# Cryptography Review of W3C Verifiable Credentials Data Model (VCDM) and Decentralized Identifiers (DIDs) Standards and Cryptography Implementation Recommendations

October 15, 2021

Prepared for:
U.S. Department of Homeland Security
Science and Technology Directorate
Silicon Valley Innovation Program

Prepared by:
Nick Genise and David Balenson
SRI International


This work was sponsored by the U.S. Department of Homeland Security Science and Technology Directorate (DHS S&T) under Contract No. HSHQDC-16-C-00034. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Department of Homeland Security and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Department of Homeland Security.

# License Terms

# Contents

# Introduction

## Goal

The Department of Homeland Security (DHS) Science and Technology Directorate (S&T) Silicon Valley Innovation Program (SVIP) works with innovation communities across the nation and around the world to harness the commercial R&D ecosystem for technologies with government applications and to co-invest in and accelerate technology transition-to-market [11].

Within SVIP, the Blockchain and Distributed Ledger Technologies (DLT) portfolio is supporting technology developers whose innovative technologies have many uses and applications, including digitally issuing currently paper-based credentials, creating immutable records and audit logs of data, supply chain traceability, and privacy respecting essential work and task licenses for workers and individuals. The effort is, in some cases, utilizing blockchain and DLT solutions with emphasis on architecture, standards, and interoperability. In all cases, the intent is to overlay any solution, whether it is using blockchain, DLTs, or non-ledger-based technologies, with global, openly developed, standards-based data models and application programming interfaces (APIs). This will prevent the development of closed technology platforms and ensure the availability of a competitive marketplace of diverse, interoperable solutions for government and industry to draw upon to deliver cost effective and innovative solutions that are in the public interest [11]. To this end, the solutions being developed under SVIP are based, in part, on the World Wide Web Consortium (W3C)'s Verifiable Credentials Data Model (VCDM) [90] and Decentralized Identifiers (DIDs) standards [89].

To be used by U.S. government customers in their operational systems, the technologies must conform to relevant federal government standards and requirements, including the Federal Information Security Management Act (FISMA) and National Institute of Technology (NIST) standards for use of cryptography. As part of its support for the development and eventual deployment and use of these technologies within U.S. government systems, DHS S&T sponsored independent nonprofit research center SRI International to review the underlying W3C VCDM and W3C DIDs standards use of cryptographic operations (i.e., hash, digital signature, encryption) for conformance to the corresponding NIST cryptographic standards.

**SRI focused primarily on the cryptographic algorithms being used in the W3C standards and not on blockchain and DLT technologies or their use in operational systems.** An algorithmic review is an important starting point to a full, system-level review for compliance to the federal standards and other requirements. The intent of our review is to provide instructive feedback to the technology innovators and to the W3C standards developers to increase their level of compliance. This is crucial to deploying technologies based on advanced cryptography in the marketplace and hence the ability of U.S. government customers to use the technologies in operational systems.

## FISMA and Federal Information Processing Standards

FISMA requires each federal agency to develop, document, and implement an agency-wide program to provide information security for the information and systems that support the operations and assets of the agency, including those provided or managed by another agency, contractor, or other sources. Federal agencies need to provide information security protections commensurate with the risk and magnitude of the harm resulting from unauthorized access, use, disclosure, disruption, modification, or

destruction of: information collected/maintained by or on behalf of an agency, and information systems used or operated by an agency or by a contractor of an agency or other organization on behalf of an agency. Also, federal agencies need to "com[ply] with the information security standards" and guidelines, and mandatory required standards developed by NIST [71]. FISMA is part of the E-Government Act (Public Law 107-347) passed in December 2002 [84] and amended by the Federal Information Security Modernization Act of 2014 (Public Law 113-283) in December 2014 [85].

Federal Information Processing Standards Publications (FIPS PUBS) are the official series of publications relating to standards and guidelines adopted and promulgated under the provisions of FISMA. FIPS PUBS are issued by NIST after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) [83] and the Computer Security Act of 1987 (Public Law 100-235) [82]. FISMA does not allow for waivers to a FIPS PUB that is made mandatory by the Secretary of Commerce.

NIST cryptographic standards are applicable to all federal departments and agencies for the protection of sensitive unclassified information that is not subject to other laws governing national security and national security systems, in particular, not subject to Title 10 United States Code Section 2315 (10 USC 2315) [1] and that is not within a national security system as defined in Title 40 United States Code Section 11103(a)(1) (40 USC 11103(a)(1)) [2].

In addition, NIST cryptographic standards may be adopted and used by non-federal government organizations, such as private and commercial organizations. In fact, NIST encourages such use when it provides the desired security for commercial and private organizations.

Since cryptographic security depends on many factors besides the correct implementation of an encryption algorithm, NIST recommends that federal government employees, and others, should also refer to Guideline for Using Cryptographic Standards in the Federal Government: Directives, Mandates and Policies [57] and Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms [58], for additional information and guidance.

## Approach

SRI's review consisted of two primary phases. During the first phase, we reviewed relevant standards and publications and developed an initial set of recommendations to increase the level of compliance with the federal standards and requirements. We did this by exploring the W3C standards as well as the underlying NIST, Internet Engineering Task Force (IETF), and other cryptographic standards used by companies in the VCDM/DIDs ecosystem.

This online documentation included the W3C VCDM and DIDs standards, as well as all the underlying documents these standards pointed to. We also reviewed the relevant standards documents not referenced by the W3C standards. This phase was crucial to form an unbiased view of the needed criteria as well as to gauge how extractable the cryptographic methods are for potential users including government clients.

We also extensively reviewed the current NIST standards and guidelines for cryptography. The main FIPS PUBS related to cryptographic standards are:

- Secure Hash Standard (SHS), FIPS PUB 180-4 [72]

- SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (SHA-3), FIPS PUB 202 [76]
- The Keyed-Hash Message Authentication Code (HMAC), FIPS PUB 198-1 [79]
- Digital Signature Standard (DSS), FIPS PUB 186-4 [36] and FIPS PUB 186-5 [37]
- Advanced Encryption Standard (AES), FIPS PUB 197 [24]
- Security Requirements for Cryptographic Modules, FIPS PUB 140-2 [73] and FIPS PUB 140-3 [74]

In addition, we reviewed the relevant NIST special publications, IETF standards and informational documents, relevant research papers, and other documents.

SRI used the information gained from reviewing the above documents and standards to form an initial set of recommendations to the SVIP performers and to the W3C standards developers and implementers to help increase their level of compliance.

During the second phase, SRI presented the initial draft recommendations to the SVIP portfolio companies in March 2021 and elicited their feedback. We also interacted with selected SVIP portfolio companies, as well as NIST cryptography experts, to learn fine-grained details and future directions of the performer technologies and the NIST standards. These interactions solidified our understanding of the cryptographic algorithms used by the SVIP portfolio companies as well as relevant details of the NIST standards.

### Organization

Our recommendations are organized into three main sections: general cryptographic recommendations, algorithm and protocol recommendations, and other recommendations. General cryptographic recommendations include security strength and key material. Algorithms and protocols include hash, digital signature encryption, key agreement, and the Transport Layer Security protocol (TLS). Finally, we address other areas including use of cryptographic modules, cryptographic agility, improved documentation, key reuse, random number generation, and canonicalization. We also include an extensive set of references and some appendices.

## General Cryptographic Criteria

This section describes recommendations that apply generally across different cryptographic algorithms and protocols, including security strength and key material.

### Security Strength (or Bit Security)

The term *bit security strength* (also called *security strength* in NIST documents) in cryptography quantifies our knowledge on the resources an adversary needs to break some cryptographic scheme. Roughly, a scheme (such as a signature scheme) has bit security $\lambda$ if the best-known attack takes time $2^{\lambda}$. Bit security strength is a fundamental metric for cryptography and implementations.

> ***RECOMMENDATION***: Track the security level, as provided by NIST, of each individual component as well as the overall bit security of the scheme. These security levels should be tracked in W3C documentation as well as SVIP performer documentation. The latter should present security strengths to potential users, including government clients.

**Table 4: Security strength time frames**

| Security Strength | | Through 2030 | 2031 and Beyond |
|---|---|---|---|
| < 112 | Applying protection | Disallowed | |
| | Processing | Legacy use | |
| 112 | Applying protection | Acceptable | Disallowed |
| | Processing | | Legacy use |
| 128 | Applying protection and processing information that is already protected | Acceptable | Acceptable |
| 192 | | Acceptable | Acceptable |
| 256 | | Acceptable | Acceptable |

*Figure 1: Table 4 from NIST Recommendation for Key Management: Part 1 [57] describes time frames for acceptable use of different (bit) security strengths cryptography. The left half of the table shows security strength and type of use, where apply protection refers to encrypting and processing refers to decrypting and verifying signatures. See NIST Transitioning the Use of Cryptographic Algorithms and Key Lengths [80] for information about specific algorithms.*

There are multiple reasons to track bit security. One important reason is cryptographic agility, the ability for a system to operate with different possible cryptographic algorithms achieving the same capability. For example, TLS allows a variety of different cipher suites. This allows a server to still use TLS even if some algorithms are no longer deemed secure. If a system is designed to be cryptographically agile, then a user can swap out old cryptographic algorithms for new ones that achieve a desired security strength.

