaintext. In order to provide integrity and origin of data assurance it is often necessary to use additional techniques.

There are three basic classes of cryptographic algorithms:

- hash algorithms,
- symmetric key algorithms, and
- asymmetric key algorithms.

Ciphers require one or more keys and possibly other keying material (e.g. initialization vectors). In a symmetric cipher, the same key is used with both the encryption and decryption algorithms. In an asymmetric cipher, different but related keys are used for encryption and decryption (i.e. each key can decrypt text enciphered by the other). The management and protection of keys (known as key management) is critically important in maintaining data confidentiality.

1.2.1 Hashing

A hash algorithm (sometime referred to as a hash function) is used to map a message of arbitrary length to a fixed-length message digest. Cryptographic hash functions are a special class among hash functions that provide additional security assurances. The following are ideal cryptographic hash principles:

- Deterministic: the same inputs to the hash function always results in the same hash value.
- Quick: it is quick to compute the hash value for any given input.
- One-way function: it is infeasible to recover a message from its hash value except by trying all possible messages (i.e. in order to determine which message produced hash output *x*, one would have to search the universe of possible messages to find the correct message). The National Institute of Standards and Technology (NIST) refers to this as preimage resistance.
- Avalanche effect: a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value.
- Collision resistant: it should be rare that two independent messages produce the same hash value.
- Pre-image attack resistant: if a message produces hash value *x*, it should be very difficult to find another independent message that produces *x*. NIST refers to this as second preimage resistance.

Cryptographic hash functions are commonly used for:

- Digital Signatures – A hash function is used to map a message of any eligible length to a fixed-length message digest. In a digital signature generation process, this message digest is then signed by a signing operation, such as an RSA private key operation, to produce a digital signature.
- Keyed-Hash Message Authentication Codes (HMAC) – Message authentication codes (MACs) provide data authentication and integrity protection. Two types of algorithms for computing a MAC have been approved:
    - MAC algorithms that are based on approved block cipher algorithms, and
    - MAC algorithms that are based on hash functions, called HMAC algorithms.
- Hash-based Key Derivation Functions (KDFs) – Hash functions can be used as building blocks in key derivation functions.
- Random bit generator (RBG) – RBGs can be constructed using hash functions. The hash function used by the RBG needs to be selected so that the RBG can provide a security strength that meets or exceeds the minimum security strength required for the random bits that it generates.
- Truncated Message Digest – When a value that is shorter than the (full-length) message digest provided by an approved hash function is needed, a shortened (truncated) message digest can be generated when certain requirements are met.

Within storage systems and ecosystems, SHA-2 cryptographic hashes of lengths 256, 384, and 512 are the most prevalently used cryptographic hashes. In addition, HMACs of length 384 and 512 based on SHA-2 are commonly used. Finally, SHA-512 hashes truncated to a length of either 224 or 256 bits are sometimes used. Compared to SHA-2, SHA-3 is seen as a more secure, but more computationally challenging to calculate, alternative to it, rather than as a replacement for it. Usage of SHA-3 will likely become more commonly used in the future.

1.2.2 Symmetric Key Encryption

Symmetric encryption is a type of encryption where only one key (a secret key) is used to both encrypt plaintext data and decrypt ciphertext data. Any party that has a secret key can gain access to data that has been encrypted by that secret key.

Symmetric encryption is faster and more computationally efficient than asymmetric encryption, so it is typically used for bulk encryption (i.e. encrypting large amounts of data under real time constraints) such as when encrypting/decrypting data being read/written to storage devices. Industry-standard symmetric encryption is also less vulnerable to advances in quantum computing compared to the current standards for asymmetric algorithms (see 7.1).

1.2.2.1 Symmetric Key Algorithms

There are two types of symmetric encryption algorithms:

- Block algorithms. One fixed-length group of plaintext bits, called a block, is encrypted with the use of a specific secret key. To encrypt more than a block of data, a mode of operation, which describes how to repeatedly apply a cipher's single-block operation, is used to encrypt data larger than a block. The most commonly used symmetric key encryption algorithm in use today is the Advanced Encryption Standard (AES), also known as the Rijndael algorithm. AES is specified in

ISO/IEC 18033-3:2010[8] and the NIST Federal Information Processing Standards (FIPS) Publications 197 (FIPS PUBS 197)[22]. AES is a block algorithm with a 16-byte block size and supports three different key lengths: 128-bit, 192-bit, and 256-bit, though 192-bit keys are rarely used. For symmetric block cipher algorithms, there are multiple ways (called modes of operations) in which the cipher can be used to encrypt plaintext (see 3.1 for additional information on modes of operation).

- Stream algorithms. Each plaintext bit is encrypted one at a time with the corresponding bit of the keystream (pseudorandom cipher digit stream) to produce a bit of the ciphertext stream.

Symmetric ciphers are commonly used to achieve other cryptographic primitives (e.g. hashing, key wrapping, and pseudorandom number generation) than just encryption.

1.2.2.2 Symmetric Key Encapsulation / Wrapping

Symmetric key wrapping is a cryptographic method of encrypting a secret key using symmetric encryption. This is done to protect the confidentiality and integrity of the secret key, which can then be stored or transmitted securely. As an example, the secret key being wrapped may itself be a symmetric key such as a Media Encryption Key (MEK) which is used to do bulk encryption of user data. The secret key being wrapped could, alternately, be the private key part of an asymmetric key pair. The symmetric key used to perform the encryption is known as a Key Encrypting Key.

The two most commonly used symmetric key wrap algorithms in storage systems and ecosystems today are both AES based:

- AES-KW (AES Key Wrap) is a symmetric key wrapping algorithm that uses AES encryption in a particular iterative way, specified by NIST, that effectively adds an 8-byte Initialization Vector (IV) into the key wrapping performed by encryption with the Key Encrypting Key (KEK) to create a wrapped key output that is 8 bytes (more if the secret key being wrapped is not divisible by AES's block length, which is 16 bytes) longer than the secret key being wrapped. That AES-KW output can later be unwrapped by AES key unwrap (decryption) with that same KEK.
- GCM-AES (Galois Counter Mode usage of AES) is a mode of operation for symmetric encryption that provides both confidentiality and integrity protection, hereafter simply GCM. GCM is a relatively new mode of operation, but it has quickly become one of the most popular due to its strong security properties and performance. When GCM is used as a key wrapping technique, a nonce must be provided for use as GCM's Initialization Vector (or IV, typically 12 bytes in length); the nonce should never be reused with a given encryption key. That IV, the secret key to be encrypted, and the KEK are all inputs to the GCM encryption operation. The output of the GCM encryption includes the ciphertext and an authentication tag (which is 16 bytes in length). To decrypt the ciphertext, GCM decryption requires the IV, the ciphertext, the authentication tag in addition to the KEK. So, a GCM wrapped key output might typically be a structure 28 bytes (more if the secret key being wrapped is not divisible by AES's block length, which is 16 bytes) longer than the secret key which was wrapped.

Symmetric key wrapping with either AES-KW or GCM provides a strong security solution for protecting the confidentiality and integrity of the secret key which is wrapped. Some type of key management is required in that the KEK used to perform the symmetric key wrapping must be kept secure, as it can be used to decrypt any data that has been encrypted using that KEK. If symmetric key wrapping is used for secure key transport, then both ends of that transport need access to the same KEK, and so there is a

key distribution issue which has to be addressed. That is, either both ends must already have the same shared secret which can be used as the symmetric key wrap's KEK, or a shared secret has to be established.

1.2.3 Conventional Asymmetric Cryptography

Asymmetric cryptography makes use of two related keys (i.e. a key pair) for cryptographic operations. The two related keys in an asymmetric key pair are typically referred to individually as the public key and the private key. The public key can be shared publicly (often in certificate form), but the associated private key must be kept secret for any confidentiality or identity assurance to be achieved.

Because asymmetric encryption is typically much more computationally intensive than symmetric encryption it is not typically used to perform bulk encryption. The three dominant use cases of asymmetric cryptography in storage applications are associated with digital signatures, key agreement, and key encapsulation.

1.2.3.1 Conventional Asymmetric Cryptography Algorithms

The most common asymmetric cryptographic algorithms in use today in storage solutions are the conventional types:

- Rivest-Shamir-Adleman (RSA),
- Elliptic Curve Cryptography (ECC),
- Diffie-Hellman (DH),
- Elliptic Curve Diffie-Hellman (ECDH),
- Digital Signature Algorithm (DSA),
- Elliptic Curve Digital Signature Algorithm (ECDSA), and
- Edwards-curve Digital Signature Algorithm (EdDSA).

Some other types of conventional asymmetric cryptography (such as ElGamal) are rarely used in storage systems and ecosystems and will not be discussed further here.

It should be noted that the security of nearly all conventional asymmetric cryptographic algorithms (e.g. RSA, ECC, and DH) are under serious threat by the recent relatively rapid advancement in the capabilities of quantum computing and use of them may need to be deprecated or sunset in the relatively near future (e.g. by 2030). That is, all conventional asymmetric cryptographic algorithms will be effectively completely broken when a Cryptographically Relevant Quantum Computer (CRQC) becomes a reality. The only known exceptions to that last statement are Leighton-Micali signature (LMS) and eXtended Merkle Signature Scheme (XMSS) (to read a discussion about those algorithms, see 7.1.3), which are stateful hash-based digital signature algorithms which are thought to be quantum-safe, and so secure against even a CRQC. New Post-Quantum Cryptography (PQC) algorithms have been developed for use instead of, or in conjunction with, conventional asymmetric cryptographic algorithms (see 7.1).

1.2.3.2 Digital Signatures

Digital electronic signatures are created/generated by the signer (by use of his private key) and checked by the checker (by use of the corresponding public key) and are typically affixed to a document, file, or message. If the signer's private key has been kept secret, then a valid (i.e. one that is validated as correct) digital electronic signature can be used to establish both the authenticity (i.e. it was created and signed by a known sender) and integrity (i.e. it has not been corrupted) of that signed message. RSA, DSA,

ECDSA, and EdDSA are the four most commonly used conventional asymmetric digital signature algorithms in use in storage systems and ecosystems today.

Digital signatures can be calculated and appended to code loads for storage systems and to components of storage systems. These appended digital signatures can be checked as part of a storage systems boot sequence to achieve Trusted Boot or Secure Boot (e.g. by a Trusted Platform Module).

Digital signatures are also used to secure digital certificates. A typical digital certificate is an X.509 digital signature which is issued by a certificate authority (CA), or by a subordinate (to the CA) registration authority (RA). A digital certificate has the CA (or RA) attesting, by affixing its digital signature (in some cases to a chain of intermediate signatures) that a given public key (the certificate's Subject Public Key) is associated with the other parameters in the digital certificate (e.g. such as certificate's Subject Name).

Digital signatures are also used to secure messages sent across open (subject to interception) communication channels such as Ethernet connections used to administer a storage system. Key agreement protocols, such as Internet Key Exchange (IKE), use digital signatures to secure communication protocols such as Transport Layer Security (TLS), Secure Shell (SSH), IP Security (IPsec), and Fibre Channel - Security Protocols (FC-SP) to enable protocols endpoints to authenticate one another.

1.2.3.3 Asymmetric Key Agreement

A key agreement protocol is a key-establishment procedure where the resultant keying material agreed upon is a function of information contributed by two or more participants, so that no party can predetermine the value of the keying material independently of the other party's contribution. If properly done, this key agreement does not reveal to any eavesdropping third-party what key material has been agreed upon, and precludes any undesired third parties from forcing a key choice on the agreeing parties. DH, ECDH, and RSA are the three most commonly used conventional asymmetric cryptographic primitives underlying key agreement protocols (such as TLS, SSH, IPsec, and FC-SP) in use in storage systems and ecosystems today.

1.2.3.4 Asymmetric Key Encapsulation/Wrapping

Asymmetric key wrapping is a cryptographic method of encrypting a symmetric key using an asymmetric key pair. This is done to protect the confidentiality and integrity of the secret key, which can then be stored or transmitted securely. As an example, the secret key being wrapped may itself be a symmetric key such as a MEK which is used to do bulk encryption. The secret key being wrapped could, alternately, be the private key part of a different asymmetric key pair. The asymmetric key used to perform the asymmetric encryption is a KEK.

Asymmetric key wrapping is often used in scenarios where a secret key needs to be stored or transmitted over an untrusted channel, such as the Internet. In these cases, the secret key can be wrapped using the public key of the intended recipient. This ensures that the symmetric key cannot be decrypted by anyone who does not have the recipient's private key.

The two most commonly used asymmetric key wrapping algorithms used in storage systems and ecosystems today are:

- RSA Encryption System-Optimal Asymmetric Encryption Padding (RSAES-OAEP).
- RSA Encryption System- Public-Key Cryptography Standard #1 (RSAES-PKCS1) v1.5.

Key management consideration: Asymmetric key wrapping requires the use of an asymmetric key pair. The private key must be kept secure, as it can be used to decrypt any data that has been encrypted using the wrapped key.

Performance consideration: Asymmetric key wrapping and unwrapping requires more computation than symmetric key wrapping and unwrapping.

Usability consideration: Asymmetric key wrapping is relatively easy to use. This makes it a good choice for applications where security is important, but usability is also a concern. For the key transport use case, key distribution in particular is often simpler with asymmetric key wrapping, in that either or both ends can simply export their public key for the other end to use as an asymmetric KEK.

Overall, asymmetric key wrapping is a valuable tool for protecting the confidentiality and integrity of secret keys.

1.3 Key Management Overview

In general, cryptography relies on the management of cryptographic keys. All ciphers, both symmetric and asymmetric, require all the communicating parties to have access to the necessary keys. This gives rise to the need for key management involving the generation, distribution, and ongoing management of those keys. An overall framework for key management is given in ISO/IEC 11770-1[7] and NIST SP 800-57 Part 1[30].

As noted in NIST SP 800-57 Part 1, keys are analogous to the combination of a safe. If a safe combination becomes known to an adversary, the strongest safe provides no protection for its contents. Similarly, poor key management may easily compromise strong cryptographic assurance. Ultimately, the confidentiality of information protected by cryptography directly depends on the strength of the keys, the effectiveness of mechanisms and protocols associated with keys, and the protection afforded to the keys. All keys need to be protected against modification, and unauthorized disclosure. Key management provides the foundation for the secure generation, storage, distribution, and destruction of keys.

1.4 Cryptographic Strength

Cryptographic strength is a measure of the amount of work an attacker must invest to conduct a brute force attack on an unknown cryptographic key. For example, a strength of 128 bits implies that an attacker might have to try a maximum of 2^{127} keys, but a brute force attack would require half this number of keys on average before hitting on the correct one. However, cryptographic strength only tells part of the story. For this measure to be relevant, the brute force attack must be the only feasible attack available to the attacker. If the cryptographic algorithm has a weakness in its operation, then an analytic attack might be much more effective. Or if there is a weakness in the implementation of an algorithm, this may simplify the attacker's task (e.g. if the keys are not randomly chosen from the entire available keyspace or are derivable from other known values). Also, if the attacker can social engineer or gain access to the keying materials, there is no need to mount a brute force attack. Choosing a well-reviewed cryptographic algorithm and a well-vetted implementation of that algorithm help avoid weaknesses that simplify the attacker's task. Strong key management will make it much more difficult to social engineer or otherwise gain access to the keying material. The intent of encryption and proper key management is to leave the brute force attack on the ciphertext as the only option for an attacker.

