**Cryptographic Algorithms and Key Rotation Guideline**

This article provides high level overview and defines a set of general guidelines for ABC Financial Services around the use of use of encryption, hashing, certificates, digital signatures, and other data protection methods used to ensure the confidentiality and integrity of the Company's information assets.

**1. Preface**

This document contains directives to carry out the requirements within the Company's BT Cybersecurity Standards. Company employees, contractors, contingent workers, affiliates, vendors, and third parties who manage or use the Company's information resources must comply with the controls herein.

**2. Purpose & Scope**

This document provides a high level overview and defines a set of general guidelines for ABC Financial Services around the use of use of encryption, hashing, certificates, digital signatures, and other data protection methods used to ensure the confidentiality and integrity of the Company's information assets. This document does not provide directives on when to use a cryptographic function. The guidance from industry best practices, PCI version v3, and NIST were considered when evaluating the direction of the enterprise. This document supplements the following policies and standards:

* [Information Asset Protection Policy (POL-237398)](https://discoverfinancial.sharepoint.com/sites/DLife/bu/crm/policies/enterprisewidetier1and2policies/Documents/Information%20Asset%20Protection%20Policy.pdf)
* [Operations Information Security Standard (STD-384873)](https://discoverfinancial.sharepoint.com/sites/DLife/bu/crm/policies/standardsunderconstruction/Documents/Operations%20Information%20Security%20Standard.pdf)
* [Encryption and Key Management Standard (STD-2988979)](https://discoverfinancial.sharepoint.com/sites/DLife/bu/crm/policies/standardsunderconstruction/Documents/Encryption%20and%20Key%20Management%20Standard.pdf)

**3. Cryptographic Algorithms Guidelines**

Cryptography can provide confidentiality, integrity, authenticity, and non-repudiation for applications, network communications, storage and more. Selection of the cryptographic and key management algorithms to use within a use case needs to begin with an understanding of the objectives of the use case. In most cryptographic functions, the key length is an important security parameter. Choosing an appropriate key size to protect your application or system from attacks is of utmost important to fully take advantage of cryptographic functions. However, ensuring the continued effectiveness of the cryptographic function without proper key security and lifecycle will weaken the cryptographic function. Breakdown in selection of an algorithm and key lifecycle management can also lead to data availability issues resulting in business impact or render the cryptographic function useless.

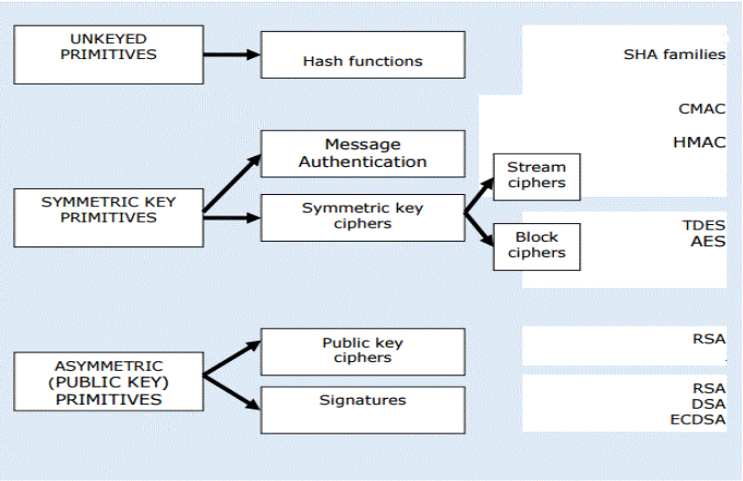


Fig . Cryptographic primitives and mapping to mechanisms

**3.1 Encryption**

NIST defines Encryption as process of changing plaintext into cipher text (scrambled value) for the purpose of security or privacy. There are two types of encryption; Symmetric and Asymmetric.

**Symmetric Key Encryption**- Is used to ensure confidentiality. It uses a single key to perform both, encryption and decryption functions. Symmetric encryption is used to encrypt large amounts of data in a high speed manner.