Security strengths are tracked by NIST. For example, Table 4 above is copied from the NIST Recommendations for Key Management [57]. This allows a developer to easily keep an understanding of the state of the art on security strengths. Further, NIST describes their short-term plan for algorithms and key lengths in Transitioning the Use of Cryptographic Algorithms and Key Lengths [80].

## Key Material

Here we describe our general recommendations for key generation, key storage, and key usage.

> ***RECOMMENDATION***: Use each cryptographic key, and the randomness used to generate the key, for a single cryptographic scheme. Further, all randomness is single use within the scheme.

This is required by all NIST standards. For example, the digital signature standard requires all keys used for signatures are single purpose [37]. The above recommendation ensures that the security of one scheme is not compromised by the reuse of a key in another scheme with a totally different security model. In most cases, cross-use of keys between cryptographic schemes is not guaranteed to be secure if both schemes are secure with independent keys.

Signatures are an area where the VCDM/DIDs technologies are reusing keys in a seemingly secure manner. This is being done in the X25519 Diffie-Hellman key agreement scheme and the Ed25519 digital signature scheme. The underlying secret scalar is being double used between both schemes, and Thormarker recently released an online paper [15] arguing security in the random oracle model. Nonetheless, such a reuse is not permitted by the DSS standard.

For randomness, an adversary can directly recover the secret key given the randomness used to create the key. Any leaked bits of this initial random string reduce the adversary's search space.

***RECOMMENDATION***: Delete all randomness used to encrypt (nonces) and used to generate keys as soon as the implementation can.

The rationale for deleting randomness used to generate keys was described above. Otherwise, reusing or leaking the randomness used to generate a signature or ciphertext can leak information about the secret key. This is most clearly seen in the case of AES encryption using the Output Feedback (OFB) mode of operation. OFB is a NIST-recommended block-cipher mode of operation.

## Cryptographic Algorithms and Protocols

This section gives recommendations for specific cryptographic algorithms and protocols, including hash functions, signature schemes, key agreement, encryption, and Transport Layer Security. For each we first give recommendations for a class of algorithms or protocols and then for specific instances of algorithms and protocols within each class. For example, we first give recommendations for all digital signature schemes, then we give specific recommendations for the Edwards-Curve Digital Signature Algorithm (EdDSA).

### Hash Functions

Hash functions are publicly known functions that take in messages of arbitrary length as input and produce a short digest as output. They are used throughout cryptography: in digital signatures, message authentication codes (MACs), deterministic random bit generators (DRBGs), and more.

Hash functions have important security properties:

- Preimage resistance – means that it is hard to invert the hash function given a specific output.
- Collision resistance – means that it is hard to find a pair of distinct inputs that hash to the same output.

Preimage resistance is important for MACs and collision resistance is important for digital signatures since we usually sign the hash of a message.

Either SHS [72] or SHA-3 [76] must be implemented wherever a secure hash algorithm is required for federal applications, including as a component within other cryptographic algorithms and protocols. Both may be adopted and used by non-federal government organizations.

SHA-3 specifies a family of functions on binary data, each based on an instance of the KECCAK algorithm that NIST selected as the winner of the SHA-3 Cryptographic Hash Algorithm Competition. The standard also specifies the KECCAK-p family of mathematical permutations, including the permutation that underlies KECCAK, which can serve as the main components of additional cryptographic functions that may be specified in the future [76]. The KECCAK-p permutations were designed to be suitable as the main components for a variety of cryptographic functions, including keyed functions for authentication and/or encryption. The six SHA-3 functions can be considered as modes of operation (modes) of the KECCAK-p[1600,24] permutation. In the future, additional modes of this permutation or other KECCAK-p permutations may be specified and approved in FIPS PUBS or in NIST Special Publications [76].

The SHA-3 family consists of four cryptographic hash functions, called SHA3-224, SHA3-256, SHA3-384, and SHA3-512, and two extendable-output functions (XOFs), called SHAKE128 and SHAKE256 [76].

The four hash functions specified in SHA-3 supplement the hash functions that are specified in SHS [72]. Together, both standards provide resilience against future advances in hash function analysis, because they rely on fundamentally different design principles. In addition to design diversity, the hash functions in SHA-3 provide some complementary implementation and performance characteristics to those in SHS.

For XOFs, the length of the output can be chosen to meet the requirements of individual applications. The XOFs can be specialized to hash functions, subject to additional security considerations, or used in a variety of other applications. The approved uses of XOFs will be specified in NIST Special Publications [76].

SHA-3 specifies that federal departments and agencies shall only use implementations of the KECCAK-p permutations within FIPS-approved or NIST-recommended modes of operation [76].

In addition to SHS and SHA-3, NIST describes functions built on top of SHA-3 in SHA-3 Derived Functions [75]. SHA-3 Derived Functions describes four specialized hashing algorithms built on top of SHA-3: cSHAKE, KMAC, TupleHash, and ParallelHash. cSHAKE is a customizable variant of hash function SHAKE, described in SHA-3. KMAC is a MAC that uses KECCAK, as described in SHA-3. KMAC can also be used as a pseudorandom function. Lastly, TupleHash and ParallelHash are hash functions which use cSHAKE as a key subroutine. TupleHash is a variable-length hash function designed to hash tuples of input strings without trivial collisions. ParallelHash is a variable-length hash function that can hash very long messages in parallel. Government use of these algorithms is allowed under FISMA. (See the “Authority” section in SHA-3 Derived Functions [75] for more details.)

Security guidelines for applications using approved hash functions are described in Recommendation for Applications Using Approved Hash Algorithms [48] and Transitioning the Use of Cryptographic Algorithms and Key Lengths [80].

> ***RECOMMENDATION***: Only use algorithms specified in SHS [72], SHA-3 [76], or SHA-3 derived functions [75].

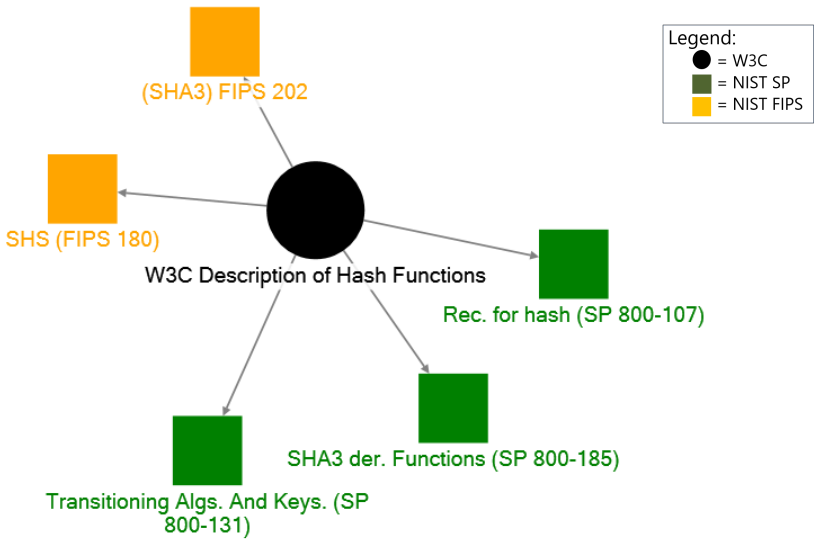


*Figure 2: A suggested documentation of graph for completely compliant hash function use.*

***RECOMMENDATION***: Implement all hash functions using approved cryptographic modules.

This criterion is required by both NIST hash standards, SHS [72] and SHA-3 [76]. Implementers may find this counter-intuitive since hash functions are the easiest to implement. On the other hand, hash functions are critical building blocks for other, more advanced cryptographic capabilities. Therefore, one needs an approved, secure hash function implementation.

***RECOMMENDATION***: Track the security strength of each hash function, as well as the type of security. Specify and describe the security strength and the type of security in the W3C standards and other specifications.

The estimated security strengths of approved hash functions are listed in Table 3 of NIST Recommendation for Key Management [57] and are copied above. Note the differing strengths of the same hash function for different applications. This illustrates the importance of tracking bit security under a specified threat model (e.g., digital signatures versus message authentication).

Also, note that each hash function has a list of approved uses. For example, SHA-1 is approved for message authentication in NIST Keyed-Hash Message Authentication Code (HMAC) [79], but not for use in digital signatures. This is because the former requires preimage resistance (inverting the hash function) versus collision resistance. For more details, see NIST Recommendation for Applications Using Approved Hash Algorithms [48].

## Block Ciphers

Block ciphers apply a deterministic encryption algorithm using a secret key to a block of text and are ubiquitous to modern cryptography. Block ciphers have many uses, including (authenticated) symmetric data encryption and decryption, random bit generation, key wrapping, key derivation, and message authentication codes (MACs). There are three block cipher families approved by NIST: Triple Data Encryption Standard (TDEA) [67], SKIPJACK [77], and AES [24]. The first two are either disallowed for encryption or being deprecated with both being disallowed by 2023 (see Figure 3 below). Overall, SKIPJACK is only allowed for legacy use (decryption) and TDEA will only be allowed for legacy use by 2023. See Transitioning the Use of Cryptographic Algorithms and Key Lengths for more details [80].

AES [24] specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. The AES algorithm can use cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

***RECOMMENDATION****:* Transition all block ciphers to AES.

We recommend that all uses of block ciphers transition to use of AES since it will be the only allowed block cipher, outside of legacy use, by 2023.

