European Payments Council

Guidelines on cryptographic algorithms usage and key management - Approved

EPC342-08 / Version 15.0 / Produced by PSSG / Date issued: 7 March 2025

This document defines guidelines on cryptographic algorithms usage and key management.

Guidelines

Cryptographic algorithms usage and key management

EPC342-08
2025 version 15.0
Date issued: 7 March 2025

European Payments Council

European Payments Council AISBL,
Cours Saint-Michel 30 B-1040 Brussels
T +32 2 733 35 33
Enterprise N°0873.268.927
secretariat@epc-cep.eu

Table of Contents

List of figures

List of tables

Executive Summary

The **purpose and scope** of this document is to provide guidance to the European payments industry in the field of cryptographic algorithms and related key management practices. It has been written mainly for payment service providers, specifically for security officers, risk managers, system engineers and system designers. Although its reading does not assume expertise in cryptology, an understanding of key concepts of cryptography is assumed and some sections require basic mathematical knowledge.

The **cryptographic landscape** is constantly evolving. New algorithms are proposed and existing ones are challenged, such that recommendations on cryptographic algorithms and key management are prone to become obsolete. Also, it is usually an over-simplification to categorise an algorithm or key management technique as being either 'good' or 'bad'. In practice, some are weaker or stronger depending on how and for what they are used and on the function of the attack technique being evaluated.

The **choice of a cryptographic technique** should always be the result of a risk assessment process. This process should not only consider the potential loss in case the cryptographic technique fails to deter an attack. The assessment should also contemplate the resilience of the technique in face of diverse attack vectors (e.g. plaintext/ciphertext attacks), the operational conditions that may be more or less favourable for some kinds of attack (e.g. sample of plaintexts and matching ciphertexts) and the progress in computational power available to an adversary. Time constraints should also be factored in so as to assure that the cryptographic technique is adequate to protect the data for the required time. Special care should be taken in the use of a cryptographic algorithm which is considered as weak by the specialists, given that, even though it may have no consequences in a given context, it may result in reputational impacts for the financial institution.

Cryptographic agility by-design should be considered so as to allow possible algorithm migration in the future. The cost and difficulty of migrating from one algorithm to another (or of changing the size of the keys) should not be underestimated. Incorporating agility features into the cryptographic architecture should be seriously considered given the undeniable benefit in case of need. This is specially critical given that never before has the cryptographic landscape been so dynamic, resulting from advancements in computational power.

A list of **recommendations** on cryptographic algorithms, security protocols, confidentiality and integrity protection and key management can be found in the Recommendations section 1.3, for which further detailed background information may be found in the subsequent sections of the document. It is crucial to remember that while algorithm selection, key sizes, and key material randomness are fundamental, other critical security implementation aspects exist, such as side-channel countermeasures and protocol layer interdependency checking. These aspects, although beyond the scope of this document, are essential for a robust cryptographic implementation.

In producing **these guidelines**, the EPC aims to provide a reference basis to support payment service providers. However, it needs to be recognised that cryptology research and development are constantly evolving. Therefore, the EPC plans to annually review and, with best endeavours, update the document to reflect the state of the art in light of major new developments and to keep it aligned with the documents referenced. This new version provides an overall review of referred standards and algorithms. Attention was given to update the information regarding cryptography in the context of quantum computing. <u>Although no one knows precisely when cryptographically relevant quantum computers will actually arrive,</u> organizations must review their cryptographic strategies to ensure they remain secure in a post-quantum world.

1 Introduction

1.1 Scope of the document

This document is aimed to provide guidance to the European payments community on algorithm usage and key management issues.

It contains some recommendations from EPC on algorithm usage and key management issues that the payment service providers may consider together with their own security policy and the relevant professional or national rules and regulations they have to comply with.

These guidelines recommend use of International Standards where appropriate.

It also addresses the points that should be considered whenever payment service providers wish to provide interoperable services based on cryptographic mechanisms. These points may be of particular interest for secured cross-border services.

The scope of this document is limited to cryptographic algorithms and key management. Amongst the mechanisms excluded from its scope are:

- error detecting mechanisms such as Cyclic Redundancy Check,
- data compression facilities such as Zip or Huffman coding,
- side-channel countermeasures and protocol layer interdependency checking,
- secret algorithms, for which no technical features are available.

The world of cryptography being wide and rapidly expanding, this document focuses on algorithms which are suitable for payment services, and which are already adopted by the financial industry or which are likely to be in the foreseeable future.