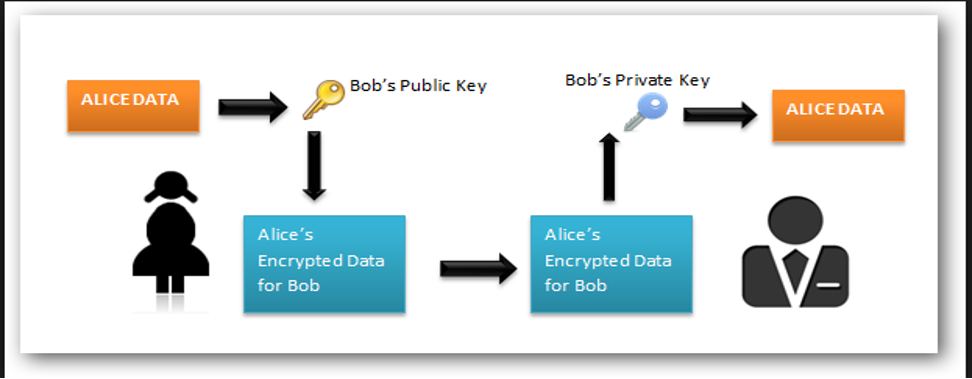
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Approved Symmetric Data Encryption Algorithm Table | | | | | |
| Algorithm | Approved Key Strength [2] | Crytoperiod [2]  Key Rotation | Modes | Restrictions | Key Exchange |
| AES | 256 | <=2 years | **CBC (Not Recommended)**  GCM (Authentication) |  | Asymmetric Encryption (KEK) (preferred) |
| PGP + Email Encryption |
| Delete email once key is received) |
| Triple DES  **(Deprecated)** | 112 | <=2 years | ANSI X9.52 Approved | **Limited to PIN Security** |  |

**\*-** AES 128 cipher is acceptable to be used in TLS - see section on TLS Ciphers

**Guidelines on Approved Symmetric-Key Algorithms**

* Keys must be generated by an approved source or solution such as DFS' Key Management Solution
  + Application teams must not generate their own keys in production environments
* Symmetric Keys must be unique and used for single purpose and single relationship (E.g. Application X can share key with application Y, but not application C)
* Where technically feasible, use authenticated encryption
* Initialization vector (IV) used for execution of encryption process must be unpredictable and unique
* Key exchange must be done via secure channel
* Keys for encrypting keys must be used separately from keys used for encrypting and decrypting data
* Where required, implement process that require split knowledge and dual control of keys
* Use of keys must be documented with application name, owner and scheduled rotation period
* Keys must be backed up and stored in approved, hardened location (e.g. Password-protected keystores such as PKCS#12, Key Management Solution or HSM)
* Cryptography keys must not be stored in the clear. At a minimum, use key store and implement least privilege access and strong authentication

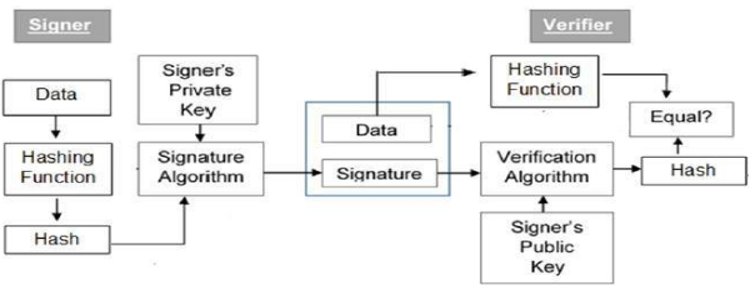
**Asymmetric Key Encryption-** Asymmetric-key encryption use a pair of keys: a public key and a private key to perform encryption and decryption. The public key may be made public without reducing the security of the process, but the private key must remain secret/secure if the data is to retain its cryptographic protection.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Asymmetric Data Encryption Algorithm Table | | | | |
| Algorithm | Approved Key Strength [2] | Crytoperiod [2]  Key Rotation | Recommendation | Public Key Exchange |
| RSA | 2048 | <=2 Years | Not recommended for large data. Use only when Symmetric Key encryption is not suitable option | Secure Portal  PGP + Email Encryption - (for any private key sharing)  Email - (For sharing Public Keys) |