**Table 1: Approval Status of Symmetric Algorithms Used for Encryption and Decryption**

| Algorithm | Status |
|---|---|
| Two-key TDEA Encryption | Disallowed |
| Two-key TDEA Decryption | Legacy use |
| Three-key TDEA Encryption | Deprecated through 2023<br>Disallowed after 2023 |
| Three-key TDEA Decryption | Legacy use |
| SKIPJACK Encryption | Disallowed |
| SKIPJACK Decryption | Legacy use |
| AES-128 Encryption and Decryption | Acceptable |
| AES-192 Encryption and Decryption | Acceptable |
| AES-256 Encryption and Decryption | Acceptable |

*Figure 3: Table 1 from Transitioning the Use of Cryptographic Algorithms and Key Lengths [80].*

AES specifies that it may be used by federal departments and agencies when an agency determines that sensitive (unclassified) information (as defined in P. L. 100-235) requires cryptographic protection. Other FIPS-approved cryptographic algorithms may be used in addition to, or in lieu of, the standard. Federal agencies or departments that use cryptographic devices for protecting classified information can use those devices for protecting sensitive (unclassified) information in lieu of this standard [24].

AES [24] further specifies that the AES algorithm shall be used in conjunction with a FIPS-approved or NIST recommended mode of operation. There are seven accompanying guidelines on recommendations for different modes of operation:

- Recommendation for Block Cipher Modes of Operation Methods and Techniques [49],
- Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication [54],
- Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality [53],
- Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC [50],
- Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices [55],
- Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, SP 800-38F [51], and
- Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption [52].

Given a block cipher, its different modes of operations are straightforward to implement. One potential implementation pitfall is the random generation of nonces for modes of operations that require a truly random nonce or initialization vector. Such nonces must be generated carefully, using the relevant NIST

guidelines [64][65][66]. We give recommendations for random bit generation below and the same recommendations apply to nonces for block ciphers as well.

As with its other cryptographic algorithm standards, NIST will formally reevaluate the AES standard every five years. Both the standard and possible threats reducing the security provided through use of AES will undergo review by NIST as appropriate, taking into account newly available analysis and technology. In addition, the awareness of any breakthrough in technology or any mathematical weakness of the algorithm will cause NIST to reevaluate this standard and provide necessary revisions. In July 2021, NIST announced the release of NISTIR 8319, Review of the Advanced Encryption Standard [70]. The publication provides a technical and editorial review of AES and will be used in the review process for the standard. Information about the review process, including the initial public comments on the review of the standard, is available on the Cryptographic Publication Review Project page [32].

## Message Authentication Codes

Message authentication is achieved via the construction of a message authentication code (MAC). MACs based on cryptographic hash functions are known as HMACs. The purpose of a MAC is to authenticate both the source of a message and its integrity without the use of any additional mechanisms. MACs are symmetric cryptographic schemes since both user endpoints use the same secret key. Approved MAC constructions are given in The Keyed-Hash Message Authentication Code (HMAC) [79], Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication [54], Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC [50], and SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash [75] (for KMAC). Note that HMAC advises that keys used for HMAC applications should not be used for other purposes [79].

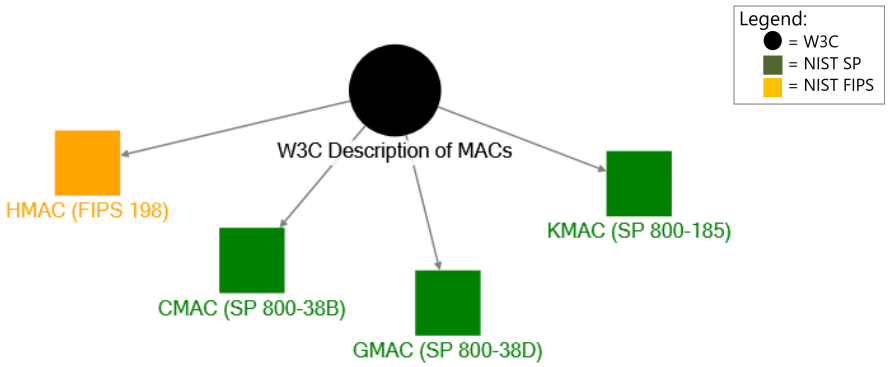


*Figure 4: A suggested documentation of graph for completely compliant MAC algorithms.*

***RECOMMENDATION:*** **Avoid using MACs built on TDEA and use at least 112-bit key lengths.**

The rationale for the above recommendation is that all other MAC algorithms will shortly be authorized only for legacy use, or already are. See Figure 5 for more details.

**Table 9: Approval Status of MAC Algorithms**

| MAC Algorithm | Implementation Details | Status |
|---|---|---|
| HMAC Generation | Key lengths < 112 bits | Disallowed |
| | Key lengths ≥ 112 bits | Acceptable |
| HMAC Verification | Key lengths < 112 bits | Legacy use |
| | Key lengths ≥ 112 bits | Acceptable |
| CMAC Generation | Two-key TDEA | Disallowed |
| | Three-key TDEA | Deprecated through 2023<br>Disallowed after 2023 |
| | AES | Acceptable |
| CMAC Verification | Two-key TDEA | Legacy use |
| | Three-key TDEA | Legacy use |
| | AES | Acceptable |
| GMAC Generation and Verification | AES | Acceptable |
| KMAC Generation and Verification | Key lengths < 112 bits | Disallowed |
| | Key lengths ≥ 112 bits | Acceptable |

*Figure 5: Table 9 from Transitioning the Use of Cryptographic Algorithms and Key Lengths [80] describing the short-term plan for MAC algorithm transition.*

## Signature Schemes

DSS has two relevant forms: the current standard [36] and a stable draft [37]. We refer to the latter as DSS since it is stable, and the updates will be adopted soon.

All approved signatures follow the hash and sign paradigm, also known as the full domain hash paradigm. Hence, a hash function is used in the signature generation process to obtain a condensed version of the data to be signed; the condensed version of the data is often called a message digest. The message digest is input to the digital signature algorithm to generate the digital signature. The hash functions to be used are specified in SHS [72]. FIPS-approved digital signature algorithms shall be used with an appropriate hash function that is specified in the SHS. The digital signature is provided to the intended verifier along with the signed data (i.e., the original message not just the digest). The verifying entity verifies the signature by using the claimed signatory's public key and the same hash function that was used to generate the signature. Similar procedures may be used to generate and verify signatures for both stored and transmitted data.

A canonicalization algorithm is used to canonicalize data which is then inputted to the hash function. To avoid easy forgeries, the canonicalization algorithm should be treated as a cryptographic function. We discuss this in more detail in the canonicalization section below.

Here we present suggestions regarding digital signature schemes. We first list the general recommendations which apply to all digital signatures, then the recommendations for distinct digital signature algorithms. The general hash and sign paradigm used in all approved signature algorithms is illustrated in Figure 7. Also, an illustration of potential references for approve algorithms is given in

Figure 6. Providing clear and encompassing documentation on digital signatures is more challenging than the previous algorithms: hashes, block ciphers, and MACs. This is due to the wider variety of digital signature algorithms. Note, these algorithms will become even more diverse once NIST transitions to post-quantum signatures [69][80]. One solution is to prepare documentation with cryptographic agility in mind (see below). This means having a general description of digital signatures, then having separate documentation for each specific signature algorithm.

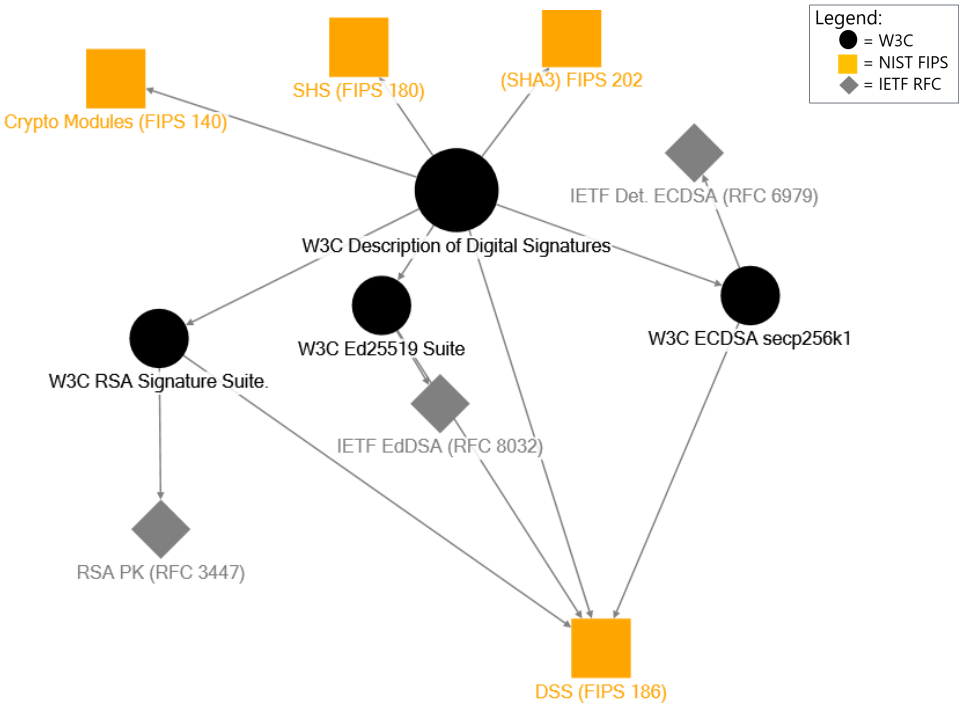


*Figure 6: A suggested documentation of graph for completely compliant digital signature schemes.*

**General**

All NIST-approved digital signature algorithms follow the framework illustrated by Figure 7 – a signer hashes the message to a digest, then signs the digest. On the receiving end, the verifier hashes the message and verifies the signature on the hash.