In order to cope with the rapid evolution of the technology, this report is updated yearly.

Several EPC experts have participated in the development of this report over time. The contributors to the 2024-2025 update are:

- Alain Hiltgen (UBS Business Solutions AG)
- Boris Hemkemeier (Commerzbank)
- Dimitrios Markakis (Electronic Money Association)
- Jurjen Bos (Worldline)
- Michael Ward (Mastercard)
- Valentim Oliveira (SIBS)
- Dirk De bruyn (EPC Secretariat)
- Valentin Vlad (EPC Secretariat)

1.2 Document structure

Section 1 specifies the scope of this document, contains the list of “Recommendations” and describes the structure of the document. Those recommendations are further elaborated in the remainder of the document.

Section 2 provides an introduction to cryptographic primitives and a taxonomy for cryptographic algorithms and their typical usage.

Section 3 discusses design issues for constructing cryptographic mechanisms from cryptographic primitives and describes implementation and interoperability issues.

Section 4 deals with key management for symmetric and asymmetric algorithms and introduces the topic of key escrow and key recovery.

Section 5 treats briefly the generation of random numbers.

ANNEX I contains definitions of terms used in this document and lists most of the acronyms used.

ANNEX II contains a bibliographical list of the textbooks, publications or standards (either international, national or de facto) referenced.

1.3 Recommendations

This section summarises the recommendations made throughout the document, in the order they appear in the document. Further background information to these recommendations may be found in the main sections of the document:

Crypto algorithms	
Rec 1	Only algorithms whose specifications have been publicly scrutinised (ideally with a public design phase), and whose strength has been assessed by crypto experts can be recommended. Algorithms specified in International Standards should be preferred. This recommendation also applies to algorithms for key generation.
Rec 2	• AES is the recommended standard for new systems. • 3TDES is still secure in use cases where there is no concern regarding reduced block sizes. • 2TDES may still be sufficiently secure for existing systems under specific conditions (see 0); plans should be made to migrate to AES. • Single DES (56 bits) should be considered broken. Further details are provided in sections 3.1.2.4 and 0.
Rec 3	• No longer use RSA keys of 768 bits and ECC keys of 130 bits, or less. • Avoid using 1024-bit RSA keys and 160-bit ECC keys for new applications unless for short-term, low-value protection (e.g. ephemeral authentication for single devices). • Use at least 2048-bit RSA or 224-bit ECC for medium-term (e.g. 10 year) protection (see [174]). • For considerations regarding low exponent RSA, section 3.1.3 should be consulted. Further details are provided in section 0.
Rec 4	In view of the current published progress of quantum computing initiatives, and especially for example concerns regarding Harvest Now Decrypt Later, public key cryptography policy is significantly impacted; however, as noted by UK NCSC [208], policy related to symmetric cryptography and hash functions is not significantly impacted. Where public key cryptographic primitives are used, crypto agility - as recommended by NIST and BSI [204], [205] - should be integrated into the

<table>
<tr><td></td><td>cryptographic services' pipeline. Moreover, where practically feasible, hybrid key establishment and hybrid digital signature techniques, combining classical and post-quantum algorithms, are generally advisable as a transitional migration strategy.</td></tr>
<tr><td>Rec 5</td><td>Although many legacy systems currently still use MACs based on DES or TDES, new systems should use CMAC based on AES or HMAC.

Further details are provided in section 3.2.4.</td></tr>
<tr><td>Rec 6</td><td>Financial service providers that decide to deploy distributed ledger-based services or processes should:
Confirm the security properties afforded by the distributed ledger to its users.
Identify the cryptographic primitives used to validate and commit changes to the ledger and confirm the status of cryptanalytic attacks targeting these primitives.
Identify the consensus protocol used to commit changes to the ledger and assess the feasibility of a successful consensus hijack attack.
Further details are provided in section 3.2.8.</td></tr>
<tr><th colspan="2">Security protocols</th></tr>
<tr><td>Rec 7</td><td>Use TLS with secure cryptographic primitives and appropriate key sizes (c.f. 3.1.3.4).
Enable TLS 1.3 support in all new systems (offers forward-secrecy by default).
Enforce the use of TLS 1.2 or higher for all use cases (preferably with ephemeral cipher suites).
Do not use TLS versions older than TLS 1.2 because of known and exploitable vulnerabilities (unless such use is approved in specific use cases through ongoing security risk assessment).
Further details are provided in section 3.4</td></tr>
<tr><th colspan="2">Confidentiality and integrity protection</th></tr>
<tr><td>Rec 8</td><td>To achieve confidentiality and integrity protection:
For the originator, sign the plaintext data first for legal signature (if required), then compress (if required), then apply authenticated encryption with the non-confidential data being treated as associated data (not encrypted).
For the recipient, perform the steps in the reverse order. Verify that signatures are from an authentic source.
The encryption and signature/MAC can be performed as separate steps or can be achieved by use of authenticated encryption (which also allows the use of traditional encrypt-then-MAC) or signcryption.
Further details are provided in section 3.4.</td></tr>
<tr><th colspan="2">Key management</th></tr>
</table>