**3.2 Digital Signatures**

A digital signature algorithm is part of Asymmetric-Key cryptography. It uses a private key and a public key to generate and verify digital signatures. The private key is used to generate signatures and must be known only to the signer (the key-pair owner); the public key is used to verify the signatures

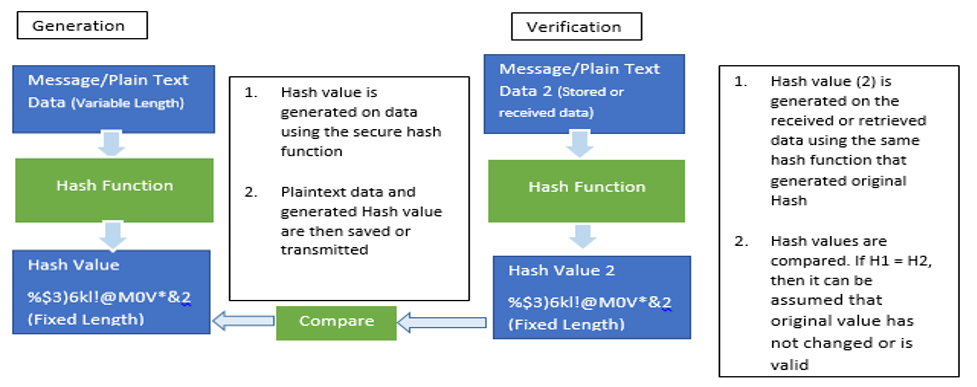


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Approved Digital Signature Algorithm Table | | | | |
| Algorithm | Approved Key Strength [2] | Rotation Period [2] | Public Key Exchange Method |  |
| RSA | 2048  3072  4096 | 1 year - for internal/external browser endpoint certificates and app certificates used for JWE/JWS  5 year - for internal API/infrastructure endpoints | Email Encryption |  |
| ECDSA | 256 & 384 | 1 year - for internal/external browser endpoint certificates and app certificates used for JWE/JWS  5 year - for internal API/infrastructure endpoints | Email Encryption |  |

**Guidelines on Approved Asymmetric-Key Algorithms**

* Asymmetric-Keys should be retricted to a single purpose (e.g., one key pair for generating and verifying digital signatures, and a different key pair for key establishment or encryption)
* Keys pairs must be generated by an approved solution such as DFS' Key Management Solution
* Use of keys must documented with application name, owner and scheduled rotation period with key provider
* Private Keys must be backed up and stored in approved, hardened location (e.g. Password-protected keystores such as PKCS#12, Key Management Solution or HSM)
* Avoid storing cryptographic keys locally. At a minimum, use key store and implement least privilege access and strong authentication
* Certificates must be signed by an approved Certificate Authority
* Certificates used for external use case (i.e. public facing apps, B2B authentication) must be signed by the company approved external Certificate Authority

**3.3 Hashing**

Hashing is an irreversible mathematical function that scrambles data, producing a consistent and unique value. It is a one-way function. It is generally used to ensure integrity and verify data. 

**Guidelines on Approved Hashing Algorithms**

|  |  |  |  |
| --- | --- | --- | --- |
| Approved Hashing Algorithms Table | | | |
| Algorithm | Approved Key Strength [2] | Restrictions | Comments |
| SHA2  SHA3 | 256  384  512 | Do not use hashing algorithm alone for purpose of securing data such as passwords. Use unique salt value per record with length of 64 or minimum 32 byte | PDKDF2 is recommended for securing passwords within application. |

**Note**: SHA2 algorithms listed are still secure. However, SHA-3 hash functions are alternatives to the SHA-2 functions, and they are designed to provide resistance against collision, preimage attacks that equals or exceeds the resistance that the corresponding SHA-2 functions provide.