In general, the most challenging aspect of implementing a digital signature scheme is random number generation. Incorrect random number generation leads to direct private key recovery attacks. For example, Breitner and Heninger's attack on Elliptic Curve Digital Signature Algorithm (ECDSA) is an explicit example [18] of poor random number generation causing a security fault. We list some general recommendations regarding randomness below.

> ***RECOMMENDATION****:* Use approved random number generators to generate random keys [64][65][66] and treat the randomness, or seed, as key material, i.e., protected as in NIST Recommendations for Key Management [57].

***RECOMMENDATION**:* Use approved random number generators to generate randomness for signatures and delete the randomness as soon as possible.

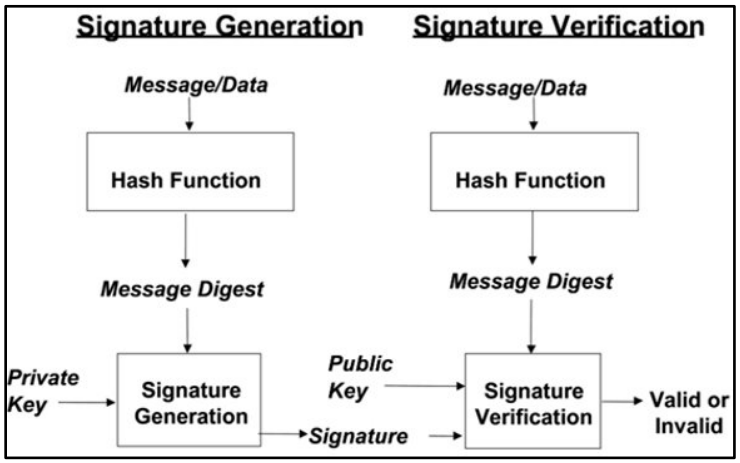


*Figure 7: A block diagram (Figure 1 in DSS [37]) illustrating digital signature algorithms.*

DSS [37] specifies the following signature schemes: RSA, ECDSA, and EdDSA. These are the only approved digital signature algorithms. Previous versions of DSS specified the DSA algorithm but it is now obsolete and can only be used to verify existing signatures. In other words, the DSA algorithm is only approved for legacy use.

Note that DSS advises that digital signature key pairs shall not be used for other purposes. This is discussed in more detail in the "Key Reuse between Ed25519 and X25519" section below under Other Recommendations. Note that digital signature algorithms are complicated and delicate. These algorithms either use randomness or are deterministic. The correct implementation of randomness is vital for security. Conversely, deterministic algorithms potentially allow timing attacks and must be constant time.

DSS specifies that digital signature implementations that comply with the standard shall employ cryptographic algorithms, cryptographic key generation algorithms, and key establishment techniques that have been approved for protecting federal government sensitive information. Approved cryptographic algorithms and techniques include those that are either: specified in a FIPS PUB, adopted in a FIPS PUB or a NIST Recommendation, or specified in the list of approved security functions for FIPS PUB 140.

Note that NIST is proposing updates to its digital signature and elliptic curve cryptographic standards to align with existing and emerging industry standards [37]. As part of these updates, NIST is proposing to adopt two new elliptic curves, Ed25519 and Ed448, for use with EdDSA. EdDSA is a deterministic elliptic curve signature scheme currently specified in the RFC 8032, Edwards-Curve Digital Signature Algorithm [86]. NIST further proposes adopting a deterministic variant of ECDSA, which is currently specified in RFC 6979, Deterministic Usage of the Digital Signature Algorithm and Elliptic Curve Digital Signature Algorithm [87].

**DSA**

Given the increased industry adoption of ECDSA within security products, the draft update to DSS also proposes the removal of DSA, which would prohibit use of DSA for generating digital signatures but allow legacy use to verify existing signatures [37]. The current draft of DSS states that the digital signature algorithm, DSA, will no longer be approved for signature generation, only verification. Therefore, we list the following criterion.

> ***RECOMMENDATION***: Use the DSA algorithm, as listed in DSS version 4, only to verify existing signatures and not to generate new signatures.

**RSA**

The RSA digital signature algorithm is specified in Section 5 of DSS and is based on PKCS#1 [19]. We highlight a few important RSA recommendations in Section 5 of DSS. Note that any implementation or cryptographic module must follow all criteria in Section 5 of DSS to be compliant with the standard.

> ***RECOMMENDATION****:* Generate the RSA modulus with approved random bit generators.

The RSA modulus' trapdoor is its prime factorization, $n = pq$. Therefore, implementations must be careful to not leak any extra information about these factors.

> ***RECOMMENDATION****:* Generate an RSA modulus $n = pq$ with two primes of the same bit-length. Keeping the bit-length of the two factors the same is needed to reach the estimated security strengths for RSA.

> ***RECOMMENDATION****:* Use an RSA modulus of at least 2048 bits. This ensures the security strength is at least 112 bits.

**Table 4: Security strength time frames**

| Security Strength | | Through 2030 | 2031 and Beyond |
|---|---|---|---|
| < 112 | Applying protection | Disallowed | |
| | Processing | Legacy use | |
| 112 | Applying protection | Acceptable | Disallowed |
| | Processing | | Legacy use |
| 128 | Applying protection and processing information that is already protected | Acceptable | Acceptable |
| 192 | | Acceptable | Acceptable |
| 256 | | Acceptable | Acceptable |

*Figure 8: Table 4 from Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography [63] lists the security strengths of RSA moduli.*

> ***RECOMMENDATION****:* Use a hash function with a security strength at least the strength of the RSA modulus bitlength.

> ***RECOMMENDATION****:* Protect the RSA secret key as described in Recommendations for Key Management [57].

Another common technique used in implementations is storing secret RSA exponents in the Chinese Remainder Theorem (CRT) form. This is detailed in Sections 6.2 and 6.3 of Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography [63]. We note that storing the secret exponent in CRT form and performing operations on it in CRT likely make side-channel attacks easier since the CRT is done over $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$ where $p, q$ are the secret RSA factors and $\varphi(\cdot)$ is Euler's totient function. Therefore, CRT operations must resist side-channels.

**Update on Elliptic Curves for Government Use**

NIST recently updated their recommendations for elliptic curves [68]. Though this recommendation is technically a draft, implementers should expect it to be stable. This NIST recommendation includes updates on curves originally specified in the previous version of DSS [79], specifies new Montgomery and Edwards curves (specified in IETF's Elliptic Curves for Security [3]), serves as a reference for the Brainpool curves (specified in IETF's Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation [20]), and specifies which curves are for legacy use and which are being deprecated. The main takeaways for implementers are the newly allowed curves and the curves being depreciated or changed to legacy-use only. According to the draft, all curves over binary fields are being deprecated and implementations should use curves over prime fields. The following curves will soon be downgraded to legacy use: B-163, K-163, and P-192.

> ***RECOMMENDATION***: Use curves over prime fields with a subgroup at least 224 bits (preferably 256 bits or more, see Figure 9 below for more details).

**ECDSA**

The main recommendation is to follow the ECDSA criteria listed in Section 6 of DSS [37]. Here we list some important details from DSS Section 6.

> ***RECOMMENDATION****:* Use a hash function with security strength at least that of the underlying elliptic curve.
>
> ***RECOMMENDATION****:* Protect the private key as described in Recommendations for Key Management [57]**.**
>
> ***RECOMMENDATION****:* If using deterministic ECDSA, keep all operations involving the private (signing) key constant time. That is, the time to compute a function which uses the private key must not depend on the key's bits. See Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) [87] and DSS [37].

**Table 1: ECDSA Security Parameters**

| Bit length of $n$ | Maximum Cofactor ($h$) | Comparable Security Strength |
|---|---|---|
| 224 - 255 | $2^{14}$ | approximately $n/2$; at least 112 bits |
| 256 - 383 | $2^{16}$ | approximately $n/2$; at least 128 bits |
| 384 - 511 | $2^{24}$ | approximately $n/2$; at least 192 bits |
| $\geq 512$ | $2^{32}$ | approximately $n/2$; at least 256 bits |

*Figure 9: Table 1 from DSS describes the security strength of approved elliptic curves. The parameter n is the number of bits to describe the discrete log group on the curve.*

**EdDSA**

The main recommendation is to follow the EdDSA criteria listed in Section 7 of DSS. Here we list some specific recommendations and their rationales.

> ***RECOMMENDATION****:* Use SHA-512 for Ed25519 as specified in the SHS (and track the security strength).
>
> ***RECOMMENDATION****:* Use SHAKE-256 for Ed448 as specified in the SHA-3 standard [76].
>
> ***RECOMMENDATION****:* Protect the private key as described in Recommendations for Key Management [57]**.**
>
> ***RECOMMENDATION****:* Keep all operations involving the secret (signing) key constant time. That is, the time to compute a function which uses the secret key must not depend on the key's bits. See Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) [87] and DSS. This includes the special case where the secret key is stored as a seed to a derivation function. All operations on the seed, as well as the secret key, must be constant time.

The above recommendations are needed because Edwards signatures are deterministic. The nonce used in the signature generation is deterministically generated using a hash function on the message and the key. Therefore, the use of an approved cryptographic hash function module is crucial here since a non-constant time implementation could leak information about the secret key.