Rec 9	If a master key needs to be backed-up outside of a TRSM then it should be split into key components in a secure and resilient manner. Secret sharing techniques should be used to allow recovery of the master key from all or a fixed number (threshold) of the master key components. The security level of the storage of the master key components should be commensurate with the protection afforded the operational master key itself. Further details are provided in section 4.1.2.
Rec 10	Symmetric keys should be dedicated to one usage (e.g. encryption or MAC computation, but not both). Further details are provided in section 4.1.5.
Rec 11	Key usage controls (e.g., making use of control vectors, key wrapping) which bind the key-to-key control information in a secure way should be employed. Further details are provided in section 4.1.5.
Rec 12	Keys should be generated inside a TRSM and private keys should never exist in clear text outside a TRSM. Further details are provided in section 4.2.1.
Rec 13	Where appropriate, public keys are to be distributed such that their integrity and the binding with the owner are preserved (e.g. by using certificates). Further details are provided in section 4.2.4.
Rec 14	Usage of X 509, version 3, format certificates is recommended. Further details are provided in section 4.2.4.
Rec 15	When verifying a digital signature, users should check that the certificate status (Valid, Expired or Revoked) at the time the signature was generated, does not render the signature invalid. When a certification path is involved, this verification should be performed for every certificate in the path up to the root certificate. Efficient verification may require that the date and time when the signature was produced is unambiguously defined. Further details are provided in section 4.2.7.
Rec 16	Whenever possible, trusted time sources should be used, in order to allow reliable verification of certificate validity. Further details are provided in section 4.2.7.
Rec 17	An asymmetric key pair should be dedicated to one usage, for instance: one of entity authentication, non-repudiation of data, symmetric keys encryption. Further details are provided in section 4.2.8.
Rec 18	Where possible, payment service providers should avoid the use of key escrow, but key recovery is part of their due diligence and business continuity obligations. Further details are provided in section 4.3.

Table 1: Recommendations

2 Algorithm Taxonomy

The choice of an algorithm may be done according to functional, technical, legal, or commercial concerns. Thus, it may be useful to propose an algorithm taxonomy based upon different criteria. In this section it is proposed to sort algorithms according to their:

- Technical characteristics
- Typical usage
- Legal or commercial status

2.1 Technical Characteristics

- Unkeyed (hash functions)
- Symmetric (stream and block ciphers)
- Asymmetric (also known as public key algorithms).

Figure 1 proposes a taxonomy of cryptographic primitives and mechanisms based on their technical characteristics. Commonly used algorithms sorted according to this taxonomy are given as examples.

It should be noted that agreement of which algorithms to use is a necessary, but not sufficient condition for interoperability. A scheme implementing a cryptographic service must also ensure consistent understanding of encoding, padding, compression and filtering of data. These methods are not the main subject of this document but discussed briefly where appropriate.

2.1.1 Primitives

The following table shows the way different cryptographic primitives are used for cryptographic mechanisms:

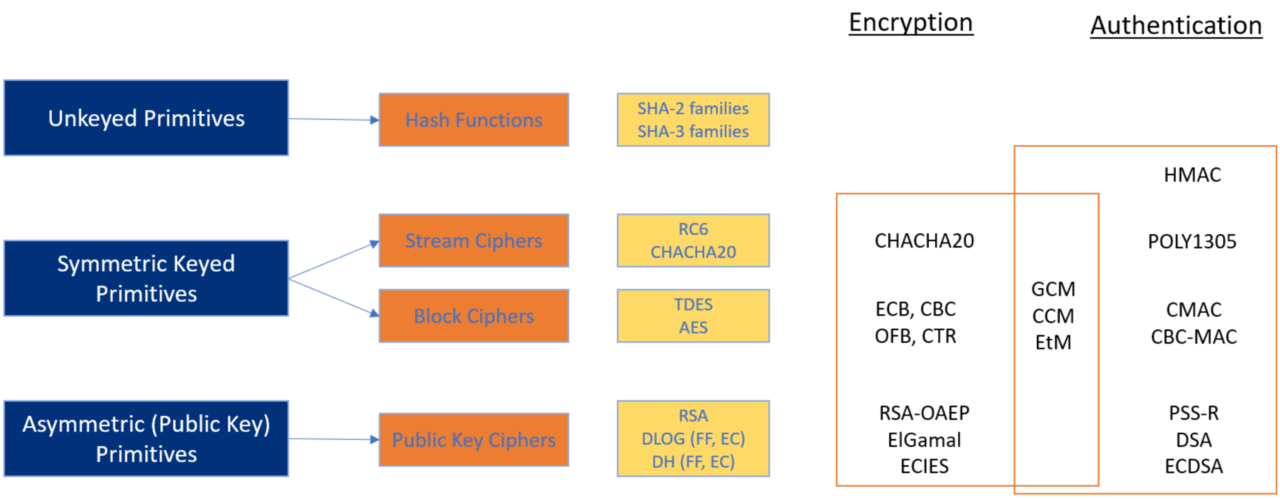

Figure 1: A technical taxonomy of cryptographic primitives and mechanisms

Note that algorithms shown in this figure are examples, not recommendations. The figure does not yet mention new post-quantum public key algorithms.

2.1.1.1 Un-keyed (Hash Functions)

The most important un-keyed cryptographic primitives are hash functions. Cryptographic hash functions take as input a message of arbitrary length and produce a fixed length message digest, providing three properties listed below:

1. ***Collision resistance*** meaning it is infeasibly hard to find two messages that map to the same message digest under the hash function.
2. ***Second pre-image resistance*** meaning it is infeasibly hard to find a second message that has the same message digest as a given first message.
3. ***One-wayness*** (also known *as pre-image resistance*) meaning it is infeasibly hard to find a message that maps to a given message digest.

The three properties are related, with collision resistance implying second pre-image resistance, which in turn implies one-wayness. If the collision resistance of a function is broken, the second pre-image resistance and one-wayness properties may still hold.

Most hash functions are specifically designed for this purpose, such as the SHA family including SHA-224 [97].There are also hash functions based on a modified block cipher, like Whirlpool [34] or on generic block ciphers [33], or functions based on modular arithmetic [35]. A hash function can also be derived from block ciphers where the key required by the algorithm is no longer secret and may be derived from the message to be processed: see ISO/IEC 10118-2 [31].

A hash function is normally employed as a component of a data integrity mechanism[1]. It is generally a central component of a digital signature scheme.

Examples: SHA-2, Whirlpool

2.1.1.2 Symmetric key

The most relevant symmetric key primitives are block ciphers and stream ciphers. Block ciphers provide a key-specific reversible mapping of a given message block to a cipher block. Stream ciphers produce a pseudo-random stream from a given key, which can be used to encipher a message.

Block ciphers are more versatile than stream ciphers, in that they facilitate constructions both for encryption and for message authentication codes (MACs). On the other hand, although usually still quite efficient and designed for hardware implementation, block ciphers are generally less efficient than pure stream ciphers. Numerous high-quality international standards for block ciphers are available.

Stream ciphers are generally very efficient and suitable for hardware implementation and are thus often used for confidentiality protection in high-speed telecommunication applications. Often they are proprietary algorithms and their specifications are confidential.

Examples: Block Ciphers (AES, TDES)

2.1.1.3 Asymmetric Key

Asymmetric or public key primitives have two different keys, a public and a private key, where data may be transformed under the public key, but the transformation can only be inverted using the private key. This enables non-repudiation functions, certain one-to-many and many-to-one

[1] E.g., integrity protection of the hash of a message can provide integrity protection of the message itself.

schemes and exchange of secret keys without a pre-existing secret channel. Public key primitives are usually much less efficient than symmetric key primitives. For that reason, they are generally only applied to small amounts of data, and where the asymmetric property is advantageous. Typical applications include key exchange, key encapsulation, signatures and hybrid encryption (see for example 2.1.2.2, and 2.1.2.4).

Examples: RSA, Diffie-Hellman

2.1.2 Elementary Constructions

This section provides a review of the most common constructions in which the primitives discussed above are used in actual applications.

2.1.2.1 Symmetric Encryption

Symmetric encryption is used to ensure confidentiality. Communication partners must hold the same (symmetric) key to encrypt or decrypt a message. Symmetric encryption can be constructed from block ciphers or from stream ciphers. Block ciphers are more flexible than stream ciphers and generally quite efficient, but not as efficient as stream ciphers.