**3.4 Message Authentication Codes**

Message Authentication Codes (MAC) are keys used to ensure message authenticity. It is a symmetric key cryptographic technique. Generally, MACs are combined with hash function to provide integrity along with authenticity.

|  |  |  |
| --- | --- | --- |
| Approved MAC Algorithm | | |
| Algorithm | Hashing Function Combination | Comments |
| HMAC | SHA256, SHA384, SHA512 | HMAC-SHA256 is recommended |

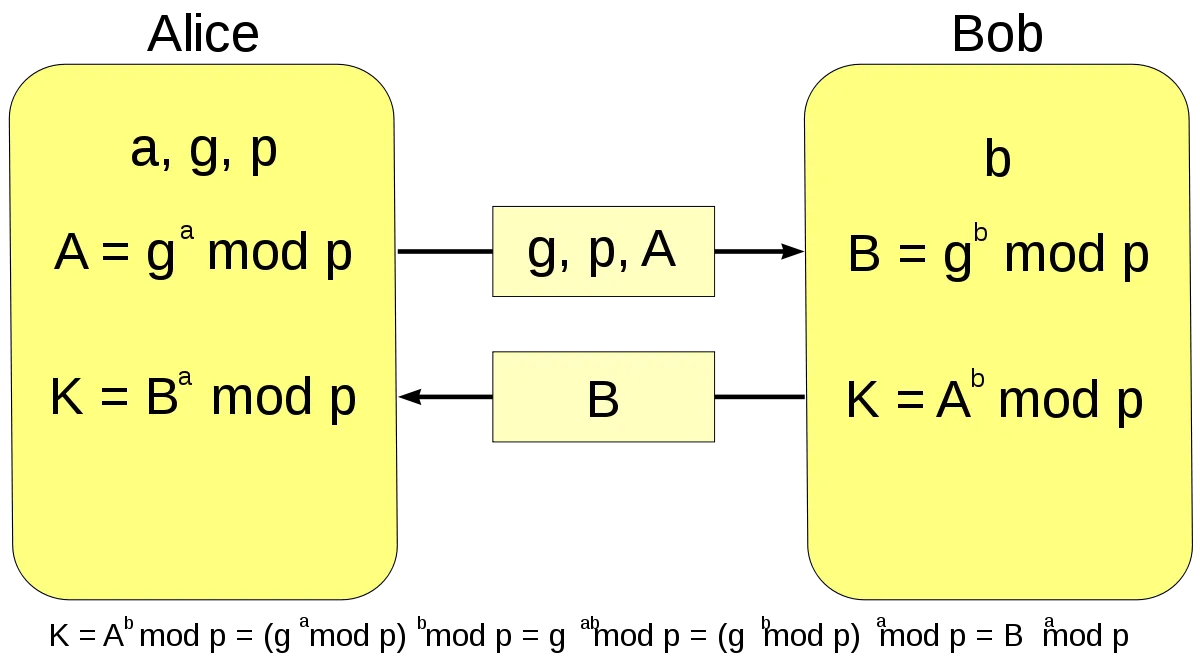
**Note:** This is an approved algorithm for use in scenarios that calls for use of HMAC. For use in JWS, refer [5].

**3.5 Key Establishment/Exchange**

Key establishment/exchange comprises of key agreement or key transport.

**Key Agreement**

Key agreement involves the two parties coming up with the key in an authenticated manner such that both parties contribute to the key.  
Key agreement involves variations of the Diffie-Hellman (DH) protocol. DH allows two parties to come up with a shared secret without exchanging the secret over an insecure channel. The figure below captures the essence of this.



Alice selects the common parameters 'g' and 'p' and generates a random number 'a' and computes 'A'. Alice, then sends, 'g', 'p', and 'A' to Bob. Bob generates the random number 'b' and computes 'B' as shown. He sends 'B' to Alice. Both sides then generate 'K'; which is the shared secret. In the context of TLS where this is typically used, Alice and Bob represent the TLS Server and the TLS Client, respectively.

|  |  |  |
| --- | --- | --- |
| Approved Key Agreement Algorithms | | |
| Protocol | Authentication Combination | Comments |
| ECDHE | RSA\* | ECDHE-RSA is recommended |
| DHE | RSA\* | DHE-RSA is recommended |

\**Refer to Section 3.2 Digital Signatures for RSA key size recommendations, as the key agreement ciphers typically do not specify the RSA key size*

DHE, is Diffie-Hellman that is implemented using modular arithmetic as per the explanation of DH above. The final E stands for Ephemeral. It implies that unique keys are generated every session; thereby allowing for Perfect Forward Secrecy (PFS). PFS refers to an encryption technique that uses unique session keys for each session, so that compromise of a session key does not impact all past and future sessions.