**BBS+**

The BBS+ signature scheme, described by Ho Au et al. [21] as an extension of the BBS group signature scheme by Boneh et al. [10], is a signature scheme often adapted to support a feature called *selective disclosure*. Selective disclosure is used when a user holds a signature for multiple messages, but the user

can prove the veracity of the signature for a selected subset of the message. In more detail, a user holds a digital signature, signed by an authority, and reveals only some of the signed messages while proving in zero-knowledge that the signature holds for the unrevealed messages. BBS+ signatures are adopted and in the process of being standardized by W3C. BBS+ signatures are not standardized by NIST and are unlikely to be standardized by NIST since they rely on elliptic curves with bilinear pairings. The latter is because curves with bilinear pairings are not secure against quantum algorithms and NIST is expected to only standardize post-quantum schemes from here on out. However, we make some recommendation that move a BBS+ implementation closer to compliance.

***RECOMMENDATION**:* Keep an up-to-date note in the W3C specification of the BBS+ algorithm on the estimated security strength of the elliptic curve and its bilinear pairing. The specification of the BBS+ algorithm [88] uses the BLS12-381 curve. This curve has roughly 117 bits of security according to the recent NCC Zcash review [23]. Security strength for bilinear pairings is less straightforward than other discrete log groups since there are three associated groups for a pairing, the two elliptic curve groups as well as the target group, which is a discrete log group in a finite field. The latter has much larger bandwidths than points on the curves.

> ***RECOMMENDATION**:* Use the SHAKE256 hash function from SHA-3 [76] with at least 508-bit output, instead of the BLAKE2b hash function. This recommendation is to preserve security strength while using an approved hash function.

***RECOMMENDATION**:* Use an approved random number generator in the BBS+ signature implementation.

Note, these recommendations will not affect if a government client uses BBS+ signatures. They cannot unless NIST standardizes BBS+.

### Key Agreement

A (pairwise) key-establishment procedure in which the resultant secret keying material is a function of information contributed by both participants so that neither party can predetermine the value of the secret keying material independently from the contributions of the other party.

NIST has multiple approved key agreement protocols based on elliptic curves and RSA-based algorithms [62][63]. These number-theoretic algorithms generate a seed. This seed is then fed to a key derivation algorithm. See the Recommendation for Key-Derivation Methods in Key-Establishment Schemes [60].

Key exchange protocols over elliptic curves are only approved for the curves in Table 24 from NIST Recommendation for Pair-Wise Key Establishment Using Discrete Logarithm Cryptography [62].

**Table 24: Appoved elliptic curves for ECC key-agreement.**

| Referenced in: | FIPS 186-4 SP 800-56A | TLS (RFC 4492) (SP 800-52) | IPsec w/ IKE v2 (RFC 5903) | Targeted Security Strengths that can be Supported |
|---|---|---|---|---|
| **Specified in:** | SP 800-186[29] | SEC 2 | RFC 5903 | |
| | P-224 | secp224r1 | - | $s = 112$ |
| | P-256 | secp256r1 | secp256r1 | $112 \le s \le 128$ |
| | P-384 | secp384r1 | secp384r1 | $112 \le s \le 192$ |
| | P-521 | secp521r1 | secp521r1 | $112 \le s \le 256$ |
| | K-233 | sect233k1 | - | $112 \le s \le 128$ |
| | K-283 | sect283k1 | - | $112 \le s \le 128$ |
| | K-409 | sect409k1 | - | $112 \le s \le 192$ |
| | K-571 | sect571k1 | - | $112 \le s \le 256$ |
| | B-233 | sect233r1 | - | $112 \le s \le 128$ |
| | B-283 | sect283r1 | - | $112 \le s \le 128$ |
| | B-409 | sect409r1 | - | $112 \le s \le 192$ |
| | B-571 | sect571r1 | - | $112 \le s \le 256$ |

*Figure 10: Table 24 of NIST Recommendation for Pair-Wise Key Establishment for Pair-Wise Key Establishment using Discrete Logarithm Cryptography [62] lists approved elliptic curves for ECC key-agreement.*

There are some other remaining challenges: instantiating the correct message authentication code (MAC) and correctly using an approved random bit generator.

> ***RECOMMENDATION**:* Follow the specifications for key agreement in Recommendation for Key-Derivation Methods in Key-Establishment Schemes [60], Recommendation for Pair-Wise Key Establishment Using Discrete Logarithm Cryptography [62], and (or) Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography [63].

The above recommendation includes using an approved elliptic curve or finite field discrete log group or using an RSA modulus of length at least 2048. In addition, one must use an approved method of key derivation and hash algorithm, or MAC algorithm as specified in NIST Recommendation for Key Derivation Methods in Key-Establishment Schemes [60].

Random bit generation is a crucially important step for key exchange protocols. Here we list some general criteria.

> ***RECOMMENDATION**:* Treat all randomness and intermediate values during the key exchange as key material. Further, safely remove these values from memory as soon as the application or algorithm allows.

> ***RECOMMENDATION**:* Generate all random values with approved random number generators with security strength at least as high as the key agreement scheme. Approved random number generators are given in the Recommendation for Random Number Generation Using Deterministic Random Bit Generators [65], the Recommendation for the Entropy Sources Used for Random Bit Generation [66], and the Recommendation for Random Bit Generator (RBG) Constructions [64].

**Table 25: Approved IKE groups for FFC key agreement.**

| IKE v2 (RFC 3526) | Targeted Security Strengths that can be Supported |
|---|---|
| MODP-2048 (ID=14) | $s = 112$ |
| MODP-3072 (ID=15) | $112 \leq s \leq 128$ |
| MODP-4096 (ID=16) | $112 \leq s \leq 152$* |
| MODP-6144 (ID=17) | $112 \leq s \leq 176$* |
| MODP-8192 (ID=18) | $112 \leq s \leq 200$* |

*Figure 11: Table 25 from Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography [62] lists the approved IKE groups for FFC key agreement.*

***RECOMMENDATION**:* When feasible, a nonce's random string should be twice the length of the targeted security strength for the key agreement scheme.

Key agreement schemes are complicated cryptographic operations since they involve multiple cryptographic building blocks and involve exchanges between semi-trusting parties. For example, identity assurances during key agreement are needed for trustworthy key generation. Some methods for assurance are in the Recommendation for Obtaining Assurances for Digital Signature Applications [61]. Further, key agreement schemes use a mix of ephemeral and static keys and message authentication codes (MACs).

Rationale for which key agreement scheme to use for certain applications is given in Section 7 of the Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography [62]. The main differences are how many ephemeral and static keys each party generates during the key-establishment protocol. For example, a key-establishment scheme with one ephemeral key is appropriate in settings where one party has limited access to truly random bits. We also note that the Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography [62] and the Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography [63] both stress that all keys and intermediate values used to derive keys should be single purpose. That is, a key and all the values used to determine that key are used either for a symmetric encryption scheme and any other cryptographic algorithm uses fresh inputs (random bits).

**Table 26: Approved TLS groups for FFC key agreement.**

| TLS (RFC 7919) | Targeted Security Strengths that can be Supported |
|---|---|
| ffdhe2048 (ID = 256) | $s = 112$ |
| ffdhe3072 (ID = 257) | $112 \leq s \leq 128$ |
| ffdhe4096 (ID = 258) | $112 \leq s \leq 152$* |
| ffdhe6144 (ID = 259) | $112 \leq s \leq 176$* |
| ffdhe8192 (ID = 260) | $112 \leq s \leq 200$* |

*Figure 12: Table 26 from Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography [62] lists the approved TLS groups for FFC key agreement.*

### Transport Layer Security

Transport Layer Security [14] (TLS) is the standard protocol for secure client-server communications. TLS uses multiple cryptographic building blocks (digital signatures, MACs, etc.) and some instantiations of these blocks are not approved by NIST for government use. The Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations [44] contains details on compliant TLS protocols, including approved cipher suites. We illustrate some important facts below.

> ***RECOMMENDATION**:* All TLS protocol implementations shall be versions 1.2 and 1.3. Version 1.3 by 2024.

> ***RECOMMENDATION**:* Use the NIST-approved cipher suites given in Sections 3 and 4 of Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations [44].

> ***RECOMMENDATION**:* Servers (and clients) in TLS must use NIST-approved cryptographic modules.

## Other Recommendations

This section provides recommendations in other areas beyond the general cryptographic criteria and algorithms, and protocols, including use of validation programs, cryptographic agility, improved documentation, key reuse, random number generation, and canonicalization.

### Cryptographic Validation Programs

The NIST cryptographic standards specify that they may be implemented as cryptographic modules comprised of software, firmware, hardware, or any combination thereof. The specific implementation may depend on several factors such as the application, the environment, the technology used, etc.

NIST established the Cryptographic Algorithm Validation Program (CAVP) [33] and the Cryptographic Module Validation Program (CMVP) [35] to validate cryptographic modules conforming to the NIST Security Requirements for Cryptographic Modules (FIPS PUB 140 series) and other FIPS cryptographic standards.

> ***RECOMMENDATION***: Implementations should use validated cryptographic algorithms and modules whenever possible.

CAVP provides validation testing of approved (i.e., FIPS-approved and NIST-recommended) cryptographic algorithms and their individual components [33]. Cryptographic algorithm validation is a prerequisite of cryptographic module validation.