Stream ciphers generate a reproducible, pseudo-random key stream from the key, that is generally used to encipher a message using bitwise modular addition. Note that stream cipher encryption is always malleable, allowing the manipulation of the transmitted data, even if confidentiality is assured.

Block ciphers can be used in one of several modes of operations to encrypt a message.

Details of the modes can be found in section 3.2.1.1.

2.1.2.2 Asymmetric Encryption

Asymmetric (or public key) encryption is used to ensure confidentiality. In the case of message transmission a sender uses the recipient's public key to encrypt the message, while the recipient uses the corresponding private key for decryption. The recipient's public key may have been obtained from a trusted key directory, for example. This allows for effective key management, when many senders are to encrypt data for a single recipient (e.g. e-mail encryption).

A popular way of establishing trust in keys is the use of public key certificates. Public key certificates may be published in an online key directory but with a trusted authority's signature (see 2.1.2.4 below) certifying the owner of the key. Prospective senders then only need to hold the authority's key to be able to establish secure communications with the intended recipient.

Asymmetric key encryption is generally not very efficient, and therefore mostly used for small data items such as symmetric keys i.e., for hybrid encryption as described below.

Examples: RSA-OAEP (PKCS#1 [112], ANSI X9.31 [71], ISO/IEC 18033-2 [54]), EL-GAMAL, ECIES (IEEE P1363 [142] ISO/IEC 18033-2 [54]), Paillier ([180] ISO/IEC 18033-6 [59]).

Typically, a public key is used for encrypting a temporary symmetric key (session key) which is then used for encrypting the message. The recipient uses its private key to decrypt this temporary key and then uses this to decrypt the message.

See section 3.2.3 for more information.

Examples: S/MIME, SSL, TLS

2.1.2.3 MAC

Message authentication codes (MACs) are used to guarantee message authenticity and integrity. Communication partners must hold the same (symmetric) key. Then upon transmitting a message over a public channel the sender, using the key, computes a MAC, which is attached to the message, and which can be verified by the recipient using his key, ensuring authenticity and integrity of the received message.

MACs are generally constructed from hash functions or block ciphers using cipher-block chaining.

Examples: HMAC (ISO/IEC 9797-2, mechanism 2 [24]), CMAC (ISO/IEC 9797-1, mechanism 5 [23]) and CBC-MAC (ISO/IEC 9797-1, mechanism 1 [23])

2.1.2.4 Signatures

Signatures employ the properties of asymmetric primitives to allow for non-repudiable authentication. To sign a message, it is usually hashed with a cryptographic hash function to obtain a short message digest. The digest is then transformed with the signer's private key to obtain a signature.

Any holder of the signer's public key can check if a signature authenticates a message under the corresponding private key, but public key holders are unable to generate signatures themselves. As such the signature uniquely authenticates the message as originating from the signer, enabling non-repudiation services.

As with asymmetric encryption, public keys can be published in a public directory.

Examples: RSA-PSS (PKCS#1 [112], ISO/IEC 9796-2 [21]), DSA (FIPS PUB 186-4 [83], ISO/IEC 14888-3 [44]), ECDSA and similar schemes (ISO/IEC 14888-3 [44])

2.2 Typical Usage

Elementary constructions may be classified according to the security services or functions they are suited for:

- Confidentiality protection
- Data confidentiality,
- Key encapsulation,
- Key establishment.
- Integrity protection
- Data origin authentication,
- Entity authentication,
- Non-repudiation.

The following table shows which kind of elementary construction is considered suitable for a given usage:

Construction Usage	Symmetric Encryption	Asymmetric Encryption	MAC	Signature
Data confidentiality (2)	Yes (one to one)	No (1)	No	No
Key encapsulation (2)	Yes (one to one)	Yes (many to one)	No	No
Key establishment	No	Yes	No	No
Data origin authentication	No	No	Yes (one to one)	Yes (one to many)
Entity authentication	No	No	Yes (one to one)	Yes (one to many)
Non-repudiation	No	No	No	Yes

Table 2: Matching of techniques and security functionalities

Notes:

(1) Can be used, but the computational complexity is generally excessive for the given purpose.

(2) Hybrid encryption as defined in section 4.1.2.3 is for data confidentiality and key encapsulation.

2.2.1 Confidentiality Protection

Confidentiality means that information is not made available or disclosed to unauthorised individuals, entities or processes. This is usually ascertained by means of encryption. An encryption scheme achieves confidentiality if an attacker cannot distinguish between any two given data items of his choosing once the data has been encrypted.