According to NIST's SP 800-57 Part 1, the effective strength of the encryption should be a minimum of 112 bits for data which does not need to be protected beyond 2030. For data which does need to be protected beyond 2030, the effective strength of the encryption should be a minimum of 128 bits.

It should be noted that the present definition of effective strength presumes the attacker can at most use a conventional computer to attack the cryptography. When a Cryptographically Relevant Quantum Computer (CRQC) using Shor's algorithm becomes a reality, the existing conventional asymmetric cryptographic algorithms will cease to provide any significant protection – to the point that those algorithms will effectively be completely broken. A realized CRQC using Grover's algorithm would also undermine the strength of symmetric cryptographic algorithms, but to a lesser degree, in that it only halves (vs demolishes) the effective strength of an algorithm such as AES-256 (which will be reduced to only providing about 128 bits of effective strength against a CRQC).

2 Encryption of Data Storage

Understanding the role of encryption with storage devices, systems, and ecosystems is important. Storage-based encryption is an important protection against data breaches due to loss of control of storage devices/media (data at rest) and, to a lesser degree, eavesdropping attack within storage systems and ecosystems (data in motion). Assuring data confidentiality and integrity extends beyond encryption within storage.

Inevitably, any encryption discussion associated with storage ecosystems will include a differentiation between encryption of data in motion and encryption of data at rest. Although difficult to define, it is important to understand these concepts, which can be summarized as:

- *Encryption of Data At Rest* — encryption that protects data while it resides on the media. It involves encrypting data that will be decrypted when that data flows through the same point (or an equivalent) in the opposite direction. The point of encryption (see 3.2) may be within a storage device (tape drive encryption) or the entity in which the data was created and/or consumed (end-to-end encryption) or at any point along the data path.
- *Encryption of Data In Motion* — encryption that protects data while it is being transferred over a physical link between two communicating entities (e.g. host bus adapters (HBAs), network interface cards (NICs), switches and routers). The two entities have negotiated and implemented some form of ephemeral encryption to protect the data while in transit.
- *Encryption of Data in Use* – encryption can also be used to protect data while computations are being performed on it, which is potentially relevant to computational storage implementations. One type of encryption-of-data-in-use has the encrypted data being decrypted temporarily, within a secure enclave, to allow computation to be performed on the cleartext data before the result of that computation is re-encrypted before it exits that secure enclave. But another type, homomorphic encryption, allows computation to be performed directly on encrypted data, producing encrypted results.

As one can see in the descriptions above, an encryption of data has the potential to protect the confidentiality of the actual data as it traverses all of the down-stream communications links, depending on where in the data path the encryption is applied. Communications-based encryption (e.g. IPsec, TLS, and SSH) protects the data in motion and it may also include integrity checks to ensure the ciphertext is not changed while it is in transit.

3 Encryption of Data at Rest

3.1 Commonly Used Cryptography in Storage

Within storage systems and ecosystems, symmetric key encryption is commonly used, and the dominant symmetric block cipher algorithm is AES. Most storage solutions which perform persistent bulk encryption use some mode of AES with 256-bit keys (i.e. AES-256). Table 1 provides a list of the more commonly used modes with references for further information. The first two entries in the table, XTS-AES-256 and GCM-AES-256 are the most commonly used with storage systems. GCM-AES-256 is the most often used for secure communication channels and is capable of handling variable block lengths (e.g. tape storage, file storage, TLS, etc.).

Table 1. Common Encryption Modes of Operation for Storage

Mode of Operation	ISO/IEC	Other
XEX Tweakable Block Ciphertext Stealing (XTS)		IEEE 1619-2018[45] NIST SP 800-38E[29]
Galois/Counter Mode (GCM)	ISO/IEC 10116:2017[1]	IEEE 1619.1-2018[46] NIST SP 800-38D[28]
Cipher Block Chaining (CBC)	ISO/IEC 10116:2017	NIST SP 800-38A[26]
Counter (CTR)	ISO/IEC 10116:2017	
Counter with Cipher Block Chaining-Message Authentication Code (CCM)		IEEE 1619.1-2018 NIST SP 800-38C[27]
Extended Codebook (XCB) Encrypt Mix Encrypt V2 (EME2)		IEEE 1619.2-2021[47]

The XTS mode of operation was developed with storage in mind to guard against cut-and-paste attacks wherein an attacker attempts to replace part of the ciphertext (see RFC 4949[63]); such an attack results in corrupted data when the data is decrypted with XTS-AES. Another consideration was keeping the size of the ciphertext as close to the size of the plaintext data; Xor–Encrypt–Xor (XEX) requires processing of full blocks, which can introduce padding to ensure the last block is full. XTS is an extension beyond XEX which adds support for ciphertext stealing which in turn enables encryption without expansion, so long as the cleartext being encrypted is more than one encryption block in length (e.g. XTS-AES can be used to encrypt 19 bytes of cleartext into 19 bytes of ciphertext).

3.2 Point of Encryption

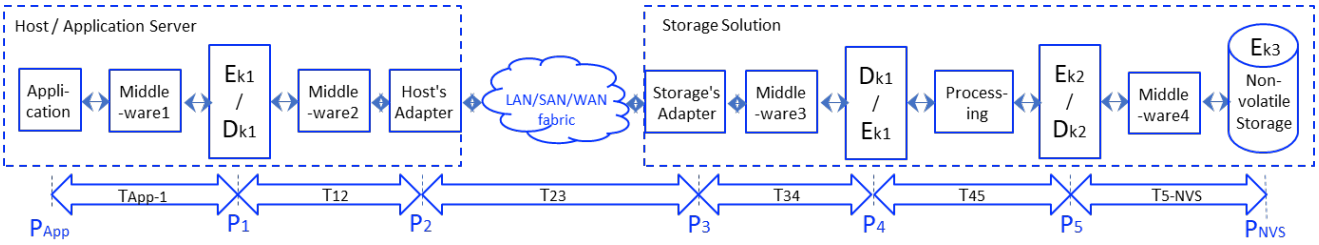
3.2.1 General

Typically, encryption of data at rest is dependent on the placement of an encryption/decryption mechanism within the data flow path, which is known as the point of encryption. The points of encryption identify where in the storage ecosystem the data is present in its plaintext form and where it is present in ciphertext.

Figure 1 shows a hypothetical scenario in which data are being written by an application to storage where the persistent data are potentially encrypted or decrypted at many points between those two endpoints. The following are noteworthy in this scenario:

- At P_{App}, encryption by the application itself.
- At P_1, inside of the host, but outside of the application (e.g. file system encryption, LUKS, VMcrypt, etc.).
- At the host's network adapter, P_2 (i.e. by the Host's network Adapter (e.g. NIC, HBA, etc.) or its device driver).
- At the storage's network adapter, P_3 (i.e. by the Storage's network Adapter (e.g. NIC, HBA, etc.) or its device driver).
- At P_4, somewhere inside of the storage solution, upstream of the non-volatile storage.
- At P_5, some other place inside the storage solution, downstream of P_4 and upstream of the non-volatile storage.
- At P_{NVS}, encryption is occurring within the non-volatile device (e.g. by a self-encrypting SSD).
- Data in Motion between any two of the points P_x and P_y above is shown as traversing an extent T_{xy}.

Figure 1. Point of Encryption Options in Storage Ecosystems



Continuing with the Figure 1 scenario, it is worth considering the following on how vulnerable data traffic is to being eavesdropped, or tampered with, over different extents along the path from P_{App} to P_{NVS}:

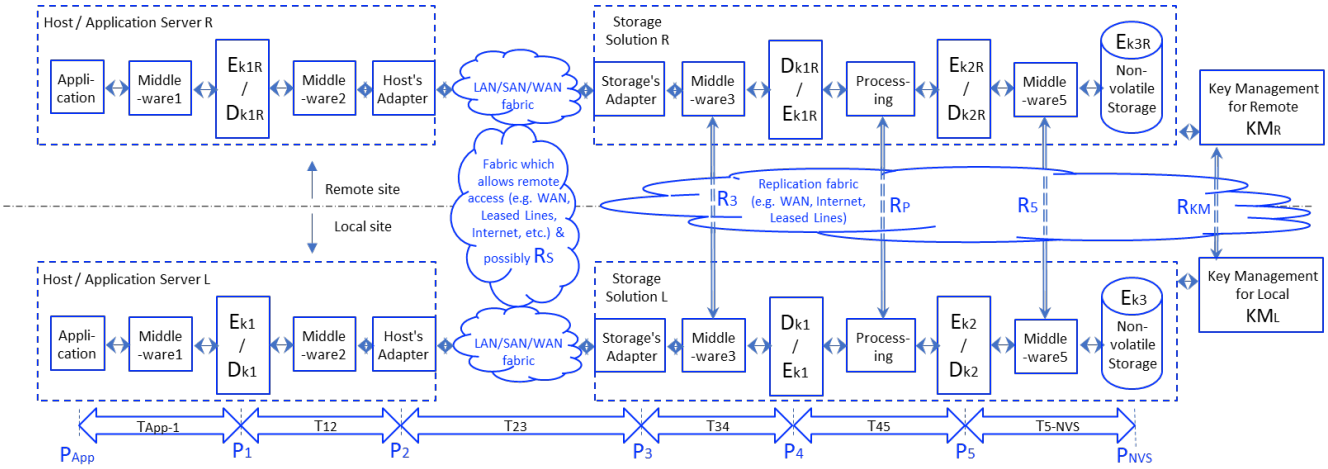
- In some cases, the storage fabric (i.e. while transiting extent T_{23}), which might be the Internet, should be considered an insecure channel.
- In other cases, the data might be considered vulnerable to being eavesdropped at any point after it leaves the host server at P_2.

- A strong form of Encryption of Data in Motion (or EDiM, e.g. TLS, IPsec, FC-SP, etc.) should be used, at minimum, to protect data traversing a truly insecure channel, even if it is already protected by a standard form of Encryption of Data at Rest, to prevent leakage of information.
- Strong EDiM typically requires mutual authentication, frequent rekeying, and use of both authenticated encryption (e.g. GCM-AES) and Perfect Forward Secrecy (PFS). Strong EDiM is ephemeral (e.g. data protected by EDiM might be encrypted at P_2 and decrypted immediately upon receipt at P_3).
- Encryption of Data at Rest is persistent encryption protecting data written to non-volatile storage and is typically rekeyed far less frequently (see 5.4.2) than EDiM and may use an unauthenticated encryption mode, such as XTS-AES, to avoid data expansion (see 3.1).
- If data is encrypted at P_1 and then decrypted at P_4, to allow processing, the Point of Encryption is where the final Encryption of Data at Rest occurs (e.g. at P_5), and is potentially vulnerable until that final point is reached (e.g. a rogue storage admin could potentially take a snapshot of the data, in the clear, after P_4 and before P_5).

As the example shows, data can be encrypted multiple times prior to being recorded on media. For example, encryption at the application layer that is then stored on a self-encrypting drive. The organization may not be fully aware of all the encryption mechanisms in use.

Further expanding the hypothetical scenario, Figure 2 adds disaster recovery elements.

Figure 2. Point of Encryption Options in Storage Ecosystems with DR



In Figure 2, data being written by an application to storage could potentially be replicated at different points within the storage solution:

- Replication of data downstream of an Encryption of Data at Rest's Point of Encryption (PoE) means that ciphertext is being replicated.
- That replicated ciphertext can only be decrypted if the decryption has access to the same key as was used to encrypt it.

- As an example, say P_5 is the final PoE locally. Data upstream of that (e.g. at P_4) is cleartext, data downstream of that is ciphertext. Replication at the *Middleware 5* layer (e.g. as represented by the R_5 arrow) is then replication of statically encrypted ciphertext.
- For Disaster Recovery of encrypted data, any keys required to decrypt must be available to the disaster recovery site.
- One way of achieving this is to have the key managers themselves replicate their key stores (see R_{KM} above).
- It should be noted that KM_L & KM_R are not necessarily collocated with Storage Solutions *L* and *R* and could be networked to almost anywhere.
- But wherever KMs are located, they can become unavailable (e.g. if network connections fail) or lost permanently (e.g. due to a disaster).
- To resiliently allow recovery of encrypted data, there should be multiple redundant copies of KM's keys at different locations (i.e. there should be *N* KMs to enable tolerance for *N-1* KMs failing, or becoming inaccessible).

3.2.2 Interactions with data reduction techniques

Data reduction techniques such as data deduplication and lossless compression do not work well on encrypted data (ciphertext). These data reduction techniques rely on patterns in data in order to gain any size reduction, while encryption attempts to destroy these patterns in data (i.e. ciphertext is usually indistinguishable from a random stream). For this reason, data is typically deduplicated/compressed prior to encryption.

3.3 Encryption Capable Drives

A self-encrypting drive (SED) is a storage device that integrates encryption of user data at rest. All user data written to the storage device is encrypted by specialized hardware implemented inside the storage device controller. The data is decrypted as it is read. This encryption and decryption is done transparently to the user.

An important difference between various storage devices that perform data at rest encryption is the way they handle key management. For the purposes of this technical paper, storage devices that employ self-contained key management are described in 3.3.1 and those that have dependencies on external key management are described in 3.3.2.

3.3.1 Internal (Self-managed) Key Management

The encryption and decryption are performed using a MEK generated internally in the storage device.

3.3.1.1 TCG Compliant Self-encrypting Drive Suite

The Trusted Computing Group (TCG) Storage Work Group (SWG) has defined and published several specifications that define architectures for storage device-based security features, which are designed to be configurable and manageable under policy-based access control. This means that the capabilities of the storage device can be configured to conform to the security policies of the trusted platform or associated organization. The primary TCG storage specification is the “TCG Storage Architecture Core

Specification"[87], or "Core Spec," which defines in detail the components that can be implemented by a storage device to provide various features. The Core Specification defines a storage interface-independent communications protocol used by host applications to manage features, as well as the data structures and commands associated with a variety of other capabilities. It is also noteworthy that the Core Specification identifies the symmetric key encryption modes in Table 2 as valid options.

Table 2. TCG Approved Encryption Modes of Operation

Cipher Block Chaining (CBC)
Cipher Block Chaining-Mask-Cipher Block Chaining (CMC)
Ciphertext Feedback (CFB)
Counter (CTR)
Counter with CBC-MAC (CCM)
Electronic Code Book (ECB)
Encrypt Mix Encrypt (EME)
Galois Counter Mode (GCM)
Liskov, Rivest, Wagner (LRW)
Output Feedback (OFB)
Xor–Encrypt–Xor (XEX)
Xor–Encrypt–Xor Tweakable Block Ciphertext Stealing (XTS)

The TCG uses "Storage Security Subsystem Class," or "SSC," specifications to provide implementation profiles for storage devices that incorporate specific functionality. These SSCs explicitly define the minimum acceptable capabilities of a storage device in a specific "class". The TCG Storage Security Subsystem Class: Enterprise[88] and TCG Storage Security Subsystem Class: Opal[89] are two examples of published SSCs.

Enterprise SSC defines the functionality for implementing the Core Specification on storage devices for high performance storage systems. Enterprise SSC provides data at rest protection via data encryption (hardware-based) and access controls, and fast repurposing of the storage device. Enterprise SSC specifies multiple storage ranges with each having its own authentication and encryption key. The range start, range length, read/write locks as well as the user read/write access control for each range are configurable.

The Opal "Family" of specifications published by the Trusted Computing Group (TCG) provides a scalable infrastructure for managing encryption of user data in a storage device, as well as extensibility to enable features beyond "data at rest protection." Ruby[90], Opalite[91] and Pyrite[92] are subset specifications to the Opal specification; the following is a summary of the major features and differences:

- Opal SSC – Opal SSC is a cryptographic toolkit that provides device access control, multiple encryption zones, administrative control of security related settings, which together provide protection of data at rest.