CMVP provides validation testing of cryptographic modules [35]. The overarching goal of CMVP is to promote the use of validated cryptographic modules and provide federal agencies with a security metric to use in procuring equipment containing validated cryptographic modules.

CAVP and CMVP leverage independent Cryptographic and Security Testing (CST) laboratories that have been accredited under the NIST National Voluntary Laboratory Accreditation Program (NVLAP) [47]. The labs test cryptographic algorithms and modules against the applicable test requirements, implementation guidance, and other programmatic guidance. Based on the results of the laboratory testing, NIST then officially validates the algorithms and modules for conformance to the applicable standards.

NIST maintains a list of algorithm implementations successfully tested by a lab and validated by NIST [34]. The list identifies the vendor, implementation, operational environment, validation date and algorithm details.

NIST also maintains a list of current Validated Modules [81] with official information of all cryptographic modules that have been tested and validated under CMVP. NIST also provides a Modules in Process List [46], which lists cryptographic modules on which the CMVP is actively working, as well as an Implementation Under Test List [45], which lists modules from vendors who have a viable contract with an accredited laboratory for the testing of a cryptographic module.

Note that NIST is in the process of developing Automated Cryptographic Validation Testing (ACVT) to improve the efficiency and effectiveness of cryptographic module testing in order to reduce the time and cost required for testing while providing a high level of assurance for federal government consumers. For more information about the overall CAVP and CMVP conformance testing process and ACVT see [25].

The CMVP continues to validate cryptographic modules to FIPS PUB 140-2, Security Requirements for Cryptographic Modules, until September 21, 2021, without an extension request. On April 1, 2022, CMVP will no longer accept FIPS PUB 140-2 submissions new validations. As of September 22, 2020, CMVP additionally began validating cryptographic modules to FIPS PUB 140-3, Security Requirements for Cryptographic Modules [74]. Note that FIPS PUB 140-3 is a modification of ISO/IEC Information Technology — Security Techniques — Security Requirements for Cryptographic Modules standard [17] and is accompanied by a series of guidelines [39][26][27][28][29][30][31].

## Cryptographic Agility

Cryptographic agility is a system's ability to easily swap out algorithms achieving the same cryptographic service, under the same or a stronger security model. For example, compliant digital signatures can be implemented with either RSA algorithms or elliptic curve algorithms. Both implementations achieve the same cryptographic service under the same security model. However, they have drastically different underlying algorithms and features. A system has cryptographic agility if it can easily swap one signature algorithm for another signature algorithm with a similar or increased security level.

Cryptographic agility is important for several reasons. First, new attacks often lower the bit-security of deployed systems and implementers need to change algorithms quickly. For example, a timing attack on a deployed digital signature scheme would require all signing operations to be changed to constant time. As another example, a deployed algorithm, e.g., MD5, can be deemed insecure completely, or cryptographically broken. Second, the deployed systems requirements might change over time. For example, an application might initially require fast signing algorithms in digital signatures whereas later it requires signatures with small bandwidth without strict signing time requirements.

Further, cryptographic agility is now more important than ever with the threat from quantum computers. It is well-known that a large quantum computer would deem cryptography based on factoring and the discrete log problem completely broken. For example, an adversary can store digital signatures today to forge a signature in the past once they have access to a large quantum computer. Some applications, like digitally signed birth certificates, have usage well into the timeframe where this a plausible scenario. Cryptographic algorithms that are secure against adversaries with quantum computers fall under *post-quantum cryptography*. Note that these cryptographic algorithms are entirely

classical, they can be run on today's non-quantum systems. See the NIST Report on Post-Quantum Cryptography for more details [69].

**Table 1 - Impact of Quantum Computing on Common Cryptographic Algorithms**

| Cryptographic Algorithm | Type | Purpose | Impact from large-scale quantum computer |
|---|---|---|---|
| AES | Symmetric key | Encryption | Larger key sizes needed |
| SHA-2, SHA-3 | ---------- | Hash functions | Larger output needed |
| RSA | Public key | Signatures, key establishment | No longer secure |
| ECDSA, ECDH (Elliptic Curve Cryptography) | Public key | Signatures, key exchange | No longer secure |
| DSA (Finite Field Cryptography) | Public key | Signatures, key exchange | No longer secure |

*Figure 13: Table 1 from Report on Post-Quantum Cryptography [69] summarizing how a full-scale quantum computer, capable of running Grover's and Shor's algorithm, would impact cryptography in-use today.*

NIST is close to standardizing post-quantum algorithms for cryptography [78]. Once the post-quantum schemes are standardized, the previous "pre-quantum" asymmetric (public-key) schemes (RSA, ECDSA, EdDSA, Diffie-Hellman) will be phased out as soon as possible. Therefore, it is crucial that implementers prepare for this by designing cryptographic agility into all systems that require cryptography.

The impact of quantum computers on symmetric cryptography will be much less than for asymmetric cryptography. In general, all key-lengths in symmetric cryptographic services will need to be doubled. This has some subtle side effects. For example, implementers need to be ready to double hash function output as well as doubling AES key lengths. This is because quantum computers have a generic halving effect on symmetric security (bit security goes from $\lambda$ to $\lambda/2$ by Grover's algorithm [16]).

> ***RECOMMENDATION**:* Develop all systems with cryptographic agility to allow swapping out different algorithms achieving the same cryptographic goals, especially considering future use of post-quantum cryptography.

Finally, note that many of the schemes implemented in the VCDM/DIDs ecosystem, from our understanding, have a somewhat low security strength (112-128 bits). These schemes include X25519, Ed25519 signatures, and BBS+ signatures. We believe implementations within this security strength are more likely than others to be deemed as not secure at some point in the future. Therefore, it is important to design a system with cryptographic agility to mitigate the costs of switching algorithms once deployed algorithms are deemed not secure.

## Documentation

Here we list some recommendations for the W3C standards, specifications, and supporting documents. Accessible documentation is crucial for government use since many government clients will not be experts in cryptography yet will need to understand a technology's underlying mechanisms for appropriate deployment.

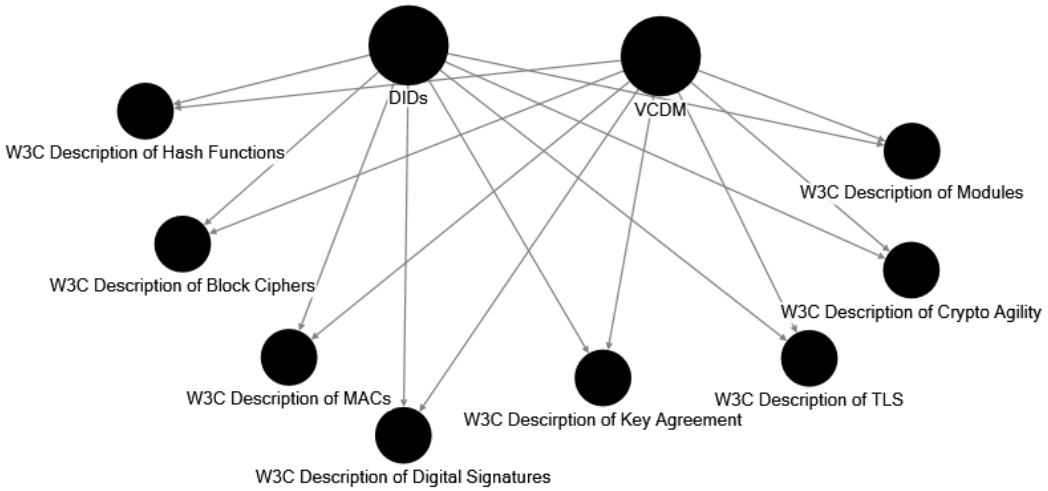


*Figure 14: A sample for potential W3C documentation. Each description of a cryptographic algorithm would have its own important references and links (Figure 2, for example).*

***RECOMMENDATION**:* Write standard documents with cryptographic agility in mind.

This recommendation can be achieved by describing each cryptographic service as an abstract, higher-level operation independent of the underlying algorithm(s).

For example, for digital signatures, specify the signature security model, and for each higher-level operation that uses a digital signature, specify the needed properties, like selective disclosure. Different digital signature algorithms can be used to meet the security model and desired properties. More specifically, a digital signature could be described simply by Figure 7 above together with the existential or strong existential unforgeability security game [22].

***RECOMMENDATION**:* Specify a government-compliant list of algorithms for each cryptographic building block. Also, include a government-compliance section for each crypto suite.

The latter part of the above recommendation means that W3C standards and other documents will have explicit directions for government compliant modes of operations. This is important since approved cryptographic modules can have unapproved modes of operation. Further, a good starting point for an overview for compliant algorithms, besides this document, are NIST's high-level guidelines for cryptographic use and implementation [42][43].

***RECOMMENDATION***: Include block diagrams describing algorithms and systems' input-output behavior (especially for generic cryptographic building blocks, e.g., hash functions). There should also be text describing input-output behavior for specific algorithms as well.

## Key Reuse between Ed25519 and X25519

The Ed25519 elliptic curve and Curve25519 (a Montgomery curve) are two commonly used curves for different cryptographic algorithms: Ed25519 for digital signatures and Curve25519 for Diffie-Hellman key exchange [7][6]. These curves are isomorphic with a simple, birational, map between them. Therefore, it is common in for implementers to double use a secret scalar (private key) between both

schemes. The private key holder then uses this map to move their secret point from one curve to the other.