2.2.1.1 Data Confidentiality

Data or message confidentiality pertains to stored information or to data being exchanged in a communication. A symmetric cipher may be used to protect the confidentiality of data in either scenario. Stream ciphers typically execute at a higher speed than block ciphers and have lower hardware complexity, but block ciphers are more flexible. An asymmetric algorithm can also be used to encrypt data but is for performance reasons only recommendable for small volumes of data. If many to one confidentiality is required, for instance for e-mail encryption, hybrid encryption is the method of choice.

2.2.1.2 Key Encapsulation

Key encapsulation mechanisms (KEMs) are a class of encryption techniques designed to secure symmetric cryptographic key material for transmission. Key encapsulation is sometimes based on symmetric key schemes, but more generally uses asymmetric (public-key) algorithms. Public keys, which are only suitable for encryption of small data volumes, are used to encipher a symmetric key to be shared between two parties.

2.2.1.3 Key Establishment

Key establishment is the process of exchanging a cryptographic key between two parties, using cryptographic techniques. Asymmetric (public key) encryption techniques lend themselves to key establishment, because they allow the secure construction of shared secret keys from freely accessible public key certificates.

2.2.2 Integrity Protection

Integrity protection refers to messages arriving as they have been sent, or data being retrieved as it has been stored. Generally, one speaks about authentication when the assurance about message origin is only towards the communication partner. If there is an added requirement that this assurance be transferable to third parties as well, one speaks of non-repudiation of origin.

2.2.2.1 Data origin / Message / Entity Authentication

Authentication means assurance that a message or data item has been delivered as sent by a specified communication partner. Entity authentication is the corroboration that an entity is the one claimed. Authentication can be achieved by means of MACs or digital signatures, however MACs cannot guarantee protection against repudiation. MACs offer better performance, but signatures allow for authentication towards multiple recipients (one-to-many), as the public verification key can be freely distributed.

Usage restrictions (key blocks) can enforce that the recipient is able only to verify MACs and not generate them (see 4.2.8).

2.2.2.2 Non-Repudiation

Non-repudiation means transferable (e.g. to a judge) assurance about the originator of a data item. Digital signatures are used to provide non-repudiation. Only the originator of the data holds the private key and can generate signatures. The public key can be freely distributed to receivers and third parties, allowing verification of data origin. Once a public key has been accepted as belonging to an originator, the originator cannot later repudiate signed data items, as he is the only party capable of generating such signatures. Symmetric techniques cannot provide non-repudiation, because sender and receivers hold the same key, making any receiver another potential originator of the message.

2.3 Standardisation

Cryptographic algorithms may be standardised by international standards bodies such as ISO (primarily by ISO/IEC JTC1/SC27) or by national bodies such as ANSI or NIST. They may also be standardised by industry/technology-specific standardisation bodies such as the IEEE and the IETF or companies/consortia such as PKCS for RSA standards and SECG for elliptic curve standards.

An algorithm that occurs or is referenced in a standard is usually published and open for scrutiny by independent researchers. Being able to withstand research and theoretical attacks for many years builds trust in such an algorithm.

Standards organisations, industry consortia or government agencies may also publish recommendations concerning key lengths or implementation guidelines for cryptographic algorithms.

Algorithms for military use are typically kept secret. Government algorithms may be released with the expectation that they will be used for government and civil business. An example of this is the algorithms and protocols made freely available by NIST. These are sometimes the results of public competitions (e.g. AES and SHA-3) and will frequently then be adopted as ISO standards.

Proprietary algorithms are usually designed for a specific industry but are often less well reviewed by the open cryptographic community, especially if they are kept confidential and so are not recommended for financial applications.

EPC recommendation 1

Only algorithms whose specifications have been publicly scrutinised (ideally with a public design phase), and whose strength has been assessed by crypto experts can be recommended. Algorithms specified in International Standards should be preferred. This recommendation also applies to algorithms for key generation.

3 Algorithm Related Design Aspects

Cryptographic functions such as unkeyed hash functions, symmetric algorithms and asymmetric algorithms can be combined in many ways to achieve a variety of objectives. Often the cryptographic 'primitives' come as a standard toolbox, which can be used flexibly by many applications. Care must be taken when developing the compound functions from the primitive functions so as to ensure interoperability, security and efficiency.

This section describes the main primitives and ways in which they are combined.