ECDHE refers to Diffie-Hellman wherein the shared key is established using elliptic curve cryptography (ECC). Each party generates their own ECC key pair with the domain parameters for the curve being agreed upon. They, then, exchange their respective public keys and compute the same point on the curve using their secret key and the other party's public key. The x-coordinate of this point is the shared secret. The final E standing for Ephemeral and has the same implication as explained for DHE.

Note that both ECDHE and DHE are subject to main-in-the-middle attacks, if unauthenticated. For example, Bob does not know if he receives 'A' from Alice or from a rogue hacker, as per the DH flow shown in the figure above.

This is where RSA comes into the cipher. For instance, as part of the TLS handshake, the Server Key Exchange contains the domain parameters to be used in ECHDE or DHE which is signed using the private key of the Server. The client verifies this signature using the public key in the Server Certificate; thereby authenticating that the common parameters were indeed from the Server and not a man-in-the-middle. Refer the section on 3.2 Digital Signatures for guidelines around signatures using RSA.

**Key transport**

Key transport involves one party generating the key and then sending it to the other party by encrypting it. This translates to key encryption. It can be performed using a symmetric key or using asymmetric key.

There are no specific recommendations for key transport using either symmetric key or asymmetric key other than the recommendations under 3.1 Encryption.

Key agreement is preferred over key transport due to the Perfect Forward Secrecy present in the key agreement algorithms recommended.

**3.6 Cryptographic Libraries and Solutions**

* Only reputable libraries and solutions that are well tested, maintained and updated must be used for cryptographic purposes.
* Cryptography solutions must be approved by Cybersecurity and Architecture Review Board

**3.7 Other Keys and Cryptoperiod**

|  |  |  |
| --- | --- | --- |
| Other Cryptography Key type Cryptoperiod Table | | |
| Key Type | Usage Period | Decryption Usage Period |
| Symmetric Master Key | <=2 years | Usage period + Data Retention Period |
| Symmetric Key Wrapping Key | <=2 years | Usage period + Data Retention Period |
| Key Encrypting Key | <=2 years | Usage period + Data Retention Period |

**3.8 TLS Configuration**

**Configuration Common to TLS Servers, TLS Clients**

TLS version 1.2 must be used in all applications and infrastructure components using TLS.

TLS v 1.3 may be used after review by Cyber Security architecture team on a case by case basis. Contact [CybersecurityArchitecture@ABC.com](mailto:CybersecurityArchitecture@discover.com), if TLS v1.3 is required.

TLS ciphers are a combination of algorithms. Cipher suites in TLS 1.2 have the form:

TLS\_KeyExchangeAlg\_WITH\_EncryptionAlg\_MessageAuthenticationAlg.

Cipher suites in TLS 1.3 are different and have the form:

TLS\_AEAD\_HASH

**Configuration - TLS Server**

The table below lists the NIST-approved ciphers for TLS 1.2 and TLS 1.3 as listed in NIST SP 800-52 [4] (latest version to date). Some of these ciphers are not included in ABC IRM Policies and Standards. However, these may be needed when configuring TLS servers for compatibility with other applications and browsers. These are marked with an asterisk.

| TLS 1.2 Ciphers | TLS 1.3 Ciphers |
| --- | --- |
| **For use with RSA certificates:**   * TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384 * TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 * TLS\_DHE\_RSA\_WITH\_AES\_256\_CCM * TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256 * TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256\* * TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256\* * TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256\* * TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256\* * TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 \* * TLS\_DHE\_RSA\_WITH\_AES\_128\_CCM\* * TLS\_DHE\_RSA\_WITH\_AES\_128\_CCM\_8\* * TLS\_DHE\_RSA\_WITH\_AES\_256\_CCM\_8 \*   **For use with ECDSA certificates:**   * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM * TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256\* * TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256\* * TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\* * TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8\* * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8 | **For use with RSA or ECDSA certificates:**   * TLS\_AES\_256\_GCM\_SHA384 * TLS\_AES\_128\_GCM\_SHA256\* * TLS\_AES\_128\_CCM\_SHA256\* * TLS\_AES\_128\_CCM\_8\_SHA256\* |

These ciphers are to be always configured in the order listed above since some TLS stacks use the ordering during the negotiation of cipher suites.