- Ruby SSC – Ruby is a subset Opal SSC for storage devices deployed within, but not limited to, data center, bulk data, and enterprise class systems.
- Pyrite SSC – Pyrite is a subset of Opal SSC that uses minimal resources and does not specify capabilities for cryptographic protection of data at rest. As such, references to and support for capabilities related to encryption, cryptographic erase, etc. are absent from Pyrite.
- Opalite SSC – Opalite is a subset of Opal SSC that uses minimal resources.

Table 3 provides a comparison of the Storage SSCs.

Table 3. Comparison of TCG SSC Capabilities

Feature	Opal V2.02 SSC	Opalite SSC v1.00 (Opal v2.01 subset)	Pyrite SSC (Non-encrypting subset of Opal)	Ruby SSC v1.00	Enterprise SSC
Core Spec Version Supported	V2.01	V2.01	V2.01	V2.01	V1.01
Activation and Life Cycle	Yes	Yes	Yes	Yes	No
Number of Admins/Users (Password authenticated authorities)	4 Admin, 8 User	1 Admin, 2 User	1 Admin, 2 User	1 Admin, 2 User	Requires 1 and allows up to 1024 “Bandmaster”, 1 “Erasemaster” (No Admin Support)
Min Number of Required LBA Ranges	Global Range + 8	Global Range only	Global Range only	Global Range Additional Locking Ranges are optional	Global Range + 1023 additional locking ranges
Min DataStore Size (General Purpose Storage)	10 MB	128 KB	128 KB	128 KB	1KB

Min MBR Table Size	128 MB	128 MB	128 MB (Optional)	128 MB (Optional)	MBR support is optional MBR min size is not defined
Configurable Access Control	Yes	Yes	Yes	Yes	No
Mandatory Feature Sets	PSID Block SID Authentication Additional Datastore Tables	PSID Block SID Authentication	PSID Block SID Authentication	PSID Block SID Authentication	None
Media Encryption	Required	Required	Not Specified	Required	Required
Crypto Erase	Revert, RevertSP, GenKey methods for device and locking range level erase granularity	Revert, RevertSP, GenKey methods for device and locking range level erase granularity	No user data erase supported – relies on native interface erase capability	Revert, RevertSP, GenKey methods for device and locking range level erase granularity	Erase Method

The Core and SSC specifications are interface agnostic. The TCG Storage Interface Interaction Specification (SIIS)[94] defines mappings to each interface's container commands. SIIS mappings also include mappings of TCG storage-defined:

- errors to relevant errors from the underlying interfaces;
- reset events to reset events from the underlying interfaces;
- interactions between TCG SSC "methods" and some operations defined by the underlying interfaces.

For the TCG SSC specifications only a single threat scenario is addressed: removal of the storage device from its host system involving a power cycle (and possibly drive resets) of the storage device and subsequent unauthorized access to data stored on that device.

Depending on a self-encrypting drive's compliance with a TCG SSC specification, the drive includes hardware-based, data at rest encryption (except Pyrite) as well as other security services such a drive locking, authentication, and storage sanitization.

3.3.1.2 Non-TCG Self-encrypting Drives

There are non-TCG compliant self-encrypting drives that provide hardware-based encryption as well as security services. These SEDs are typically bundled with vendor software that is required to use the SED functionality (i.e. proprietary interface). Drives of this type are capable of performing bulk encryption for protecting data at rest.

3.3.2 External Key Management

External key management for a drive refers to key management that is performed external to the encryption-capable drive. External key management has been implemented in software running in a larger storage controller, in a VM, be part of a host application that is using the storage, or be run in a stand-alone appliance.

In environments which require high availability, multiple key servers are used and they replicate their key stores between one another in a key management plex, so if one key server in that plex becomes unavailable, the keys required to encrypt and decrypt the storage can be obtained from a different key server in that plex. In some cases, a given key server plex includes one or more local servers and one or more remote key servers. Local key servers can be very local (e.g. running inside the storage controller or automation with the encryption-capable drive). Remote key servers are sometimes quite remote (even on another continent).

In addition to high availability use cases, there may be requirements for geographically diverse key management services. Such requirements are intended to address force majeure or smoking crater events where the key management servers at a particular location may become unavailable.

It should be noted that even “self-encrypting” drives, once ownership has been established, require a PIN to be provided to them, to unlock them on boot. While there are different ways that PIN might be provided, that PIN might be kept in an external key manager and served to the “self-encrypting” drive directly. Or the external key manager might serve a higher-level master key to a storage controller containing the “self-encrypting” drives, allowing the storage controller to unwrap or derive the PINs necessary to unlock those “self-encrypting” drives.

3.3.2.1 Tape Drives

Tape drives were the first type of storage building block to be extended to add native support for strong bulk symmetric encryption. The two most commonly used tape drive families available today which are encryption-capable and used for data storage are the Linear Tape Open (LTO) family and IBM’s machine type 3592 family. These two tape drive families both rely on external key management and use GCM-AES-256 encryption, but differ somewhat as far as what that external key management is doing.

- Encryption-capable LTO drives (encryption-capability was first added to LTO-4 drives and has been available on all subsequent generations) require a symmetric key to be served to the LTO drive, which is sometimes referred to as the “direct” key model. The symmetric key served to the LTO drive was used in earlier generations directly as the MEK used to perform GCM-AES-256 encryption of the data written to tape. In that case, the external key manager provided the only key storage of the MEKs used to encrypt those LTO tape cartridges, though a given MEK could be used to encrypt one or many LTO tape cartridges. Later generations have provided a layer of indirection, such that the symmetric key served to the LTO drive is used by the tape drive as the KEK required to unwrap a symmetrically-wrapped MEK stored on the LTO tape cartridge. In that

case, while the external key manager is not storing the MEK directly, it provides the only storage of the symmetric KEK(s) required to gain access to the MEK(s) used by the LTO drives to encrypt a LTO tape cartridge.

- Encryption-capable 3592 drives (encryption-capability was first added to 3592 second generation drives) support both the “direct” key model used by LTO as well as an alternate “wrapped” key model. In the “wrapped” key model, the external key manager serves not only the MEK, but also a wrapped version of that MEK, to the 3592 drive. The external key manager does the key wrapping required, using either a symmetric or an asymmetric KEK, depending on how the key manager is configured to wrap, which in turn depends on what use cases that are to be enabled. When using the “wrapped” model, the 3592 drive writes that wrapped MEK to the 3592 tape cartridge. The MEK is the secret inside that wrapped key stored to the 3592 tape cartridge. On a subsequent load, the wrapped key containing the MEK is transferred over to the external key manager, which unwraps it and then securely serves the resultant MEK to the 3592 drive. In this case, the MEKs are not stored persistently by the key manager, though the key manager must retain the key needed to unwrap the wrapped MEKs stored to the cartridges.

Organizations often use tape as part of their business continuity management and cyber recovery strategies[44] and these tapes are typically encrypted to guard against data breaches. When these tapes are to be used in an alternate or backup site, key management issues should be carefully considered. In addition, the backup software and servers that recorded the data to tape may not be available, so alternate systems will need to have details associated with the tapa catalogue as well as the required encryption keys.

3.3.2.2 Key Per I/O Drives

Key Per I/O (KPIO) is a fine-grained data-at-rest encryption mechanism within some SSDs that builds upon NVMe’s Key Per I/O feature in the NVMe 2.0[77] specification family. The Trusted Computing Group (TCG) has specified a particular version of KPIO with specific details included in the TCG Storage Security Subsystem Class (SSC): Key Per I/O v1.0[93] and the TCG Storage Interface Interactions Specification (SIIS) v1.11 or later. The TCG Storage Key Per I/O SSC provides security management interfaces that enable a host to control and specify MEKs that a Storage Device uses for user data encryption.

KPIO is a particularly useful encryption technology for scenarios where an SSD is being shared by multiple hosts and/or multiple tenants on a host (e.g. VM, containers, etc.). Since a KPIO-enabled SSD plays no significant role in the key management lifecycle, the keys can come from a variety of sources. This can result in a drive having encrypted data (ciphertext) protected by numerous data encryption keys, which may only be known to a particular host and/or tenant.

The basic concept is that the encryption engine (XTS-AES-256) of a storage device is used to encrypt some or all data in the storage device, but the key management lifecycle is handled external to the storage device. To do this, a small volatile key cache (typically a few hundred to a few thousand keys) on the storage device is loaded with data encryption keys that the storage device will use. All read and write operations for the KPIO managed portion of the storage device include a key tag that is used as a pointer into the key cache. The storage device uses the data encryption key stored in the specified key cache location to either encrypt (for writes) or decrypt (for reads).

The alignment of host/tenants is typically for the whole drive (sub-system) or some number of namespaces. In NVMe terminology, a namespace is a collection of logical block addresses (LBA)

accessible to host software. For a KPIO-enabled drive, the key cache is carved up in such a way that each namespace is permitted to inject and use a maximum number of keys (the total for all namespaces must not exceed the SSD's key cache size); key tags specified on reads/writes are relative to the namespace (e.g. key tag 1 can exist for each namespace). The host/tenant is not restricted to a maximum number of keys and may actually use a very large number of keys. This is accomplished by deleting and injecting keys as they are needed (typically before actual I/O) while remaining within the limits of the number of keys for the namespace.

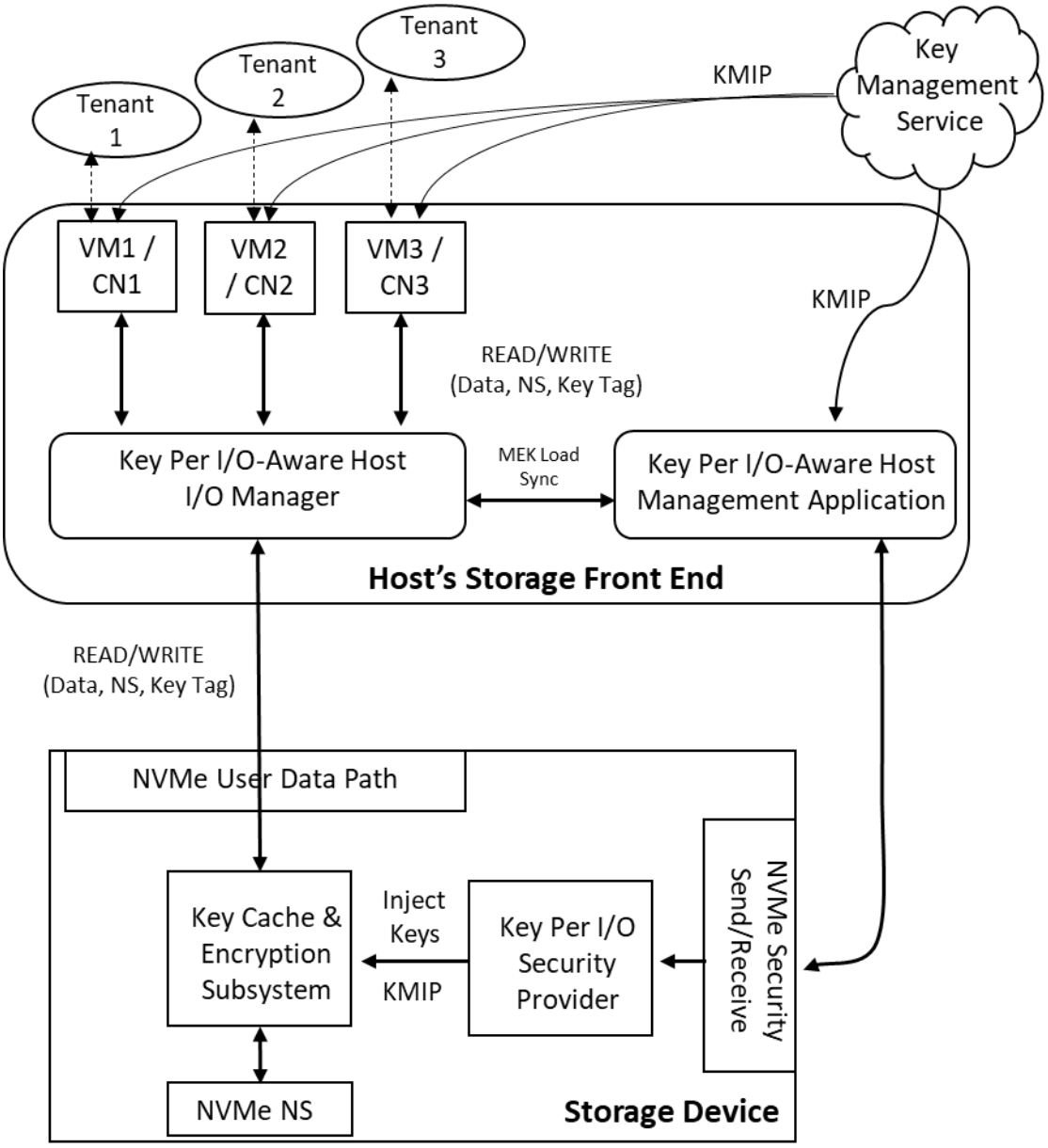
KPIO itself has no provisions to secure the communications with the SSD, so transport encryption should be used to protect data and keys (e.g. PCIe IDE[79]). The KPIO point of encryption is within the SSD.

To use the KPIO functionality, KPIO must be activated. Once activated, the first KEK can be injected, using a wrapped KEK using a PKI public key or a plaintext KEK either over a secure communication path (e.g. PCIe IDE) or in a secure environment. Once this initial KEK has been injected, it can then be used to inject a KEK for each namespace. These namespace KEKs are then used to wrap the MEK associated with each of the namespaces. These KEKs are typically retained in non-volatile storage so that they do not have to be reloaded after SSD resets/power cycles; MEKs, however, typically need to be reloaded after a power cycle.

Due in part to the popularity of the OASIS Key Management Interoperability Protocol (KMIP)[72] in handling keys for storage-based encryption, KPIO uses KMIP v2.0 based commands to inject and manage keys in the SSD; these KMIP commands are compatible with the KMIP 2.1 specification. Under the right conditions it may be possible for a host to leverage a key management server (see Figure 3), using KMIP, in such a way that the host may be able to serve as a KMIP proxy between the KMS and the SSD.

Configuring and managing KPIO may be separate from the virtual machine (VM), container (CN), or host using KPIO. This may be necessary when one or more hosts have limited visibility of SSD's namespaces. Figure 3 shows such an example where multiple VMs on a host are using Key Per I/O, but the host is controlling the injection of keys for all the VMs as opposed to each VM injecting its own keys into the storage device.

Figure 3. Key Per I/O Example



3.4 Storage Controller Encryption

Some storage arrays and NAS filers include encryption capabilities that operate at the controller level. Basically, the disk array controllers (or similar technology) encrypt the data prior to sending it to the drives. All data written to the drives are encrypted.

The encryption mechanism can be implemented in software or may be accelerated using hardware.

Most storage controller encryption solutions include both integrated key management capabilities as well as being able to leverage centralized key management. In the latter case, the key management is achieved using the OASIS Key Management Interoperability Protocol (KMIP)[72], which further described in 5.5.

3.5 Network-based Encryption

Early implementations of storage-based encryption were achieved using encryption appliances within storage area networks (SANs) and/or in front of NAS filers. As encryption has become an embedded capability in storage systems, many of these early solutions have disappeared.