Though there is no obvious attack from this double-use, and Thormarker's recent paper [15] claims to prove it is secure in the random oracle model, double using key material is not allowed in NIST-compliant implementations.

## Random Number Generation

Here we discuss recommendations for generating cryptographically secure random bits. We emphasize that random bit generation is a crucial building block for secure cryptographic implementations. Furthermore, weak random bit generation is often the first target in attacks. The full details are in the Recommendation for Random Number Generation Using Deterministic Random Bit Generators [65], the Recommendation for Entropy Sources Used for Random Bit Generation [66], and the Recommendation for Random Bit Generator (RBG) Constructions [64].

There are two main notions of random bit generation: deterministic random bit generators (DRBGs) and nondeterministic random bit generation (NRBGs). DRBGs are deterministic functions which take as input a sufficiently random string and expand it to a pseudorandom bitstring. The entropy of this random bitstring is the most important security measure of DRBGs. For example, the entropy in DRBGs must be at least the security strength in hashed-based DRBGs (see Figure 15). We note that there are multiple inputs to each DRBG. On the other hand, NRBGs produce an output with full entropy (truly random output). We note that there are only two main components in random bit generation: the entropy source and a method to produce (pseudo)random bits for a cryptographic application. DBRGs and NBRGs are the latter component.

Earlier in this document we repeatedly said that the random bits in a cryptographic system, e.g., the random bits used to sign a message, are single use. That is, they are to be used once then deleted. The same holds for the output of the entropy source, as expected.

***RECOMMENDATION**:* All entropy source outputs are single use.

DRBGs have multiple inputs (entropy source, nonce, personalization string, and the additional input) and use cryptographic building blocks like hashes and block ciphers. However, there are only a few inputs which are treated as secrets. As a rule of thumb, we suggest the following.

***RECOMMENDATION**:* Treat the entropy source as key material.

Another potential layer of security is a derivation function for DRBGs. Derivation functions use hash functions and block ciphers.

**Table 2: Definitions for Hash-Based DRBG Mechanisms**

| | SHA-1 | SHA-224 and SHA-512/224 | SHA-256 and SHA-512/256 | SHA-384 | SHA-512 |
|---|---|---|---|---|---|
| **Supported security strengths** | See [SP 800-57] | | | | |
| ***highest_supported_security_strength*** | See [SP 800-57] | | | | |
| **Output Block Length (*outlen*)** | 160 | 224 | 256 | 384 | 512 |
| **Required minimum entropy for instantiate and reseed** | *security_strength* | | | | |
| **Minimum entropy input length (*min_length*)** | *security_strength* | | | | |
| **Maximum entropy input length (*max_ length*)** | $2^{35}$ bits | | | | |
| **Seed length (*seedlen*) for Hash_DRBG** | 440 | 440 | 440 | 888 | 888 |
| **Maximum personalization string length (*max_personalization_string_length*)** | $2^{35}$ bits | | | | |
| **Maximum additional_input length (*max_additional_input_length*)** | $2^{35}$ bits | | | | |
| ***max_number_of_bits_per_request*** | $2^{19}$ bits | | | | |
| **Maximum number of requests between reseeds (*reseed_interval*)** | $2^{48}$ | | | | |

*Figure 15: Table 2 from Recommendation for Random Number Generation Using Deterministic Random Bit Generators [65].*

> ***RECOMMENDATION**:* If a DBRG uses a hash or block cipher, then the security strength of the block cipher or the hash should be larger than the desired security strength of the whole system.

For example, say a DBRG is used in a digital signature scheme. Then the DBRG must have security strength at least that of the scheme, and every component within the DBRG must have a security strength at least that of the digital signature. Otherwise, the digital signature scheme likely has a lower true security strength.

Lastly, all entropy sources must be validated as cryptographic modules themselves.

> ***RECOMMENDATION**:* Use approved, validated, entropy sources.

## Canonicalization

A canonicalization algorithm deterministically maps input to a canonical form. Canonicalization algorithms are used on data that is to be hashed or signed by a digital signature. It is critical that users agree on the shared canonicalization algorithm beforehand. See the JSON Canonicalization Scheme [4] for the state of the art in canonicalization algorithms.

NIST does not specify canonicalization algorithms in the DSS, SHS, or SHA-3 standards. A flawed canonicalization implementation cannot leak private keys or any other information. However, canonicalization algorithms have direct cryptographic implications since an attacker could attain a forged signature from a user if that user employs an incorrect canonicalization implementation.

***RECOMMENDATION**:* Treat the system's canonicalization algorithm as a part of the hash function suite.

## Export Controls

The NIST FIPS PUBS for cryptographic algorithms note that certain cryptographic devices and technical data regarding them are subject to federal export controls. Exports of cryptographic modules implementing the standard and technical data regarding them must comply with these federal regulations and be licensed by the Bureau of Industry and Security of the U.S. Department of Commerce. Information about export regulations is available on their website [5].

# References


[1] 10 U.S.C. 2315, Law inapplicable to the procurement of automatic data processing equipment and services for certain defense purposes. https://www.govinfo.gov/app/details/USCODE-2019-title10/USCODE-2019-title10-subtitleA-partIV-chap137-sec2315

[2] 40 U.S.C 11103, Applicability to national security systems. https://www.govinfo.gov/app/details/USCODE-2019-title40/USCODE-2019-title40-subtitleIII-chap111-sec11103

[3] A. Langley, M. Hamburg, and S. Turner, Elliptic Curves for Security, RFC 7748, January 2016. https://www.rfc-editor.org/rfc/rfc7748.html

[4] A. Rundgren, B. Jordan, and S. Erdtman, JSON Canonicalization Scheme (JCS), RFC 8785, June 2020. https://www.rfc-editor.org/rfc/rfc8785

[5] Bureau of Industry and Security, U.S. Department of Commerce (webpage). http://www.bis.doc.gov/index.htm

[6] Bernstein, Daniel J., "Curve25519: new Diffie-Hellman speed records." International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2006. https://iacr.org/archive/pkc2006/39580209/39580209.pdf

[7] Bernstein, Daniel J., et al. "High-speed high-security signatures." Journal of cryptographic engineering 2.2 (2012): 77-89. https://doi.org/10.1007/s13389-012-0027-1

[8] Certicom Research, SEC 1: Elliptic Curve Cryptography, May 21, 2009, Version 2.0. https://www.secg.org/sec1-v2.pdf

[9] Certicom Research, SEC 2: SEC 2: Recommended Elliptic Curve Domain Parameters, January 27, 2010, Version 2.0. https://www.secg.org/sec2-v2.pdf

[10] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short group signatures." Annual international cryptology conference. Springer, Berlin, Heidelberg, 2004. http://crypto.stanford.edu/~dabo/pubs/papers/groupsigs.pdf

[11] DHS S&T, Blockchain Portfolio (webpage). https://www.dhs.gov/science-and-technology/blockchain-portfolio

[12] DHS S&T, Silicon Valley Innovation Program (webpage). https://www.dhs.gov/science-and-technology/svip

[13] Dustin Moody, Rene Peralta, Ray Perlner, Andrew Regenscheid, Allen Roginsky, and Lily Chen, Report on Pairing-Based Cryptography, Journal of Research of the National Institute of Standards and Technology, Volume 120 (2015). http://dx.doi.org/10.6028/jres.120.002

[14] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, August 2018. https://www.rfc-editor.org/rfc/rfc8446

[15] Eric Thormarker, On using the same key pair for Ed25519 and an X25519 based KEM, Ericsson. https://eprint.iacr.org/2021/509.pdf

[16] Grover, Lov K., "A fast quantum mechanical algorithm for database search," in Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing, July 1996. https://doi.org/10.1145/237814.237866

[17] ISO/IEC, Information Technology — Security Techniques — Security Requirements for Cryptographic Modules, 19790:2012, August 2012. https://www.iso.org/standard/52906.html

[18] Joachim Breitner and Nadia Heninger, Biased Nonce Sense: Lattice Attacks Against Weak ECDSA Signatures in Cryptocurrencies, Financial Cryptography 2019. https://eprint.iacr.org/2019/023

[19] Kathleen M. Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch, PKCS #1: RSA Cryptography Specifications Version 2.2, RFC 8017, November 2016. https://www.rfc-editor.org/rfc/rfc8017.html

[20] M. Lochter and J. Merkle, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, RFC 5639, March 2010. https://datatracker.ietf.org/doc/html/rfc5639

[21] Man Ho Au, Willy Susilo, and Yi Mu, Constant-Size Dynamick-TAA, International conference on security and cryptography for networks. Springer, Berlin, Heidelberg, 2006. https://eprint.iacr.org/2008/136.pdf

[22] Mathew Green, EUF-CMA and SUF-CMA. https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/

[23] NCC, Zcash Overwinter Consensus and Sapling Cryptography Review, January 30, 2019. https://research.nccgroup.com/wp-content/uploads/2020/07/NCC_Group_Zcash2018_Public_Report_2019-01-30_v1.3.pdf

[24] NIST, Advanced Encryption Standard (AES), FIPS PUB 197. https://doi.org/10.6028/NIST.FIPS.197

[25] NIST, Automated Cryptographic Validation Testing (ACVT) (webpage). https://csrc.nist.gov/Projects/Automated-Cryptographic-Validation-Testing

[26] NIST, CMVP Approved Authentication Mechanisms: CMVP Validation Authority Requirements for ISO/IEC 19790 Annex E and ISO/IEC 24579 Section 6.17, SP 800-140E, March 2020. https://csrc.nist.gov/publications/detail/sp/800-140e/final