3.1 Primitives

3.1.1 Unkeyed

3.1.1.1 Hashes

Primitive hash functions are unkeyed and designed to create a short 'digest' of a long string of input data. These functions are designed to be one-way (i.e. difficult to find a pre-image of a given hash) and collision resistant (i.e. difficult to find two inputs that hash to the same digest). Most signature functions depend on the use of hash functions as do some asymmetric encryption functions (see section 0).

SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 are algorithms defined by NIST (FIPS 180-4 [82]) and that produce 160, 224, 256, 384 and 512-bit hash results, respectively. The longer result algorithms tend to be known collectively as the SHA-2 family. Together with Whirlpool (which produces a 512-bit hash result) and RIPEMD-160 [151] (which produces a 160-bit hash result smaller than the recommended minimum of 256 bits) they are now also standardised in ISO/IEC 10118-3 [34]. SHA-224 is based on SHA-256. Computation of a SHA-224 hash value involves two steps. First, the SHA-256 hash value is computed, except that a different initial value is used. Second, the resulting 256-bit hash value is truncated to 224 bits. In August 2015, NIST published FIPS 202 (and revised FIPS 180) thereby standardising SHA-3 that can produce 224, 256, 384 and 512-bit hash results (see [82], [86], [175]).

MD5 is an old algorithm producing only a 128-bit hash result (see section 5.1.3.4). MD5 has been so extensively attacked that it shall no longer be used.

There are also collision search attacks on SHA-1, first in 2005 (see [156]) and more recently by Marc Stevens, who has also developed a method [172] for discovering an attack attempt, given a message. If it is necessary for some reason (e.g., backward compatibility) to still support SHA-1, it is recommended to apply such a method. In 2015 a “freestart collision” for SHA-1 was realised (see [178]) which computed the first practical break of the full SHA-1 algorithm reaching all 80 out of 80 steps. In February 2017 researchers in Amsterdam (see [185]) announced a collision attack against the full SHA-1 algorithm without the benefit of a “freestart”, allowing an attacker to create two different files that have the same hash value. The 2017 attack confirmed in practice what was already known in theory, that the algorithm is fundamentally broken – its collision resistance is broken from 80 bits down to less than 64 bits and there is speculation that its pre-image resistance may similarly be reduced from 160 bits down to less than 128 bits.

In consequence, for message signing, MD5 should no longer be used and legacy message signing systems that rely on the collision-resistance of SHA-1 should be migrated to a member of the SHA-2 family or to SHA-3. In addition, [132] disallows SHA-1 for digital signature generation.

More recent research by Gaetan Laurence and Thomas Peyrin (see https://sha-mbles.github.io/ and [193]) improves the previous SHA-1 collision attacks by using new techniques to turn collision attacks into so-called 'chosen-prefix collision attacks' where for any given prefixes P and P' the attacker might find two messages M and M' that begin with these prefixes and have the same hash. The authors showed that such collisions can be computed with a complexity of $2^{63.4}$ SHA-1 calculations whereas ad-hoc SHA-1 collisions require $2^{61.2}$.

3.1.1.2 Length recommendations

In the long term, hash functions with less than 256 bits output length should not be used. It is recommended to use SHA-2 or SHA-3 (see Table 4). Where collision-resistance matters, legacy systems using SHA-1 should replace SHA-1 according to the above recommendations.

3.1.2 Symmetric Key

3.1.2.1 Triple DES (TDES)

The key length of TDES is 112 bits (for two-key TDES, 2TDES) or 168 bits (for three-key TDES, 3TDES). TDES consists of three invocations of the DES primitive. The key is split into two or three single DES keys. First the data is encrypted with the first key, then it is decrypted with the second key, and finally encrypted again with the first key (in case of 2TDES) or the third key (in case of 3TDES).

An ISO Technical Report (ISO TR 19038 [5]) on TDES modes of operation was published in 2005. This technical report specifies modes of operation consistent with ISO/IEC 10116 [19] and provides implementation guidelines specifically for TDES and the financial industry.

Further security considerations may be found in section 0.

3.1.2.2 Advanced Encryption Standard

The Advanced Encryption Standard (AES) has been developed by NIST as a replacement for DES and has been approved, as Federal Information Processing Standard (FIPS 197 [84]), in 2001.

AES is a symmetric block cipher that supports block lengths of 128 bits and key lengths of 128, 192 and 256 bits. It has been designed to be considerably more efficient than TDES. It is now standardised in ISO/IEC 18033-3 [56].