There are, however, network-based implementations that are embedded in storage network routers/switches (e.g. Fibre Channel). These types of solutions typically intercept the storage networking traffic, encrypt the payload destined to be stored (using a data at rest technique rather than a data in motion technique), and then forward the ciphertext to the storage system. This in-line data at rest encryption is handled transparently to the hosts and storage systems.

3.6 Host & Application Encryption

While not considered storage-based encryption, host-level and application-level encryption are often employed. The use of these types of encryption technologies typically are not a source of problems for down-stream encryption (e.g. at the drive level), but they can have other impacts. For example, their use can impact the effectiveness of data reduction technologies such as compression and deduplication. In addition, the existence of encryption mechanisms has to be factored into data backup and business continuity management strategies.

The following are common forms of encryption mechanisms:

- Full Disk Encryption (FDE), also known as whole disk encryption, is the process of encrypting all the data on the drive used to boot a computer, including the computer's operating system, and permitting access to the data only after successful authentication to the FDE product. A user must log into the Pre-Boot Environment (PBE) with valid credentials. Once the user is authenticated to the PBE, the software FDE decrypts and boots the operating system.
- File encryption is the process of encrypting individual files or sets of files on a system and permitting access to the encrypted data only after proper authentication is provided.
- Platform encryption is provided by the operating system for platform-wide data encryption, transparently encrypting sensitive user data. With the exception of the hardware-specific requirements (e.g. hardware backed secure key storage) there is little distinction between platform encryption and file encryption implementations.

Some of these solutions can use centralized key management.

At the application level, specific files and certain records (e.g. in a database) can be encrypted under the control of the application. At the application level it may be possible for clients to inject keys.

3.7 Encryption for Virtual Environments

Many virtual environments (e.g. virtual machines and containers) can employ data at rest encryption. The encryption mechanism is typically implemented in software.

For hypervisors and virtual machines (VMs), the VMs are designated as encrypted. As part of configuring the VM, virtual disks are added and then designated as being encrypted. While it may be possible to have unencrypted virtual disks associated with an encrypted VM, this is not recommended. Once configured properly, all data written to an encrypted virtual disk is stored as ciphertext on the physical storage devices/media.

Options for encrypting containers vary and differ from VMs in that containers do not include an operating system. As such, traditional approaches for data at rest encryption that involve the use of host-based capabilities (see 3.6) may be incompatible with containers[42].

Some of these solutions can use centralized key management.

3.8 Cloud Storage-based Encryption

For the purpose of this document, cloud storage-based encryption is focused on encryption mechanisms that are supplied by the cloud service provider (CSP). That said, the encryption options can vary depending on the service being used. For example, a software as a service (SaaS) implementation is likely to be completely controlled by the CSP and could be host or application-based (see 3.6). For platform as a service (PaaS) or infrastructure as a service (IaaS), the cloud service customer (CSC) or the CSP may have provided the encryption mechanism, but again, it is likely to be host or application based.

4 Encryption of Data in Motion

High value or sensitive data are frequently exchanged between systems over wide area networks using TCP/IP, Fibre Channel over IP (FCIP)[56], iSCSI[66][67], and other protocols. Data may also be transferred from host computer systems to storage devices through storage area networks (SANs) using Fibre Channel, Serial SCSI (SAS), and other SAN protocols. In each case, there may be specific security protocols available with each particular network protocol to secure transport of the data. In some cases, there may also be a transport level security mechanism which can protect data being transmitted. Choosing the specific protection mechanism and points of encryption are important factors not only in securing the data, but also in meeting applicable compliance requirements.

In general, data in motion encryption only provides protection while data are flowing across a communication medium (e.g. a network). Unlike data at rest encryption, which uses static keys, data in motion encryption typically uses ephemeral keys that are designed to be used only for a single session or transaction. To guard against future compromises of past sessions, data in motion encryption may also employ PFS techniques that include changing the keys used to encrypt and decrypt information, frequently and automatically.

Even if data is already encrypted by the filesystem, host bus adapter (HBA), etc., end-to-end data in motion encryption offers additional protection such as support for mutual authentication, use of authenticated encryption (e.g. GCM-AES), etc. as described in section 3.2.1. Where data in motion

protection is not end-to-end, protect data wherever they may be vulnerable to eavesdropping or manipulation.

4.1 Transport Layer Security (TLS)

TLS is an IETF-specified protocol that provides communications security over networks. It allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. TLS is layered on top of a reliable transport protocol (e.g. TCP), and it is used for encapsulation of various higher-level protocols (e.g. HTTP).

There are multiple versions of TLS. Version 1.2 of TLS is specified in IETF RFC 5246[64] and is widely used. The more recent version 1.3 of TLS is specified in IETF RFC 8446[70] and is expected to supplant TLS Version 1.2 over time. Earlier, and less secure, versions of TLS are also specified and are less widely used: TLS versions 1.0 is specified in IETF RFC 2246[50] and TLS versions 1.1 is specified in IETF RFC 4346[62]. The predecessor to TLS, The Secure Sockets Layer (SSL), and in particular, version 3.0 is also still used, but is also considered less secure; SSL 3.0 is documented in the historical IETF RFC 6101[65].

TLS provides endpoint authentication and communications privacy over the network using cryptography. Typically, only the server is authenticated (i.e. its identity is ensured) while the client remains unauthenticated; this means that the end user (whether an individual or an application) has a measure of assurance with whom they are communicating. Mutual authentication (the identities of both endpoints are verified) requires, with few exceptions, the deployment of digital certificates (along with the associated private keys) on the client.

TLS involves three basic phases:

- Peer negotiation for algorithm support.
- Key exchange and authentication.
- Symmetric cipher encryption and message authentication.

During the first phase, the client and server negotiate cipher suites, which determine the ciphers to be used, the key exchange, authentication algorithms, and the Message Authentication Codes (MACs). The key exchange and authentication algorithms are typically public key algorithms. The MACs are made up from a keyed Hash Message Authentication Code (HMAC).

SNIA provides specific requirements and recommendations in the *SNIA TLS Specification for Storage Systems*, also known as ISO/IEC 20648[15].

4.2 IP Security Protocol (IPsec)

IETF RFC 7146[68] require the use of IPsec Version 3 along with Internet Key Exchange (IKE) version 2 to secure the communication channel to protect sensitive or high value data for both iSCSI and FCIP. IPsec version 3 is described by a suite of IETF documents: RFC 4301[58], RFC 4302[59], RFC 4303[60], and RFC 4306[61].

In addition, IETF RFC 3723[57] places the following requirements on the IPsec suite used with iSCSI and Fibre Channel over TCP/IP (FCIP) protocols:

- Confidentiality: ESP with 3DES in CBC mode as described in IETF RFC 2451[53] must be supported, although AES in Counter mode as described in IETF RFC 3686[55] should be supported. The use of AES encryption in new implementations is highly recommended (see NIST Special Publication 800-111[32]).
- Authentication and Integrity: HMAC-SHA1 described in IETF RFC 2404[51] must be supported. AES in CBC MAC mode with XCBC extensions described in RFC 3566[54] should be supported. The use of AES encryption in new implementations is highly recommended. DES in CBC mode should not be used.
- IPsec Modes: ESP in tunnel mode per IETF RFC 2406[52] must be supported. IPsec with ESP in transport mode may be supported.

4.3 Fibre Channel (FC) ESP_Header

Fibre Channel (FC) storage area networks can leverage an assortment of security capabilities, including protocols to authenticate Fibre Channel entities, set up session keys, negotiate parameters to ensure frame-by-frame integrity and confidentiality, and define and distribute policies across a Fibre Channel fabric.

ANSI INCITS 545–2019 (FC-FS-5)[18] defines optional headers that can be used within Fibre Channel frames. Of these optional headers, the ESP_Header and ESP_Trailer play an important security role because they are the mechanism used to provide origin authentication, integrity assurance, anti-replay protection, and confidentiality.

IETF RFC 4303 describes an updated version of Encapsulating Security Payload (ESP), which is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality.

ANSI INCITS 496-2012 (FC-SP-2)[19] defines how to use ESP in Fibre Channel. When FC-SP-2 is used, the Authentication option is required, and Confidentiality may be negotiated by the two communicating FC_Ports (see FC-SP-2).

FC-FS-5[18] requires that end-to-end ESP_Header processing be applied to FC frames in transport mode (see RFC 4303, and link-by-link ESP_Header processing be applied to FC frames in tunnel mode[3] (see RFC 4303).

NOTE - An intended application of Link-by-link ESP_Header processing is to secure a link in a Fabric or between Fabrics without requiring use of ESP by every Nx_Port[4] .

Many of these mechanisms can be complex to understand and challenging to configure properly. Recognizing this situation, SNIA has developed a separate technical paper, *SNIA Storage Security: Fibre Channel Security*[74], which covers FC security in more depth.

---

[3] In "tunnel mode" the internal routing information is protected by encrypting the header of the original packet/frame whereas "transport mode" only protects the payload with encryption.

[4] The term Nx_Port is used to refer to either an N_Port (node port) or an NL_Port (node loop port)

4.4 PCIe IDE

PCI Express (PCIe)[78] is a high-speed serial computer expansion bus standard, which is based on a point-to-point topology, with separate serial links connecting every device to the root complex (host). PCIe is specified by the Peripheral Component Interconnect Special Interest Group (PCI-SIG). The PCI-SIG has also specified an optional Integrity and Data Encryption (IDE)[79] mechanism which defines confidentiality, integrity, and replay protection capabilities on the PCIe interface to perform hardware encryption and integrity checking, using AES-GCM with 256-bit keys, on packets transferred across PCIe links. Two types of IDE Streams are defined:

- Link – valid only from one device directly connected to another.
- Selective – intended to be carried across PCIe switches (e.g., from the root complex to the device).

Mechanisms within the IDE mechanism protect against further attacks such as forcing retries and injecting packets in attempts to force repeated transmission of the same data to expose the cryptographic keys being used.

Self-encrypting storage devices that use the NVMe protocol over the PCIe interface typically transfer data in the clear. With the PCIe IDE mechanism, across-the-wire protections (both confidentiality and integrity) can be added to mitigate specific risks, e.g., confidentiality in a multi-tenet storage system, or integrity in real time control systems such as in the automotive environment.

The PCIe IDE mechanism is also used in PCIe's Trusted Execution Environment (TEE) where a Virtual Machine (VM) is configured for privacy with IDE before enabling the VM for user activity, and where the configuration remains locked until the VM is torn down.

4.5 SPDM Secure Sessions

The DMTF Secure Protocol and Data Model (SPDM)[83][84] defines message formats, data objects, and sequences for performing message exchanges that can be used for a wide range of security functions, including device identity collection, device authentication, measurement collection, and device secure session establishment.

SPDM is being leveraged by other groups, including:

- Peripheral Component Interconnect Special Interest Group (PCI-SIG) with PCI express (PCIe) Integrity and Data Encryption (IDE)[79] [80][81] Key Management (IDE_KM) protocol for PCIe link encryption and TEE Device Interface Security Protocol (TDISP)[82].
- Compute Express Link (CXL) Consortium[85] with CXL Integrity and Data Encryption (IDE) Key Management (IDE_KM) protocol for CXL link encryption and TDISP for confidential computing.
- Mobile Industry Processor Interface (MIPI) with Service Association Configuration Protocol (SACP)[86] for automotive system security management.
- Trusted Computing Group (TCG) with Platform Firmware Profile Standard rely upon SPDM-based device attestation for the associated the platform.
- Open Compute Project (OCP) with Attestation of System Components requirements.

The SPDM protocol is similar to the network Transport Layer Security (TLS) protocol, but it is customized for the communication between two device entities. SPDM differs in that individual supported algorithms, such as hash algorithms, responder direction asymmetric digital signature algorithms, requester direction asymmetric digital signature algorithms, key exchange algorithms and Authenticated Encryption with Associated Data (AEAD) ciphers can be negotiated separately.

SPDM defines a secure session establishment mechanism between two entities by using Diffie Hellman ephemeral (DHE) or Elliptic Curve DHE (ECDHE) key exchange with asymmetric authentication such as RSA or Elliptic Curve Digital Signature Algorithm (ECDSA); future versions of SPDM are expected to add support for PQC algorithms as well. For a device that only supports symmetric cryptography, the secure session can also be established with a pre-shared key (PSK). Once the session is created, two entities can use Authenticated Encryption with Associated Data (AEAD) for message communication.

5 Key Management

5.1 General

Proper key management for encrypted data, both at rest and in motion, is crucial to the confidentiality and availability of that data. While key management of data in motion is important, keys used in this situation are usually ephemeral and generated through automated key exchange methods. Further, the amount of data secured with a single key is usually small. Keys for the protection of data at rest require greater care, since the lifespan of the encrypted data may be long and the amount of data secured by a single key may be substantially greater than those used for data in motion.

According to ISO/IEC 11770-1, key management is the administration and use of the following services associated with keys and keying material:

- Generation – generate keys in a secure way for a particular cryptographic algorithm.
- Derivation – creates a potentially large number of keys (derived keys) using a secret original key called the derivation key, non-secret variable data, and a transformation process (which also need not be secret).
- Registration – associates a key with an entity.
- Certification – assures the association of a public key with an entity and is provided by a certification authority (e.g. creates a key certificate).
- Distribution – set of procedures to provide key management information objects securely to authorized entities.
- Installation – establishment of the key within a key management facility in a manner that protects it from compromise; mark the key as “in use.”
- Storage – provides secure storage of keys intended for current or near-term use or for backup.
- Archiving – provides a process for the secure, long-term storage of keys after normal use.
- Revocation – assures the secure deactivation of keys when the compromise of a key is suspected or known.

- Deregistration – procedure provided by a key registration authority that removes the association of a key with an entity. Typically part of the destruction process.
- Destruction – provides a process for the secure destruction of keys that are no longer needed.

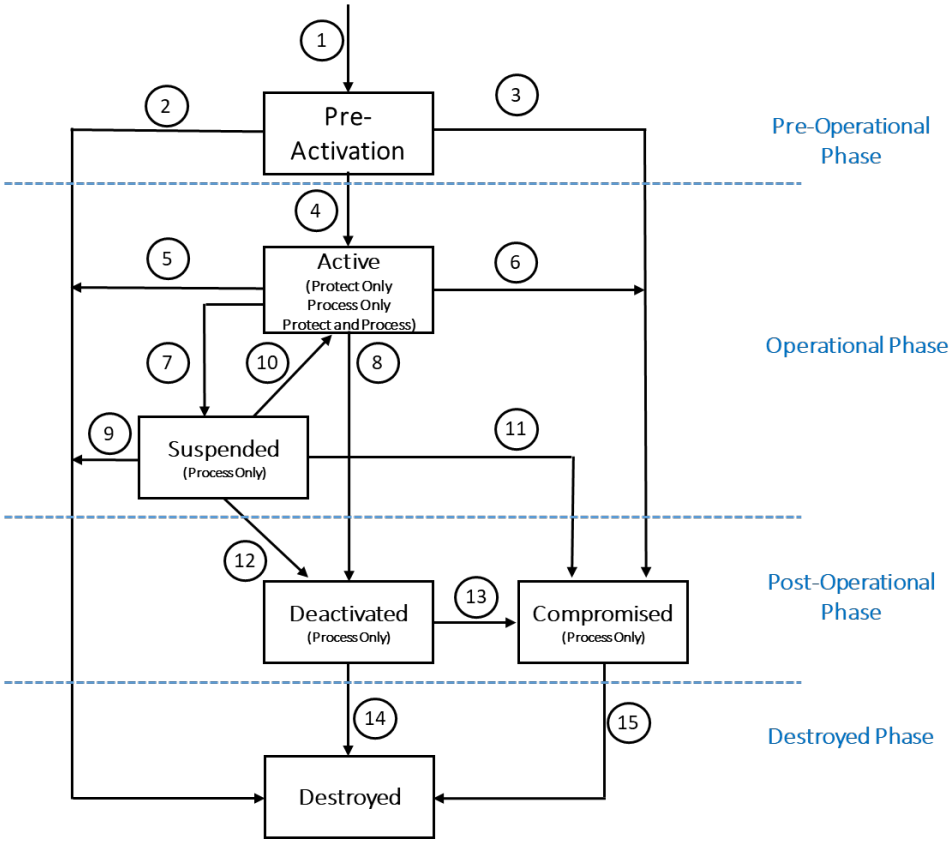
These services may be part of a key management system or be provided by other service providers. The use or application of a key may determine the services for that key. For example, a system may decide not to register session keys, since the registration process may last longer than their lifetime. In addition, keys for particular cryptographic techniques will use different combinations of services during their lifecycles (see 5.2).