[27] NIST, CMVP Approved Non-Invasive Attack Mitigation Test Metrics: CMVP Validation Authority Updates to ISO/IEC 24759, Special Publication 800-140F, March 2020. https://csrc.nist.gov/publications/detail/sp/800-140e/final

[28] NIST, CMVP Approved Security Functions: CMVP Validation Authority Updates to ISO/IEC 24759, Special Publication 800-140C, March 2020. https://csrc.nist.gov/publications/detail/sp/800-140c/final

[29] NIST, CMVP Approved Sensitive Parameter Generation and Establishment Methods: CMVP Validation Authority Updates to ISO/IEC 24759, Special Publication 800-140D, March 2020. https://csrc.nist.gov/publications/detail/sp/800-140d/final

[30] NIST, CMVP Documentation Requirements: CMVP Validation Authority Updates to ISO/IEC 24759, Special Publication 800-140A, March 2020. https://csrc.nist.gov/publications/detail/sp/800-140a/final

[31] NIST, CMVP Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B, Special Publication 800-140B, March 2020. https://csrc.nist.gov/publications/detail/sp/800-140b/final

[32] NIST, Crypto Publication Review Project (webpage). https://csrc.nist.gov/projects/crypto-publication-review-project

[33] NIST, Cryptographic Algorithm Validation Program (CAVP) (webpage). https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program

[34] NIST, Cryptographic Algorithm Validation Program Validation Search (webpage). https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program/validation-search

[35] NIST, Cryptographic Module Validation Program (CMVP) (webpage). https://csrc.nist.gov/projects/cryptographic-module-validation-program

[36] NIST, Digital Signature Standard (DSS), FIPS PUB 186-4. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[37] NIST, Digital Signature Standard Draft (DSS), FIPS PUB 186-5. https://doi.org/10.6028/NIST.FIPS.186-5-draft

[38] NIST, Escrowed Encryption Standard (Withdrawn), FIPS PUB 185, February 1994. https://csrc.nist.gov/csrc/media/publications/fips/185/archive/1994-02-09/documents/fips185.pdf

[39] NIST, FIPS 140-3 Derived Test Requirements (DTR): CMVP Validation Authority Updates to ISO/IEC 24759, Special Publication 800-140, March 2020. https://csrc.nist.gov/publications/detail/sp/800-140/final

[40] NIST, Getting Ready for Post-Quantum Cryptography: Exploring Challenges Associated with Adopting and Using Post-Quantum Cryptographic Algorithms, April 2021. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04282021.pdf

[41] NIST, Guideline for Implementing Cryptography in the Federal Government, Special Publication 800-21 Second Edition, December 2005. https://csrc.nist.gov/publications/detail/sp/800-21/second-edition/archive/2005-12-01

[42] NIST, Guideline for Using Cryptographic Standards in the Federal Government: Directives, Mandates and Policies, Special Publication 800-175A, August 2016. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175A.pdf

[43] NIST, Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms, Special Publication 800-175B Revision 1, March 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175Br1.pdf

[44] NIST, Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations, Special Publication 800-52 Revision 2, August 2019. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf

[45] NIST, Implementation Under Test List, Cryptographic Module Validation Program (webpage). https://csrc.nist.gov/Projects/cryptographic-module-validation-program/modules-in-process/Modules-In-Process-List

[46] NIST, Modules in Process List, Cryptographic Module Validation Program (webpage). https://csrc.nist.gov/Projects/cryptographic-module-validation-program/modules-in-process/Modules-In-Process-List

[47] NIST, National Voluntary Laboratory Accreditation Program (NAVLAP) (webpage). https://www.nist.gov/nvlap

[48] NIST, Recommendation for Applications Using Approved Hash Algorithms, Publication 800-107 Revision 1, August 2012. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf

[49] NIST, Recommendation for Block Cipher Modes of Operation Methods and Techniques, Special Publication 800-38A, December 2001. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

[50] NIST, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, Special Publication 800-38D, November 2007. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

[51] NIST, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, Special Publication 800-38F, December 2012. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38f.pdf

[52] NIST, Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption, Special Publication 800-38G, March 2016. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38G.pdf

[53] NIST, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, Special Publication 800-38C, May 2004. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

[54] NIST, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication 800-38B, May 2005. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf

[55] NIST, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, Special Publication 800-38E, January 2010. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf

[56] NIST, Recommendation for Key Derivation Methods in Key-Establishment Schemes, Special Publication 800-56C Revision 2, August 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf

[57] NIST, Recommendation for Key Management: Part 1 – General, Special Publication 800-57 Part 1 Revision 5, May 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5

[58] NIST, Recommendation for Key Management Part 2 – Best Practices for Key Management Organizations, Special Publication 800-57 Part 2 Revision 1, May 2019. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf

[59] NIST, Recommendation for Key Management Part 3: Application-Specific Key Management Guidance, Special Publication 800-57 Part 3 Revision 1, January 2015. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf

[60] NIST, Recommendation for Key-Derivation Methods in Key-Establishment Schemes, Special Publication 800-56C Revision 2, August 2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf

[61] NIST, Recommendation for Obtaining Assurances for Digital Signature Applications, Special Publication 800-89, November 2006. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-89.pdf

[62] NIST, Recommendation for Pair-Wise Key Establishment Using Discrete Logarithm Cryptography Special Publication 800-56A Revision 3, April 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf

[63] NIST, Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography, Special Publication 800-56B, March 2019. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf

[64] NIST, Recommendation for Random Bit Generator (RBG) Constructions, Special Publication 800-90C, April 2016. https://csrc.nist.gov/CSRC/media/Publications/sp/800-90c/draft/documents/sp800_90c_second_draft.pdf

[65] NIST, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication 800-90A Revision 1, June 2015. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

[66] NIST, Recommendation for the Entropy Sources Used for Random Bit Generation, Special Publication 800-90B, January 2018. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

[67] NIST, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication 800-67 Revision 2, November 2017. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-67r2.pdf

[68] NIST, Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters, Special Publication 800-186, October 2019. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-186-draft.pdf

[69] NIST, Report on Post-Quantum Cryptography, April 2016. https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf

[70] NIST, Review of the Advanced Encryption Standard, NISTIR 8319, July 2020. https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8319.pdf

[71] NIST, Risk Management Framework, Federal Information Security Modernization Act (FISMA) Background (webpage). https://csrc.nist.gov/projects/risk-management/fisma-background

[72] NIST, Secure Hash Standard (SHS), FIPS PUB 180-4. https://doi.org/10.6028/NIST.FIPS.180-4

[73] NIST, Security Requirements for Cryptographic Modules, FIPS PUB 140-2. https://csrc.nist.gov/publications/detail/fips/140/2/final

[74] NIST, Security Requirements for Cryptographic Modules, FIPS PUB 140-3. https://doi.org/10.6028/NIST.FIPS.140-3

[75] NIST, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash, Special Publication 800-185, December 2016. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf

[76] NIST, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (SHA-3), FIPS PUB 202. https://doi.org/10.6028/NIST.FIPS.202

[77] NIST, SKIPJACK and KEA Algorithm Specification, Version 2.0, May 1998. https://csrc.nist.gov/CSRC/media//Projects/Cryptographic-Algorithm-Validation-Program/documents/skipjack/skipjack.pdf

[78] NIST, Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process, July 2020. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf

[79] NIST, The Keyed-Hash Message Authentication Code (HMAC), FIPS PUB 198-1. https://doi.org/10.6028/NIST.FIPS.198-1

[80] NIST, Transitioning the Use of Cryptographic Algorithms and Key Lengths, Special Publication 800-131A Revision 2, March 2019. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf

[81] NIST, Validated Modules, Cryptographic Module Validation Program (webpage). https://csrc.nist.gov/Projects/cryptographic-module-validation-program/validated-modules

[82] Public Law 100-235, Computer Security Act of 1987, January 8, 1988. https://www.govinfo.gov/app/details/STATUTE-101/STATUTE-101-Pg1724

[83] Public Law 104-106, National Defense Authorization Act for Fiscal Year 1996, February 10, 1996. https://www.govinfo.gov/app/details/PLAW-104publ106

[84] Public Law 107-347, E-Government Act of 2002, December 17, 2002. https://www.govinfo.gov/app/details/PLAW-107publ347

[85] Public Law 113-283, Federal Information Security Modernization Act of 2014, December 18, 2014. https://www.govinfo.gov/app/details/PLAW-113publ283/

[86] S. Josefsson and I. Liusvaara, Edwards-Curve Digital Signature Algorithm (EdDSA), RFC 8032, January 2017. https://datatracker.ietf.org/doc/html/rfc8032

[87] T. Pornin, Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), RFC 6979, August 2013. https://datatracker.ietf.org/doc/html/rfc6979

[88] Tobias Looker and Orie Steele, BBS+ Signatures 2020, W3C, June 2021. https://w3c-ccg.github.io/ldp-bbs2020/

[89] W3C, Decentralized Identifiers (DIDs) v1.0: Core architecture, data model, and representations, W3C Candidate Recommendation, 17 July 2021. https://www.w3.org/TR/did-core/

[90] W3C, Verifiable Credentials Data Model 1.0: Expressing verifiable information on the Web, W3C Recommendation, 19 November 2019. https://www.w3.org/TR/vc-data-model/