Further, if the tool supports it, a specific configuration setting corresponding to "TLS Cipher Suite ordering" is to be applied to ensure that stronger suites are selected first from among the above cipher suites.

Additionally, certain components support a configuration that turns on TLS server side cipher preferences (provided the client, e.g. the browser, supports this cipher). For instance, nginx has a configuration setting

*ssl\_prefer\_server\_ciphers on;*

**Configuration - TLS Clients**

TLS Clients can have a more restricted set of TLS Ciphers configured in adherence to ISPO Policies as it does not have the same compatibility requirements as that of a TLS Server.

| TLS 1.2 Ciphers | TLS 1.3 Ciphers |
| --- | --- |
| **For use with RSA certificates:**   * TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384 * TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384   **For use with ECDSA certificates:**   * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM * TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8 | **For use with RSA or ECDSA certificates:**   * TLS\_AES\_256\_GCM\_SHA384 |

Above ciphers are to be configured in the order listed.

**4. References**

1. DFS IRM, "IRM Policies and Standards", [IRM](https://discoverfinancial.sharepoint.com/sites/DLife/bu/crm/Information%20Security%20and%20Technology%20Risk/ISPO%20Policies%20and%20Standards/Pages/IRM%20Policies%20and%20Standards.aspx)
2. NIST Special Publication (SP) 800-57, Rev 5, "Recommendation for Key Management: Part 1 - General", May 2020, [NIST SP 800-57](https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final)
3. DFS Cybersecurity Architecture, "Recommended Cryptographic Cipher suites for JSON Web Encryption (JWE) and JSON Web Signature (JWS)", [JWE and JWS](https://discoverfinancial.sharepoint.com/:b:/r/sites/InfoSecServ/CSSA/Reference%20Architectures%20Patterns%20%20Guidelines/Reference%20Architectures,%20Patterns%20and%20Guidelines%20Catalog/Data%20Security/CS%20Architecture%20-%20JWE%20and%20JWS%20Crypto%20Algorithms%20Recommendations.pdf?csf=1&web=1&e=kwL0ui)
4. NIST Special Publication (SP) 800-52 Rev 2, Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations, [NIST SP 800-52](https://doi.org/10.6028/NIST.SP.800-52r2)
5. European Payment Council, "Guidelines on cryptographic algorithms usage and key management", [EPC Document](https://www.europeanpaymentscouncil.eu/sites/default/files/kb/file/2020-03/EPC342-08%20v9.0%20Guidelines%20on%20Cryptographic%20Algorithms%20Usage%20and%20Key%20Management_0.pdf)
6. NIST Special Publication (SP) 800-175B, Rev 1, Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms, March 2020, [NIST SP 800-175B](https://doi.org/10.6028/NIST.SP.800-175Br1)

[Reuse](https://dta.discoverfinancial.com/product/openworx/reuse)

[Architect](https://dta.discoverfinancial.com/product/craftworx/think/architect)

[Security](https://dta.discoverfinancial.com/technologies/security)

**Content**

1. [1. Preface](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#1-preface)
2. [2. Purpose & Scope](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#2-purpose--scope)
3. [3. Cryptographic Algorithms Guidelines](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#3-cryptographic-algorithms-guidelines)
   1. [3.1 Encryption](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#31-encryption)
   2. [3.2 Digital Signatures](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#32-digital-signatures)
   3. [3.3 Hashing](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#33-hashing)
   4. [3.4 Message Authentication Codes](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#34-message-authentication-codes)
   5. [3.5 Key Establishment/Exchange](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#35-key-establishmentexchange)
   6. [3.6 Cryptographic Libraries and Solutions](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#36-cryptographic-libraries-and-solutions)
   7. [3.7 Other Keys and Cryptoperiod](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#37-other-keys-and-cryptoperiod)
   8. [**3.8 TLS Configuration**](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#38-tls-configuration)
4. [4. References](https://dta.discoverfinancial.com/architectures/cryptography-algorithm-guide#4-references)