5.2 Key Management Lifecycle

The lifecycle of keys and keying material is an important consideration of any key management scheme. Key management considerations include the generation of keys, secure distribution of the keys, and activation and deactivation of keys. Additionally, procedures must be put into place governing how keys are archived, destroyed at the end of their useful life, and how key compromises should be handled.

The NIST key lifecycle system is shown in Figure 4[30].

Figure 4. NIST Key Lifecycle



The cryptographic key-management lifecycle can be divided into four phases. During each phase, the keys are in certain specific key states. In addition, within each phase, certain key-management functions are typically performed. These functions are necessary for the management of the keys and their associated metadata. The four phases of key management are:

A. Pre-operational phase: The keying material is not yet available for normal cryptographic operations. Keys may not yet have been generated, or have been generated but are as yet unused and so in the pre-activation state. System or enterprise attributes such as associated metadata (key name, keyID, Globally Unique Identifier (GUID), key type, cryptoperiod, usage period, etc.) are typically established during this phase as well.

B. Operational phase: The keying material is available and in normal use. Keys are in the active or suspended state. Keys in the active state may be designated as for protect only, for process only, or for both protect and process; keys in the suspended state can be used for processing only.

C. Post-operational phase: The keying material is no longer in normal use, but access to the keying material is possible, and the keying material may be used for processing protected information. Keys are in the deactivated or compromised states. Keys in the post-operational phase may be in an archive.

D. Destroyed phase: Keys are no longer available. Records of their existence may or may not have been deleted. Keys are in the destroyed state. Although the keys themselves may have been destroyed, some or all of the key's associated metadata (e.g. key name, keyID, GUID, key type, cryptoperiod, and usage period) may be retained.

A key may be used differently, depending upon its state in the key's lifecycle. Key states are defined from a system point of view as opposed to the point of view of a single cryptographic module. Additional states may be applicable for some systems, and some of the identified states may not be needed by some other systems.

The following state transitions are identified in Figure 4:

- State Transition 1: A key enters the pre-activation state immediately upon generation.
- State Transition 2: If a key is in the pre-activation state, and it has been determined that the key will not be needed in the future, the key shall then be destroyed and thus transition directly from the pre-activation state to the destroyed state.
- State Transition 3: When a key is in the pre-activation state, and the integrity of the key or the confidentiality of a key requiring confidentiality protection becomes suspect, then the key shall transition from the pre-activation state to the compromised state.
- State Transition 4: Keys shall transition from the pre-activation state to the active state when the key becomes available for use. This transition may occur upon reaching an activation date or may occur because of an external event. In the case where keys are generated for immediate use, the transition occurs immediately after entering the pre-activation state.

This transition marks the beginning of the cryptoperiod (see 5.4.2).

- State Transition 5: Several key types transition directly from the active state to the destroyed state if no compromise has been determined and either the key's cryptoperiod has been reached or the key has been replaced.

- State Transition 6: A symmetric key or asymmetric key pair shall transition from the active state to the compromised state when the integrity of the symmetric key or the confidentiality of an asymmetric key requiring confidentiality protection becomes suspect. In this case, the key or key pair shall be revoked.
- State Transition 7: When a suspended state is used by an application, a symmetric key or both keys of a key pair shall transition from the active state to the suspended state if, for some reason, the key or key pair is not to be used for a period of time (i.e. the suspension period). For example, a private signature key may be suspended because the entity associated with the key is on a leave of absence or there is suspicion that the key may have been compromised. In the latter case, the suspension will allow for an investigation of the key's status before initiating costly revocation and replacement processes.
- State Transition 8: A key or key pair in the active state shall transition to the deactivated state when it is no longer to be used to apply cryptographic protection to data. The transition to the deactivated state may be because a symmetric key was replaced, the end of the originator-usage period has been reached, or the key or key pair was revoked for reasons other than a compromise (e.g. the key's owner is no longer authorized to use the key to encrypt data).
- State Transition 9: Several key types transition from the suspended state to the destroyed state if no compromise has been determined.
- State Transition 10: A key or key pair in the suspended state shall transition to the active state when the reason for the suspension no longer exists and the end of the originator-usage period has not been reached.
- State Transition 11: A key or key pair in the suspended state shall transition to the compromised state when the integrity of the key or the confidentiality of a key requiring confidentiality protection becomes suspect or is confirmed. In this case, the key or key pair shall be revoked.
- State Transition 12: Several key types transition from the suspended state to the deactivated state if no compromise has been determined and the suspension is no longer required.
- State Transition 13: A key shall transition from the deactivated state to the compromised state when the integrity of a key or the confidentiality of a key requiring confidentiality protection becomes suspect. In this case, the key or key pair shall be revoked.
- State Transition 14: When no longer needed (e.g. to decrypt data), symmetric (secret) and asymmetric private keys in the deactivated state shall transition to the destroyed state.
- State Transition 15: A compromised symmetric (secret) or asymmetric private key shall transition to the destroyed state.

5.3 Key Types and Other Information

There are several different types of cryptographic keys, each used for a different purpose; Table 3 shows the types of keys defined in NIST SP 800-57 Part 1 (Revision 5). Key types that are commonly used within storage systems and ecosystems are highlighted in Table 4.

Table 4. NIST SP 800-57 Part 1 Key Types

Key Type	Description
Private signature key	The private key of asymmetric-key (public-key) key pairs that are used by public-key algorithms to generate digital signatures intended for long-term use.
Public signature-verification key	The public key of an asymmetric-key (public-key) key pair that is used by a public-key algorithm to verify digital signatures that are intended to provide source authentication and integrity authentication as well as support the non-repudiation of messages, documents, or stored data.
Symmetric authentication key	A key used with symmetric key algorithms to provide identity authentication and integrity authentication of communication sessions, messages, documents, or stored data.
Private authentication key	The private key of an asymmetric-key (public-key) key pair that is used with a public-key algorithm to provide assurance of the identity of an entity (i.e., identity authentication) when establishing an authenticated communication session or authorization to perform some action.
Public authentication key	The public key of an asymmetric-key (public-key) key pair that is used with a public-key algorithm to provide assurance of the identity of an entity (i.e., identity authentication) when establishing an authenticated communication session or authorization to perform some action.
Symmetric data-encryption key	Key used with symmetric-key algorithms to apply confidentiality protection to data (i.e., encrypt plaintext data).
Symmetric key-wrapping key	Key used with symmetric-key algorithms to encrypt other keys.
Symmetric random number generation keys	Key used to generate random numbers or random bits.
Symmetric master key/key-derivation key	A symmetric master key is used to derive other symmetric keys (e.g., data-encryption keys or key-wrapping keys) using symmetric cryptographic methods.
Private key-transport key	The private keys of asymmetric key (public-key) key pairs that are used to decrypt keys that have been encrypted

	with the corresponding public key using a public-key algorithm.
Public key-transport key	The public keys of asymmetric-key (public-key) key pairs that are used to encrypt keys using a public-key algorithm.
Symmetric key-agreement key	Key used to establish symmetric keys (e.g., key-wrapping keys, data-encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors) using a symmetric key-agreement algorithm.
Private static key-agreement key	Key is the long-term private key of asymmetric-key (public-key) key pairs that are used to establish symmetric keys (e.g., key-wrapping keys, data-encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).
Public static key-agreement key	Keys is the long-term public key of asymmetric-key (public-key) key pairs that are used to establish symmetric keys (e.g., key-wrapping keys, data-encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).
Private ephemeral key-agreement key	Key is the short-term private key of asymmetric-key (public-key) 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).
Public ephemeral key-agreement key	Key is the short term public key of asymmetric key pairs that are used in a single key-establishment transaction 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).
Symmetric authorization key	Key used to provide privileges to an entity using a symmetric cryptographic method.
Private authorization key	Key is the private key of an asymmetric-key (public-key) key pair that is used to prove the owner's right to privileges (e.g., using a digital signature).
Public authorization key	Key is the public key of an asymmetric key (public-key) key pair that is used to verify privileges for an entity that knows the associated private authorization key.

In general, a single key is typically required to be used for only one purpose (e.g. encryption, integrity authentication, key wrapping, random bit generation, or digital signatures).

In addition, there is other information that is specifically related to cryptographic algorithms and keys and this information (e.g. initialization vectors, shared secrets, seeds, random numbers, passwords, etc.) may need to be protected.

5.4 Important Key Management Considerations

5.4.1 Entropy and Key Space

Within cryptography, entropy is a measure of the disorder, randomness or variability in a source of data, and it is expressed as an entropy value that is between 0 and 1; the higher an entropy value is, the more unpredictable a data source is. Entropy is used to generate random numbers or keys that are essential for secure communication and encryption. Without a good source of entropy, cryptographic protocols can become vulnerable to attacks that exploit the predictability of the generated keys. It is noteworthy that humans are notoriously weak sources of entropy.

The key space (also known as key size and key length) refers to the number of bits in a key used by a cryptographic algorithm. In the case of randomly generated symmetric keys, the probability of selecting any given n-bit key should be effectively equal to 1/(2^n), which should also be the probability of selecting any other n-bit key.

Security strength is a number, typically specified in bits of strength, associated with the amount of work that is required to break a cryptographic algorithm or system. The security strength is often the same as the number of bits in the key size; however, a low entropy source could reduce the security strength significantly (e.g. an entropy value of 0.5 for a key size of 256 bits would have a security strength of not more than 128 bits).

With certain ciphers (e.g. DES and GMAC), a very small subset of keys (known as weak keys) can make the cipher behave in some undesirable ways. For these ciphers, it is important to avoid the use of weak keys and to check for them.

In general, user selected keys should not be used directly as a data encryption key but may be used as input to a key generating function to produce the encryption key. Further, keys should not be guessable by an attacker. See NIST SP-800-132[34] for recommended guidelines.

To help avoid problems with entropy sources, weak keys, and security strength, consider using a validated cryptographic module (e.g. NIST Cryptographic Module Validation Program).

5.4.2 Key Limitations

A cryptoperiod is the time span during which a specific key is authorized for use by legitimate entities or the keys for a given system will remain in effect. NIST SP 800-57 Part 1 states that a suitably defined cryptoperiod:

- Limits the amount of information that is available for cryptanalysis to reveal the key (i.e. the amount of ciphertext encrypted with the key);
- Limits the amount of exposure if a single key is compromised;
- Limits the time available for attempts to penetrate physical, procedural, and logical access mechanisms that protect a key from unauthorized disclosure;

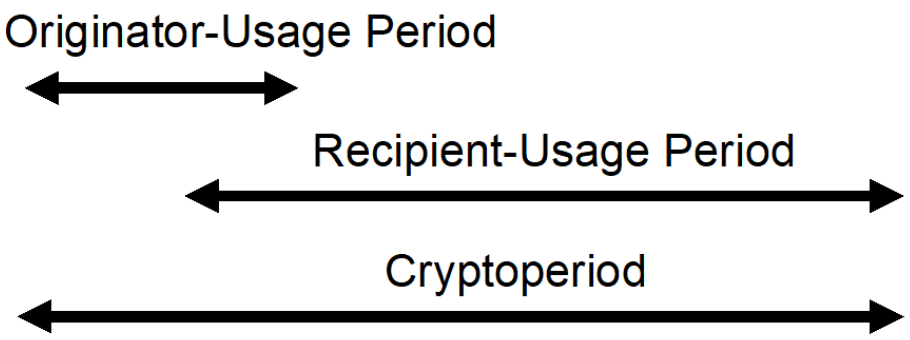
- Limits the period within which information may be compromised by inadvertent disclosure of a cryptographic key to unauthorized entities; and
- Limits the time available for computationally intensive cryptanalysis.

Sometimes, cryptoperiods are defined by an arbitrary time period or by the maximum amount of data which can be protected by the key. However, trade-offs associated with the determination of cryptoperiods involve the risk and consequences of exposure, which should be carefully considered when selecting the cryptoperiod. In general, short cryptoperiods enhance security.

Keys that are used for confidentiality protection of communication exchanges may often have shorter cryptoperiods than keys used for the protection of stored data. Cryptoperiods are generally made longer for stored data because the overhead of generating new keys and re-encrypting all data that was encrypted using the old keys may be burdensome (e.g. costs associated with changing keys used for encryption of data at rest are typically painfully high).

For symmetric keys, the period of time during which cryptographic protection may be applied to data is called the originator-usage period, and the period of time during which the protected information is processed is called the recipient-usage period (see Figure 5). The (total) “cryptoperiod” of a symmetric key is the period of time from the beginning of the originator-usage period to the end of the recipient-usage period, although the originator-usage period has historically been used as the cryptoperiod for the key.

Figure 5. Symmetric-key cryptoperiod



The originator-usage period recommended for the encryption of large volumes of data over a short period of time (e.g. for link encryption) is on the order of a day or a week. An encryption key used to encrypt smaller volumes of data might have an originator-usage period of up to two years. A recipient-usage period of no more than three years beyond the end of the originator-usage period is recommended. Where data is maintained in encrypted form, a symmetric data-encryption key needs to be maintained until that data is re-encrypted under a new key or destroyed. Note that confidence in the confidentiality of the data is reduced with the passage of time.

Rekeying data encryption keys or media encryption keys associated with significant amounts of stored data can be time consuming and potentially disruptive. As such, many organizations rekey their data at the time of technology refreshes, which typically involve migration of data. These technology refreshes commonly occur on about a three year cycle; however, in circumstances where the refresh cycle will be

longer than three years, then consideration should be given to rekeying the data where it currently resides.

5.5 Centralized Key Management using KMIP

Key management is one of the more challenging cryptography elements to implement correctly. Storage systems and ecosystems often need key management services that:

- Enforce strict controls for key generation, change, and distribution.
- Include standardized interfaces that facilitate interoperability.
- Provides the necessary redundancy in key management servers so that failure or loss of access to one (or more) can be tolerated.
- Provides key backup services.

Storage systems often use centralized key management to meet these needs. In such situations, the Key Management Interoperability Protocol (KMIP)[72] from OASIS (Organization for the Advancement of Structured Information Standards) is commonly used by storage systems to support data at rest encryption. The KMIP standards consists of the KMIP Specification[72] and the KMIP Profiles[73], which specifies conformance clauses that define the use of objects, attributes, operations, message elements and authentication methods within specific contexts of KMIP server and client interaction. Version 2.1 is the latest version of KMIP to be approved as an OASIS Standard, but work is underway on Version 3.x.

KMIP defines a protocol used for the communication between clients and servers to perform management operations on objects (typically encryption keys) stored and maintained by a key management system. It provides mechanisms for organizations to manage cryptographic keys, eliminating the need for redundant, incompatible key management processes throughout the key lifecycle — including the generation, submission, retrieval, and deletion of cryptographic keys.

Use of KMIP by storage clients is an effective way to "outsource" many of the more problematic elements (e.g. random key generation). When using KMIP or considering its use, it is important to consider the following:

- Availability of Key Materials — When a storage system is dependent on an external key management server for access to its data encryption keys, or key wrapping keys used to access those data encryption keys, it means the data on the storage system (after being powered up or otherwise reset) cannot be accessed until these keys are available; it may also be impossible to perform certain operations on the ciphertext (e.g. replication, backups, etc.). Consequently, it is important to have multiple, redundant key management servers which can be accessed to provide the necessary keys. Additionally, the storage system should block attempted user or host access to the data until the appropriate keys are available.
- Secure Transport — Encryption keys are considered sensitive information and must be protected at all times, especially when they are transmitted. The KMIP Specification requires this protection but defers the details to the KMIP Profiles; these profiles specify the use of TLS in the *Basic Authentication Suite*. KMIP Servers are required to support TLS 1.3 and should support TLS 1.2; KMIP Clients should support TLS 1.3 and support TLS 1.2; earlier version of TLS or SSL are not permitted. The Basic Authentication Suite mandates the following cipher suites for servers:

  - TLS13-CHACHA20-POLY1305-SHA256 and TLS13-AES-256-GCM-SHA384.
  - IF TLS 1.2 is supported, TLS_RSA_WITH_AES_256_CBC_SHA256 and TLS_RSA_WITH_AES_128_CBC_SHA256.

- Audit Security — When a storage system is dependent on the interactions with a KMIP Server, logging of the KMIP transactions can be critical to identifying the root cause of problems. As such, all KMIP client-to-server operations need to be logged with sufficient details that a problem can be diagnosed; and key values must never be stored or exposed in those logs. In addition, many organizations need to retain records that serve as proof-of-encryption. Some of the KMIP operations play a role in these activities, which means the appropriate event entries need to be captured in the audit logs.
- KMIP Server Compatibility — An important motivation for using KMIP-based key management is that conforming KMIP Clients can use key management servers from a variety of vendors (e.g. Cryptsoft, HP, IBM, Quintessence Labs, Thales e-Security, and Townsend Security). KMIP Client implementations must be carefully designed to avoid accidentally constraining compatibility (and possibly conformance) due to incorrect interpretations of the KMIP Specification.

5.6 Recovery plan

Recovery of encrypted data requires both the encrypted data and the key(s) needed to decrypt that data. As a result, keys need to be distributed to allow redundancy, but must be exchanged in a secure fashion to avoid compromise or corruption of the key. Unintended alteration of the key will also cause a loss of access to the data, so the key management system must provide redundancy and disaster recovery mechanisms for the keys. Recovery plans must also be available in the event that a key is compromised.

6 Other Encryption and Key Management Issues

6.1 Data Eradication on Storage

6.1.1 Storage Sanitization

Storage sanitization refers to the general process of rendering access to target data on storage infeasible for a given level of effort.

Depending on the type of storage (logical versus media-based), this storage sanitization can take the form of logical sanitization or media-based sanitization.

Storage sanitization is typically an element of an organization's data governance process. The decision to use storage sanitization should be based on the organization's data classification scheme and, focusing on the data that are classified as sensitive. Sensitive data in this context is data for which disclosure can have an impact on organizational mission, result in damage to organizational assets, or result in financial loss or harm to the organization or individuals.

Per ISO/IEC 27040[17] and IEEE 2883[48], multiple sanitization methods can be used, depending on the storage (logical or media-based), and they take the form of:

- Clear - Involves the use of software or hardware methods to replace target data with non-sensitive data.
- Purge – Involves the use of physical or logical techniques that make recovery infeasible, even by use of state of the art laboratory techniques, while preserving the storage in a potentially reusable state.
- Destruct - Involves the use of physical techniques to destroy the storage. This sanitization method is not applicable to logical storage.

In an effort to reduce the amount of storage destined for landfills, industry efforts are underway to adopt purge methods (as opposed to destruct) as the preferred storage sanitization method. Of the options for the purge method, cryptographic erase is a particularly desirable technique.

6.1.2 Cryptographic Erase

Conceptually, cryptographic erase leverages the encryption of data and the eradication of the encryption key used to encrypt the data. This leaves only the ciphertext remaining in the storage, effectively sanitizing the data. To be considered a purge sanitization method, ISO/IEC 27040 stipulates that the following conditions must be met, at a minimum:

- all data intended for cryptographic erase must be encrypted prior to recording on the storage media;
- the strength of the cryptographic algorithm (including mode of operation) used to encrypt the target data must be at least 128 bits;
- the bits of entropy must be at least the number of bits used by the encryption key which is used to encrypt the target data;
- all copies of the encryption keys used to encrypt the target data must be eradicated; if the data's encryption keys are, themselves, encrypted with one or more wrapping keys, it is acceptable to perform cryptographic erase by eradicating a corresponding wrapping key.

The quality of the cryptography used in cryptographic erase mechanism is of concern because weak or flawed implementations can result in recovery of data. Therefore, special attention should be focused on:

- *Key generation:* the original encryption keys had been generated from a validated random bit generator.
- *Media encryption:* the security strength and validity of implementation of the encryption algorithm/mode used for protection of the target data.
- *Key level and key wrapping:* determining whether the media encryption key/data encryption or a key used to wrap (that is, encrypt) the media encryption key is being eradicated. In the latter case, the security strength of the wrapping techniques used should be commensurate with the level of strength of the media encryption key.

When deciding whether to rely upon cryptographic erase, it should also be considered whether the encryption keys can be recovered either internally or externally (e.g. injected from a key management

server or from a key escrow service). If the encryption key (or any key at or below the level[5] of the key eradicated during cryptographic erase) exists outside of the storage, there is a possibility that the key can be used in the future to recover data stored on the encrypted storage.

It should be noted that inadvertent loss of the cryptographic keys can produce the same result as performing a cryptographic erase of the data. An inadvertent loss of control which permits untracked copies to be made of cryptographic keys may make cryptographic erase impossible.

6.2 Legal/Regulatory Compliance

6.2.1 General

Compliance aspects of storage and key management systems that would be of concern in an audit include accountability, traceability, sanitization, privacy, legal, detection, and monitoring requirements. While many of these processes may be in place for the overall computing infrastructure, it is important to extend audit logging to the storage layer. This means maintaining a secure audit log for such events as data encryption, decryption, or destruction of data, and the creation, deletion, and use of keys. Sufficient information must be collected so that the source and a specific individual making such changes can be identified.

6.2.2 Import/Export Controls

It is important to understand and comply with government regulations for both the import and export of encryption technologies between various countries. Frequently, such regulations prohibit the import of a high speed strong encryption means into a country. Likewise, strong encryption technologies where both the clear text data and the ciphertext data can be viewed are considered general purpose encryption devices that usually face export restrictions. These issues are complicated by differing trade agreements that have been constructed with different governments, and usually apply to both data encryption and key management equipment.

Additionally, key escrow may be required either by governmental or corporate requirements. This is an arrangement where keys needed to decrypt encrypted data are held in escrow so that, under certain circumstances, an authorized third party may gain access to those keys. Note that key escrow may be required for both encryption of data in motion and encryption of data at rest.

As an example, a great deal of practical information about United States regulations related to the export of encrypting technologies and equipment can be found at the US Department of Commerce's *Bureau of Industry and Security* website: https://www.bis.doc.gov/index.php/policy-guidance/encryption.

[5] As an example, eradicating a key wrapping key that is used to encrypt a media encryption key, where the media encryption key had been previously escrowed is not likely to produce the desired result because the media encryption key may still be available.

6.2.3 Safe Harbors

Data breaches have become enough of a concern that many jurisdictions require notifications whenever security breaches/incidents associated with certain types of data occur. What is considered a security breach and the covered types of data can vary significantly.

To further incentivize the adoption of security best practices to prevent data breaches in the first place, some jurisdictions have included “safe harbor” provisions in their laws/regulations to relieve an organization from the expense and humiliation of having to send out breach notifications, if appropriate security measures are in use. To use a safe harbor to avoid triggering notifications after a security incident, the breached organization must prove that it secured (typically encrypted) the sensitive data in accordance with the jurisdictional requirements.

Unfortunately, the safe harbor requirements, definition of encryption, definition of breach, and the types of sensitive data that are relevant are not consistent across jurisdictions. In addition, the laws and regulations in this space are changing all the time. Current trends indicate a change from a blanket safe harbor when encryption is employed to one of “reasonable” or risk-based security.

6.2.4 Proof of Operations

In general, organizations are expected to maintain records of their data at rest encryption to identify the storage media that were protected, as well as when and how they were encrypted. When an organization is suspected of losing control of its storage media, which contain sensitive data, these records or proof of encryption can be instrumental in demonstrating that no data breach occurred, thereby avoiding costly data breach notifications and other liabilities. Such proof can take the form of appropriate audit log entries (e.g. activation, rekeying, verification, etc.).

As with proof of encryption, organizations are expected to maintain records of their storage sanitization activities to document which storage media were sanitized, when and how they were sanitized, and the final disposition of the storage media. Such record keeping applies to use of cryptographic erasure (see 6.1.2). Often when an organization is suspected of losing control of its information, it is because of inadequate record keeping of storage media sanitization.

6.2.5 Data Retention and Preservation

Organizations that have data retention and preservation obligations need to store data in a manner that blocks records destruction or alteration (i.e. immutable) along with integrity verification (e.g. hashing) and enforcement of explicit retention periods (e.g. legal holds) that need to be honored. To meet immutability (non-editable) requirements, organizations can use write once read many (WORM)-based storage or object-based storage implementations that combine WORM with metadata that can be used to perform explicit integrity checks as well as enforce data expirations.

The use of data at rest encryption can complicate meeting these obligations. In particular, organizations need to ensure that the associated encryption keys as well as the ciphertext are handled correctly. The loss of the keys effectively sanitizes the data (6.1) and may cause a failure in meeting the data retention and preservation obligations.

6.3 Security Certifications Relevant to Storage

6.3.1 General

The implementation of security features and capabilities within storage systems is of increasing importance to organizations around the world. Many of these organization (e.g. government, financial, energy, telecommunications, and other markets) seek assurances that storage technology has passed rigorous testing by an accredited lab, that the test results have been validated, and that the product can be used to secure sensitive information.

Formal certification schemes with established security evaluation criteria have been establish for cryptographic modules (see 6.3.2) as well as for the entire suite of security capabilities of product (see 6.3.3).

6.3.2 Crypto Module Certification

International cryptographic module certifications (e.g., Japan, Malaysia, Spain, Turkey, and Korea) are possible based on some or all the following:

- ISO/IEC 19790:2012[11], which specifies requirements, based on a draft version of FIPS 140-3[20] (final version of FIPS 140-3 was published in May 2019).
- ISO/IEC 24759:2017[16], which specifies testing requirements for demonstrating the conformity to the requirements specified in ISO/IEC 19790:2012.
- ISO/IEC 18367:2016[10], which provides guidelines for cryptographic algorithms and security mechanisms conformance testing methods.
- ISO/IEC TS 20540:2018[13], which provides recommendations and checklists which can be used to support the specification and operational testing of cryptographic modules in their operational environment within an organization's security system.
- ISO/IEC 20543:2019[14], which specifies a methodology for the evaluation of random bit generators intended to be used for cryptographic applications.

Note that evaluation laboratories may be subject to conformance to ISO/IEC 19896[12] (Competency requirements for information security testers and evaluators).

ISO/IEC 19790 specifies four cryptographic module security levels (from 1 as the lowest to 4 as the highest):

- Level 1: Baseline level of security for production-grade equipment and externally tested algorithms.
- Level 2: Adds requirements for physical tamper-evidence and role-based authentication; this is the highest security level attainable by a pure software module.
- Level 3: Adds requirements for physical tamper-resistance and identity-based authentication. Physical or logical separation between the interfaces by which “critical security parameters” enter and leave the module are required. Private keys can only enter or leave the module in encrypted form. The module is required to detect and react to out-of-range voltage or temperature (environmental failure protection, or EFP), or alternatively undergo environmental failure testing (EFT).

- Level 4: Adds physical security requirements that include the ability to be tamper-active, erasing the contents of the device if it detects various forms of environmental attack. EFP and protection against fault injection is required as well as multi-factor authentication.

Table 5 shows the 11 requirements areas identified in ISO/IEC 19790.

Table 5. ISO/IEC 19790 Requirements Areas

Cryptographic module specification
Cryptographic module interfaces
Roles, services, and authentication
Software/Firmware security
Operational environment
Physical security
Non-invasive security
Sensitive security parameter management
Self-tests
Life-cycle assurance
Mitigation of other attacks

The specific security requirements in the areas can change, depending on the security level.

The NIST FIPS 140-3[20] is a US government standard that defines minimum security requirements for cryptographic modules in information technology products and systems. FIPS 140-3 is based on ISO/IEC 19790:2012[11] and ISO/IEC 24759:2017[16]. As such, it also includes 4 levels and 11 security requirements areas.

Testing against the FIPS 140-3 standard is maintained by the Cryptographic Module Validation Program (CMVP), which is a joint effort between the NIST and the Canadian Centre for Cyber Security, a branch of the Communications Security Establishment (CSE) of Canada.

The CMVP manages the variances allowed in the ISO/IEC 19790 and ISO/IEC 24759 through the NIST SP 800-140x documents. Specifically, the NIST SP 800-140[35] provides additional evidence and testing that is necessary to meet CMVP cryptographic module requirement evidence, while also providing to ISO/IEC recommended adjustments to the existing standard when next reviewed. The remaining NIST SP 800-140A through NIST SP 800-140F[36][37][38][39][40][41] provide additional requirements for vendor evidence, security policy, approved encryption and key management, authentication and non-invasive physical security requirements.

Clarification or interpretation of the requirements and assurance measures are included in the Implementation Guidance, which are often technology specific. A standalone document, the FIPS 140-3 CMVP management manual, addresses the programmatic procedures and requirements of the process. In addition, National Voluntary Laboratory Accreditation Program (NVLAP)Handbook 150-17 identifies CMVP specific NVLAP requirements which includes requirements in quality systems, personnel,

environmental conditions, test and calibration methods, equipment, test quality assurance, and reporting results control.

6.3.3 Product Certification

The Common Criteria for Information Technology Security Evaluation (referred to as Common Criteria or CC) is a product-level security certification. The current version is CC:2022, which is the first major revision since being published as CC v3.1 Revision 5 in 2017. The CC standard along with the Common Evaluation Methodology (CEM:2022) were developed and are maintained by the participating nations of the Agreement on the Recognition of Common Criteria Certificates in the field of IT Security, which is also known as the Common Criteria Recognition Arrangement (CCRA); the standards are published by ISO/IEC as:

- ISO/IEC 15408-1:2022, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 1: Introduction and general model.
- ISO/IEC 15408-2:2022, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 2: Security functional components.
- ISO/IEC 15408-3:2022, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 3: Security assurance components.
- ISO/IEC 15408-4:2022, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 4: Framework for the specification of evaluation methods and activities.
- ISO/IEC 15408-5:2022, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 5: Pre-defined packages of security requirements.
- ISO/IEC 18045:2022, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Methodology for IT security evaluation.

Vendors can subject their product implementations and claims about the security attributes of their products, contained in a Security Target (ST) that identifies the Target of Evaluation (TOE), to accredited testing laboratories that evaluate the products to determine if they actually meet the claims; the results are provided to a certification body, which makes the determination as to whether to issue a CC certification.

All CC evaluations are conducted against a chosen Evaluation Assurance Level (EAL). The EAL level describes the depth and rigor of an evaluation as the EALs have specific requirements laid out by CC. The following are CC-specified levels:

- EAL1: Functionally Tested.
- EAL2: Structurally Tested.
- EAL3: Methodically Tested and Checked.
- EAL4: Methodically Designed, Tested and Reviewed.
- EAL5: Semi-Formally Designed and Tested.
- EAL6: Semi-Formally Verified Design and Tested.

- EAL7: Formally Verified Design and Tested.

While each EAL level includes specific Security Assurance Requirements (SARs) and Security Functional Requirements (SFRs), a specific product evaluation may include additional SARs and/or SFRs that are reflected in the certification (e.g., EAL2+ ATE_COV.3).

To help ensure the use of a consistent set of security requirements for a certain type of product and/or across multiple jurisdictions, security requirements can be specified in one or more protection profile (PP) or a collaborative protection profile (cPP). Products to be evaluated can reference relevant PPs and cPPs.

CC certification does not guarantee a secure product, but it can ensure that claims about the security attributes of the evaluated product were independently verified.

Table 6 list PPs and cPPs that may be relevant to storage oriented products.

Table 6. PP and cPP Relevant to Storage

PP/cPP	CC Version	Assurance Level	Issued	Certified
collaborative Protection Profile for Full Drive Encryption - Encryption Engine v2.0 + Errata 20190201	3.1R5	None	2019-02-01	Yes
collaborative Protection Profile for Full Drive Encryption - Authorization Acquisition v2.0 + Errata 20190201	3.1R5	None	2019-02-01	Yes
collaborative Protection Profile for Full Drive Encryption – Authorization Acquisition v1.0	3.1R4	None	2015-02-27	Yes
collaborative Protection Profile for Full Drive Encryption - Encryption Engine v1.0	3.1R4	None	2015-02-27	Yes
collaborative Protection Profile for USB Portable Storage Devices	3.1R5	None	2020-11-10	No
collaborative Protection Profile Module for Full Drive Encryption – Enterprise Management	3.1R5	None	2018-03-23	No
Protection Profile - Encrypted Storage Device	3.1R3	EAL2+ ATE_COV.3	2012-04-26	Yes
File Encryption Protection Profile	3.1R5	EAL3 ALC_FLR.2	2018-07-04	Yes
PP-Module for File Encryption Version 1.0 Supporting Document		EAL1	2019-07-25	Yes

During the transition to CC:2022, CCV3.1R5 may optionally be used for evaluation starting no later than 2024-06-30. Security Targets conformant to CC:2022 based on PPs certified according to CC3.1 will be accepted up to 2027-12-31. Assurance continuity activities (maintenance, re-evaluation and re-

assessment) based on CC 3.1 evaluations can be started for up to 2 years from the initial certification date.

7 Trends in Storage Encryption and Key Management

7.1 Post-Quantum Cryptography (PQC)

7.1.1 General

A sufficiently powerful future quantum computer has the potential to easily break every form of legacy asymmetric cryptography the IT industry relies on today (RSA, ECC, DH, etc.) that are based on either the Factoring Problem or the (Elliptical Curve) Discrete Logarithm Problem, which will be efficiently solvable by a future quantum computer using Shor's algorithm. In 2016, responding to this threat (it should be noted that the "harvest now & decrypt later" attack is a threat to be concerned about today), NIST began a Post-Quantum Cryptography (PQC) standardization project. On July 5, 2022, NIST announced it would pursue standardization of four of the algorithms that survived the third round of the PQC project. On August 24, 2023, NIST made draft standard versions available of the first three of those four (i.e. for all of the four other than FALCON) Quantum-Resistant Cryptographic Algorithms it would pursue standardization of, as a first result of that project:

- For general encryption, NIST selected a quantum-resistant Key Encapsulation Method (KEM) that can be used to encapsulate a shared secret using asymmetric keys:
  1. CRYSTALS-Kyber[23].

A KEM can be used, for example, by two endpoints to establish a shared secret symmetric key to enable symmetric encryption of data-in-motion (e.g. a future version of TLS). It can also be used to negotiate symmetric encryption keys using a key management framework (e.g. KMIP) to support encryption of data-at-rest.

- For digital signatures, NIST has selected three quantum-resistant algorithms with different properties:
  1. CRYSTALS-Dilithium (primary), referred to as the Module Lattice Digital Signature Algorithm (ML-DSA)[24].
  2. FALCON.
  3. SPHINCS+ (read as "Sphinx plus"), referred to as the Stateless Hash-based Digital Signature Algorithm (SLH-DSA)[25].

NIST is expected to complete standardization of these four algorithms sometime in 2024. Even so, NIST has announced it is not yet done with its Post-Quantum Cryptography standardization project – it is entering the fourth round of that project to evaluate four additional alternate (to the four specified above) quantum-resistant algorithms. It is possible that sometime in the future NIST will announce that it will also pursue standardization of some subset of those four additional alternate algorithms based on the result of the fourth-round vetting. NIST also issued a call for additional signature schemes with shorter signature sizes and fast verification. In total, 40 submissions were published in July 2023, the standardization process for the additional signature schemes is expected to take multiple years.

A sufficiently powerful future quantum computer also has the potential to undermine the strength of symmetric encryption using Grover's quantum algorithm. For symmetric key encryption the near-term recommendation to address this threat is to double the key size (e.g. use AES-256 instead of AES-128).

7.1.2 Hybrid Use of PQC algorithms

Hybrid use of PQC algorithms has the PQC algorithms being used in conjunction with legacy asymmetric algorithms in such a way that the resultant hybrid algorithm is only as weak as the stronger of the two underlying algorithms. As an example, hybrid implementations of key establishment exist, as do hybrid implementations of digital signatures. NIST SP 800-56C Rev. 2 allows for "hybrid" key-establishment usage by FIPS modules. ANSSI (France), BSI (Germany), and GCHQ (United Kingdom) recommend the use of hybrid cryptography in high security applications – to hedge against the case one of the two underlying algorithms is broken.

SNIA's position is to recommend hybrid use of PQC algorithms, wherever PQC algorithms are to be used. This is recommended not only now, before NIST has completed standardization of the PQC algorithms it has announced it will standardize have in fact been standardized, but also afterwards – at least until use of a cryptographically relevant quantum computer is readily accessible, after which use of legacy asymmetric cryptographic algorithms will cease to add value as a hedge.

7.1.3 Stateful Hash-Based Signature (HBS) scheme

As stated in 7.1.1, all current commonly used digital signature schemes will be broken if large scale quantum computers are ever built. While standards for post-quantum secure digital signature schemes are being developed, stateful hash-based signature (HBS) schemes described in NIST SP 800-208[43] may offer secure alternatives for applications with the following characteristics:

- it is necessary to implement a digital signature scheme in the near future;
- the implementation will have a long lifetime; and
- it would not be practical to transition to a different digital signature scheme once the implementation has been deployed.

In a stateful HBS scheme, an HBS private key consists of a large set of one-time signature (OTS) private keys. The signer needs to ensure that no individual OTS key is ever used to sign more than one message. The HBS scheme is secure so long as the OTS key is never reused. HBS is intended for systems and applications that have constraints to replace the signatures or the signing scheme, once the product is deployed. For example, products with embedded firmware or software elements with implementation having a long lifespan; and it would not be practical to transition to a different digital signature scheme after the product is deployed.

NIST SP 800-208 proposes two schemes based on Merkle trees:

1. eXtended Merkle Signature Scheme (XMSS): RFC 8391[69].
2. Leighton-Micali Signature (LMS): RFC 8554[71].

NSA recommends Leighton-Micali with SHA-256/192[75], but all algorithms specified in NIST SP 800-208 are approved for software/firmware signing use case.

Timeline: NSA encourages vendors to begin adopting NIST SP 800-208 signatures immediately. The Commercial National Security Algorithm (CNSA), Suite 2.0[75] gives a proposed timeline for adoption of other PQC standards as well.

7.2 Changing Algorithms

As the threat landscape evolves, adjustments to encryption and key management become necessary. These changes can result in new classes of algorithms, deprecation of algorithms, and modification to practices.

At the time of this writing, the following are under consideration:

- Update to XTS-AES mode block cipher algorithm

  NIST is planning to update SP800-38E[29] using the XTS-AES Mode for Confidentiality on storage devices. The updated publication will mention the security vulnerability that results when the two AES (sub)keys are improperly generated to be identical, as discussed in Annex C.I of Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program. For additional information, see https://csrc.nist.gov/publications/detail/sp/800-38e/final.

- Revision to FIPS 180-4[21], Secure Hash Standard

  NIST has decided to revise FIPS 180-4 and will revise the text to:

  - Remove the SHA-1 specification.
  - Add any guidance from NIST SP 800-107[31].

  For additional information, see https://csrc.nist.gov/news/2023/decision-to-revise-fips-180-4.

- SHA-1 transition

  NIST is introducing a plan to transition away from the current limited use of the Secure Hash Algorithm 1 (SHA-1) hash function. Other approved hash functions are already available. The transition will be completed by December 31, 2030, and NIST will engage with stakeholders throughout the transition process. The plan includes the following deliverables:

  - Publish FIPS 180-5 (a revision of FIPS 180) to remove the SHA-1 specification,
  - Revise SP 800-131A[33] and other affected NIST publications to reflect the planned withdrawal of SHA-1, and
  - Create and publish a transition strategy for the Cryptographic Module Validation Program (CMVP) and the Cryptographic Algorithm Validation Program (CAVP).

  For additional information, see https://csrc.nist.gov/news/2022/nist-transitioning-away-from-sha-1-for-all-apps.

- Proposal to Revise Password-Based Key Derivation Function 2 (PBKDF2)

  NIST proposes to revise SP 800-132 with following objectives:

  - to approve an additional memory-hard password-based key derivation function and password hashing scheme, and
  - to provide additional guidelines and clarifications on the use of PBKDF2.

7.3 Privacy Preserving Computing Technologies

7.3.1 Encryption of Data in Use

There is increased interest in encrypting data in use, as witnessed through the formation of the Confidential Computing Consortium and investigations within ISO/IEC JTC 1/SC 27 on potential standardization. A significant focus of these activities is on Trusted Execution Environments (TEEs), which provide secure computation capability through a combination of special-purpose hardware in modern processors and software (firmware) built to use those hardware features. In general, the special-purpose hardware provides a mechanism by which a process can run on a processor without its memory or execution state being visible to any other process on the processor, even the operating system or other privileged code. Thus, the TEE approach provides Input Privacy.

Computation in a TEE is not performed on data while it remains encrypted. Instead, the execution environment is made secure by the special hardware provided. Such a protected execution environment is often termed an enclave. Typically, the memory space of each enclave application is protected from access while resident on the processor chip, and then encrypted (typically using some mode of AES encryption) when and if it is stored off-chip. Registers and other processor-local state of the enclave are protected from access. Code entry and exit points are tightly controlled, so that execution cannot easily switch between the enclave and the unprotected application that envelops it.

Another significant feature of enclaves is that other processes (whether local or remote) that must trust an enclave can receive attestation that the enclave is genuine, and that the code running in it (and in fact the static parts of its memory space) are exactly what is expected. Such attestation is guaranteed using cryptographic capabilities such as digital signatures and hash functions. Enclaves can enable Output Privacy (published results do not contain identifiable input data beyond what is allowable by input parties who are responsible for protecting the data) and access control when the attested code includes specific computations that provide those features.

Virtualized TEEs on hosts may be used to isolate workloads and to securely access one or more devices. The PCI-SIG has specified the TEE Device Interface Security Protocol (TDISP)[82] for just such a purpose.

7.3.2 Homomorphic Encryption

Homomorphic encryption refers to a family of encryption schemes with a special algebraic structure that allows computations to be performed directly on public-key encrypted data without requiring that the data be decrypted first. The most capable form of homomorphic encryption is known as Fully Homomorphic Encryption (FHE). In principle, FHE can be used to perform arbitrary Boolean and arithmetic computations on encrypted data without revealing the cleartext form of either the input data or the result of the computation to the party that performs the FHE computation. Instead, the result can only be decrypted by a party that has access to the associated private key (which is typically the owner of the input data). But, even though an attacker without access to the private key cannot decrypt the FHE ciphertext, if the attacker has access to not only the FHE ciphertext, but also to the “public” evaluation key needed to do FHE computations, the attacker can potentially manipulate that ciphertext (which is to say that FHE encrypted data is malleable). Where such an attack is feasible, additional measures (such as use of an authentication tag, signature, etc.) must additionally be used to protect the FHE ciphertext, to detect any malicious manipulation of it.

FHE's functionality makes it a powerful technology upon which cryptographically-secure cloud storage and computation services can be built. FHE can also be used as a building block for higher-level cryptographic primitives and protocols that rely on such functionality.

All known forms of FHE build on lattice-based cryptography, just as the PQC algorithms CRYSTALS Kyber and CRYSTALS Dilithium do. And, just as the Kyber and Dilithium algorithms are, FHE encryption is considered to be quantum-resistant.

Note that computational storage, which is capable of not only storage, but also of computation on the data stored there, could potentially perform computations on homomorphically encrypted data, if given the public evaluation key needed to do such computations.

7.4 Dual Layer Encryption

In some environments (e.g. National Security Agency Commercial Solutions for Classified Data-at-Rest Capability Packages[76]), data at rest solutions are required to use multiple layers of approved encryption. Such requirements are not because of a deficiency in the cryptographic algorithms, but to mitigate the risk of a failure in one of the cryptographic components due to accidental misconfiguration, operator error, or malicious exploitation of an implementation vulnerability, which results in the exposure of sensitive information.

There can be additional data at rest encryption requirements such as requiring solutions to:

- come from different manufacturers, where neither manufacturer is a subsidiary of the other;
- be different products from the same manufacturer, where it has been determined that the products meet the criteria for implementation independence;
- that have cryptographic libraries used by the "Inner" and "Outer" data at rest layers that are independently developed and implemented.

Such multi-layer encryption approaches/strategies can help reduce vendor lock-in as well as to facilitate cryptographic agility (see 7.1.2).

8 Summary

While encryption and key management capabilities are pervasive within storage systems and ecosystems, there are many issues and considerations to be navigated to realize the full value of these capabilities. It is also important to understand the role storage-based encryption and key management capabilities can play in an organization's data protection program. This SNIA technical paper provides important details that will help organizations with their storage security implementations.

Encryption and key management technologies and practices are not static. Likewise, storage technologies continue to evolve. This technical paper explores a range of encryption and key management issues and considerations that are anticipated to have impacts on future data protection implementations and strategies.

Lastly, this technical paper identifies relevant standards and specifications for the use of encryption and key management within storage systems and ecosystems. In the case of ISO/IEC 27040, this technical paper is intended to provide supplementary information that can help an organization better understand the controls and guidance in the standard.

9 Abbreviations

Abbreviations used in this paper:

3DES	Triple Data Encryption Standard
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Algorithm
AES-KW	AES Key Wrap
CA	Certificate Authority
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CC	Common Criteria
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CCRA	Common Criteria Recognition Arrangement
CEM	Common Evaluation Methodology
CFB	Ciphertext Feedback
CMC	CBC-Mask-CBC
CMVP	Cryptographic Module Validation Program
CN	Container
cPP	Collaborative Protection Profile
CRQC	Cryptographically Relevant Quantum Computer
CSC	Cloud Service Customer
CSE	Communications Security Establishment
CSP	Cloud Service Provider
CTR	Counter
CXL™	Compute Express Link™
DH	Diffie-Hellman
DHE	Diffie-Hellman Ephemeral
DOE	Data Object Exchange
DSA	Digital Signature Algorithm
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
EFP	Environmental Failure Protection
EFT	Environmental Failure Testing
EME	Encrypt Mix Encrypt
ESP	Encapsulating Security Payload
FC	Fibre Channel
FC-SP	Fibre Channel - Security Protocols
FDE	Full Disk Encryption
FHE	Fully Homomorphic Encryption
FIPS	Federal Information Processing Standard

GCM	Galois Counter Mode
GUID	Globally Unique Identifier
HBA	Host Bus Adapter
HBS	Stateful Hash-Based Signature
HMAC	Keyed-Hash Message Authentication Codes
IaaS	Infrastructure as a Service
IDE	Integrity and Data Encryption
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	IP Security
iSCSI	Internet Small Computer System Interface
IV	Initialization Vector
KDF	Key Derivation Function
KEK	Key Encryption Key
KEM	Key Encapsulation Method
KMIP	Key Management Interoperability Protocol
KPIO	Key Per I/O
LBA	Logical Block Address
LMS	Leighton-Micali Signature
LRW	Liskov, Rivest, Wagner
LTO	Linear Tape-Open
MAC	Network Attached Storage
MEK	Media Encryption Key
ML-DSA	Module Lattice Digital Signature Algorithm
NAS	Message Authentication Code
NIC	Network Interface Controller
NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
NVMe™	NVM Express™
OASIS	Organization for the Advancement of Structured Information Standards
OCP	Open Compute Project
OFB	Output Feedback
OTS	One-Time Signature
PaaS	Platform as a Service
PBE	Pre-Boot Environment
PBKDF	Password-Based Key Derivation Function
PCI-SIG	Peripheral Component Interconnect Special Interest Group
PCIe™	PCI Express™
PFS	Perfect Forward Secrecy
PoE	Point of Encryption
PP	Protection Profile
PQC	Post-Quantum Cryptography

PSID	Physical Secure ID
PSK	Pre-Shared Key
RA	Registration Authority
RFC	Request For Comment
RGB	Random Bit Generator
RSA	Rivest-Shamir-Adleman
RSAES-OAEP	RSA Encryption System-Optimal Asymmetric Encryption Padding
RSAES-PKCS	RSA Encryption System- Public-Key Cryptography Standard
SaaS	Software as a Service
SAN	Storage Area Network
SAR	Security Assurance Requirement
SAS	Serial Small Computer System Interface
SED	Self-Encrypting Drive
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SID	Secure ID
SIIS	Storage Interface Interaction Specification
SLH-DSA	Stateless Hash-based Digital Signature Algorithm
SP	Security Partition
SPDM	Secure Protocol and Data Model
SSC	Security Subsystem Class
SSD	Solid-State Drive
SSH	Secure Shell
SSL	Secure Sockets Layer
ST	Security Target
TCG	Trusted Computing Group
TCP/IP	Transmission Control Protocol/Internet Protocol
TDISP	TEE Device Interface Security Protocol
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TOE	Target of Evaluation
VM	Virtual Maching
WORM	Write Once Read Many
XCB	Extended Codebook
XEX	Xor–Encrypt–Xor
XMSS	eXtended Merkle Signature Scheme
XTS	XEX Tweakable Block Ciphertext Stealing

10 Acknowledgments

10.1 About the Author

Eric A. Hibbard is the Director, Product Planning – Security at Samsung Semiconductor, Inc. and a cybersecurity and privacy leader with extensive experience in industry (PrivSec Consulting LLC, Hitachi, Raytheon, Hughes, OAO Corp), U.S. Government (NASA, DoE, DoD), and academia (University of California). Mr. Hibbard holds leadership positions in standards development organization and industry associations, including ISO/IEC, INCITS, IEEE, SNIA, ABA, and CSA. He has also served as editor of ISO/IEC 27040, ISO/IEC 27050 series, ISO/IEC 22123 series, and IEEE 1619-2018.

Mr. Hibbard possesses a unique set of professional credentials that include the (ISC)2 CISSP-ISSAP, ISSMP, and ISSEP certifications; IAPP FIP, CIPP/US and CIPT certifications; ISACA CISA and CDPSE certifications; and CSA CCSK certification. He has a BS in Computer Science. Learn more at https://www.linkedin.com/in/ericahibbard/.

10.2 Reviewers and Contributors

The SNIA Security Technical Work Group (TWG) wishes to thank the following for their contributions to this technical paper:

Glen Jaquette, IBM

Thomas Rivera, VMware, Inc.

Paul Suhler, Kioxia Corporation

Mark Carlson, Kioxia Corporation

John Geldman, Kioxia Corporation

James Borden, Kioxia Corporation

Sridhar Balasubramanian, NetApp

Tim Chevalier, NetApp

Jim Hatfield

Gary Sutphin

Richard Austin

Dan Helmick, Samsung Semiconductor Inc.

Tim Jones, Samsung Semiconductor Inc.

Rajesh Koul, Samsung Semiconductor Inc.

Bibliography

[1] ISO/IEC 10116:2017, *Information technology — Security techniques — Modes of operation for an n-bit block cipher*

[2] ISO/IEC 15408-1:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 1: Introduction and general model*

[3] ISO/IEC 15408-2:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 2: Security functional components*

[4] ISO/IEC 15408-3:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 3: Security assurance components*

[5] ISO/IEC 15408-4:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 4: Framework for the specification of evaluation methods and activities*

[6] ISO/IEC 15408-5:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 5: Pre-defined packages of security requirements*

[7] ISO/IEC 11770-1:2010, *Information technology -- Security techniques -- Key management -- Part 1: Framework*

[8] ISO/IEC 18033-3:2010, *Information security — Security techniques — Encryption algorithms — Part 3: Block ciphers*

[9] ISO/IEC 18045:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Methodology for IT security evaluation*

[10] ISO/IEC 18367, *Information technology — Security techniques — Cryptographic algorithms and security mechanisms conformance testing*

[11] ISO/IEC 19790, *Information technology – Security techniques – Security requirements for cryptographic modules*

[12] ISO/IEC 19896 (all parts), *IT security techniques — Competence requirements for information security testers and evaluators*

[13] ISO/IEC 20540, *Information technology — Security techniques — Testing cryptographic modules in their operational environment*

[14] ISO/IEC 20543, *Information technology — Security techniques — Test and analysis methods for random bit generators within ISO/IEC 19790 and ISO/IEC 15408*

[15] ISO/IEC 20648, *TLS specifications for storage systems*

[16] ISO/IEC 24759, *Information technology – Security techniques – Test requirements for cryptographic modules*

[17] ISO/IEC 27040, *Information technology – Security techniques – Storage security*

[18] ANSI INCITS 545–2019, *Information Technology — Fibre Channel — Framing and Signaling-4 (FC-FS-5)*

[19] ANSI INCITS 496-2012, *Information Technology - Fibre Channel - Security Protocols - 2 (FC-SP-2)*

[20] NIST Federal Information Processing Standards (FIPS) Publications 140-3 (FIPS PUBS 140-3), *Security Requirements for Cryptographic Modules*

[21] NIST Federal Information Processing Standards (FIPS) Publications 180-4 (FIPS PUBS 180-4), *Secure Hash Standard (SHS)*

[22] NIST Federal Information Processing Standards (FIPS) Publications 197 (FIPS PUBS 197), *Advanced Encryption Standards (AES)*

[23] NIST draft Federal Information Processing Standards 203 (FIPS 203), *Module-Lattice-based Key-Encapsulation Mechanism Standard*

[24] NIST draft Federal Information Processing Standards 204 (FIPS 204), *Module-Lattice-Based Digital Signature Standard*
[25] NIST draft Federal Information Processing Standards 205 (FIPS 205), *Stateless Hash-Based Digital Signature Standard*
[26] NIST Special Publication (SP) 800-38A, *Recommendation for Block Cipher Modes of Operation*
[27] NIST Special Publication (SP) 800-38C, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*
[28] NIST Special Publication (SP) 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*
[29] NIST Special Publication (SP) 800-38E, *Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices*
[30] NIST Special Publication (SP) 800-57 Part 1-3, *Recommendation for Key Management*
[31] NIST Special Publication (SP) 800-107, *Recommendation for Applications Using Approved Hash Algorithms*
[32] NIST Special Publication (SP) 800-111, *Guide to Storage Encryption Technologies for End User Devices*
[33] NIST Special Publication (SP) 800-131A, *Transitioning the Use of Cryptographic Algorithms and Key Lengths*
[34] NIST Special Publication (SP) 800-132, *Recommendation for Password-Based Key Derivation Part 1: Storage Applications*
[35] NIST Special Publication (SP) 800-140, *Derived Test Requirements (DTR): CMVP Validation Authority Updates to ISO/IEC 24759*
[36] NIST Special Publication (SP) 800-140A, *CMVP Documentation Requirements: CMVP Validation Authority Updates to ISO/IEC 24759*
[37] NIST Special Publication (SP) 800-140B, *CMVP Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B*
[38] NIST Special Publication (SP) 800-140C, *CMVP Approved Security Functions: CMVP Validation Authority Updates to ISO/IEC 24759*
[39] NIST Special Publication (SP) 800-140D, *CMVP Approved Sensitive Parameter Generation and Establishment Methods: CMVP Validation Authority Updates to ISO/IEC 24759*
[40] NIST Special Publication (SP) 800-140E, *CMVP Approved Authentication Mechanisms: CMVP Validation Authority Requirements for ISO/IEC 19790 Annex E and ISO/IEC 24579 Section 6.17*
[41] NIST Special Publication (SP) 800-140F, *CMVP Approved Non-Invasive Attack Mitigation Test Metrics: CMVP Validation Authority Updates to ISO/IEC 24759*
[42] NIST Special Publication (SP) 800-190, *Application Container Security Guide*
[43] NIST Special Publication (SP) 800-208, *Recommendation for Stateful Hash-Based Signature Schemes*
[44] NIST Special Publication (SP) 800-209, *Security Guidelines for Storage Infrastructure*
[45] IEEE 1619-2018, *IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices*
[46] IEEE 1619.1-2018, *IEEE Standard for Authenticated Encryption with Length Expansion for Storage Devices*
[47] IEEE 1619.2-2021, *IEEE Standard for Wide-Block Encryption for Shared Storage Media*
[48] IEEE 2883-2022, IEEE *Standard for Sanitizing Storage*
[49] IETF RFC 1334, *PPP Challenge Handshake Authentication Protocol (CHAP)*

[50] IETF RFC 2246, *The TLS Protocol Version 1.0*
[51] IETF RFC 2404, *The Use of HMAC-SHA-1-96 within ESP and AH*
[52] IETF RFC 2406, *IP Encapsulating Security Payload (ESP)*
[53] IETF RFC 2451, *The ESP CBC-Mode Cipher Algorithms*
[54] IETF RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use with IPsec*
[55] IETF RFC 3686, *Using Advanced Encryption Standard (AES) Counter Mode*
[56] IETF RFC 3821, *Fibre Channel Over TCP/IP (FCIP)*
[57] IETF RFC 3723, *Securing Block Storage Protocols over IP*
[58] IETF RFC 4301, *Security Architecture for the Internet Protocol*
[59] IETF RFC 4302, *IP Authentication Header*
[60] IETF RFC 4303 *IP Encapsulating Security Payload (ESP)*
[61] IETF RFC 4306 *Internet Key Exchange (IKEv2) Protocol*
[62] IETF RFC 4346, *The Transport Layer Security (TLS) Protocol Version 1.1*
[63] IETF RFC 4949, *Internet Security Glossary, Version 2*
[64] IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*
[65] IETF RFC 6101, *The Secure Sockets Layer (SSL) Protocol Version 3.0*
[66] IETF RFC 7143, *Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)*
[67] IETF RFC 7144, *Internet Small Computer System Interface (iSCSI) SCSI Features Update*
[68] IETF RFC 7146, *Securing Block Storage Protocols over IP: RFC 3723 Requirements Update for IPsec v3*
[69] IETF RFC 8391, *XMSS: eXtended Merkle Signature Scheme*
[70] IETF RFC 8446, *The Transport Layer Security (TLS) Protocol Version 1.3*
[71] IETF RFC 8554, Leighton-Micali Hash-Based Signatures
[72] OASIS *Key Management Interoperability Protocol Specification Version 2.1*
[73] OASIS *Key Management Interoperability Protocol Profiles Version 2.1*
[74] SNIA, *Storage Security: Fibre Channel Security*
[75] National Security Agency (NSA) Commercial National Security Algorithm (CNSA), Suite 2.0, https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF
[76] National Security Agency (NSA) Cyber Security Directorate (CSD), Commercial Solutions for Classified (CSfC) Data-at-Rest (DAR) Capability Packages (CP), V5.0, https://www.nsa.gov/Resources/Commercial-Solutions-for-Classified-Program
[77] NVM Express Inc, NVM Express™ (NVMe™) Base Specification, Revision 2.0b, January 2022
[78] PCI-SIG, PCI Express™ (PCIe™) Base Specification Version 5.0
[79] PCI-SIG, PCI Express™ (PCIe™) Integrity and Data Encryption (IDE) ECN
[80] PCI-SIG, PCI Express™ (PCIe™) Data Object Exchange (DOE) ECN
[81] PCI-SIG, PCI Express™ (PCIe™) Component Measurement and Authentication (CMA) ECN
[82] PCI-SIG, PCI Express™ (PCIe™) TEE Device Interface Security Protocol (TDISP) ECN
[83] DMTF, DSP0274 Secure Protocol and Data Model (SPDM), Version 1.2
[84] DMTF, DSP0277 Secured Messages using SPDM Specification
[85] Compute Express Link Consortium, Inc., Compute Express Link™ (CXL™) Specification 3.0
[86] MIPI Alliance, Service Association Configuration Protocol (SACP)
[87] Trusted Computing Group (TCG), Storage Architecture Core Specification
[88] Trusted Computing Group (TCG), Storage Security Subsystem Class: Enterprise
[89] Trusted Computing Group (TCG), Storage Security Subsystem Class: Opal
[90] Trusted Computing Group (TCG), Storage Security Subsystem Class (SSC): Ruby
[91] Trusted Computing Group (TCG), Storage Security Subsystem Class (SSC): Opalite

[92] Trusted Computing Group (TCG), Storage Security Subsystem Class (SSC): Pyrite
[93] Trusted Computing Group (TCG), Storage Security Subsystem Class (SSC): Key Per I/O v1.0
[94] Trusted Computing Group (TCG), Storage Interface Interaction Specification (SIIS)

About SNIA

SNIA is a not–for–profit global organization, made up of member companies spanning the global storage market. SNIA's mission is to lead the storage industry worldwide in developing and promoting standards, technologies, and educational services to empower organizations in the management of information. To this end, the SNIA is uniquely committed to delivering standards, education, and services that will propel open storage networking solutions into the broader market. For more information, visit http://www.snia.org.

SNIA
5201 Great America Parkway, Suite 320, Santa Clara, CA, 95054
Phone:719-694-1380 • Fax: 719-694-1385 • www.snia.org