In [124] NIST has defined modes of operation for AES consistent with ISO/IEC 10116 [19].

Further information on AES may be found at: http://csrc.nist.gov/groups/ST/toolkit/.

3.1.2.3 Efficiency

In software implementations, the computation of an AES encryption takes about as much as one or two DES encryptions. This implies that encryption of a file or a data stream by AES will be much faster than TDES, since TDES takes six encryptions per 128 bits.

In the special case of PIN block encryption, both DES and AES encryptions are just one block of data, but AES encryption will still be faster.

However, for PIN block encryption, the block size is an issue. Where the standard PIN block length (defined in ISO 9564 [2]) used to be 64 bits, which is smaller than the AES block length, the standard has now been updated to include a new PIN block format of 128 bits. This new PIN block format is called Format 4. New systems should be designed to support PIN blocks.

3.1.2.4 Algorithm and Key length recommendations

EPC recommendation 2

- AES is the recommended standard for new systems.
- 3TDES is still secure in use cases where there is no concern regarding reduced block sizes.
- 2TDES may still be sufficiently secure for existing systems under specific conditions (see 0); plans should be made to migrate to AES.
- Single DES (56 bits) should be considered broken.

Note that 2TDES may have an effective key length of less than 112 bits if the volume of data enciphered under a key is large. See section 0 for more details on security levels of TDES. In general, due to potential internal collisions when processing large amounts of data (> $2^{b/2}$ blocks where b is the block size) under the same key, block ciphers with a larger block size (e.g. AES with 128-bit block size) can provide better security than those with a smaller block size (e.g. TDES with 64-bit block size). For examples see [183] and [182].

3.1.3 Asymmetric key

The primary asymmetric key primitives considered in this document are primitives whose security is based on the difficulty of integer factorisation, such as RSA and primitives whose security is based on the difficulty of solving discrete logarithms in a finite group (and related problems), such as DSA and Diffie-Hellman (DH). The latter category can be separated into those where the group is in a finite field (such as DSA and DH) and those where the group elements are points on an elliptic curve[2] (such as the elliptic curve versions of DSA and DH).

3.1.3.1 RSA specific issues

An RSA key pair comprises

- A public key modulus N (that is the product of two large secret prime numbers p and q) and a public exponent e.
- A private exponent d that is determined given knowledge of p and q.

The RSA public key operation (the basis for encryption and signature verification) applied to data X is X^e mod N.

The RSA private key operation (the basis for decryption and signature generation) applied to data Y is Y^d mod N.

Computational Enhancements

RSA computation can be made significantly quicker for the owner of the private key if the implementation takes advantage of the Chinese Remainder Theorem (CRT). Using the CRT the signer (or decryptor) performs calculations modulo each prime factor of the RSA modulus N, instead of performing the calculations modulo N. Because the factors of N are half the length of N, the CRT method is much faster. The CRT method requires that the prime factors of the modulus are kept and used for the computation involving the private key. These prime factors must be stored with the same level of security as the private key.

[2] In this case the attacker cannot utilise the faster algorithms which exploit the field structure of the group and so cryptographic keys can be shorter.

Using CRT when signing (or decrypting) is purely a local decision and has no impact on the signature verification process (or encryption process). CRT can make signing and decrypting 3 to 4 times faster.

Some attacks on RSA implementations using CRT have been published. These attacks involve the accidental or controlled introduction of errors into the signature or decryption computation. They may be prevented by a sound design (e.g. verify generated signatures before releasing them).

RSA computation can be made significantly quicker for the user of the public key if the public exponent is small compared to the modulus. This approach is sometimes referred to as low-exponent RSA. Typical values for the exponent are 3 (low exponent) and 65537 (2^{16}+1). The use of a small public exponent can deliver performance benefits for resource-constrained RSA encryption and RSA signature verification operations. However, the use of low-exponent RSA may expose the system to certain attacks when used to perform encryption operations without random padding. The use of secret random padding becomes even more important when using e=3 (see for example [144], [150]). For example, the use of exponent 3 immediately reveals the plaintext if used for message encryption to at least three recipients (see [144]).

3.1.3.2 Elliptic Curve Cryptosystem (ECC) specific issues

Domain Definition

ECC keys and certificates are shorter in terms of storage and communications than keys and certificates of some other schemes. Whereas some other schemes, such as RSA, use ordinary integers for the underlying number system, ECC systems use points that are solutions for a particular elliptic curve, with the result that ECC systems require users to make a choice of parameters to define the *domain*:

- The *curve* itself – defined by two parameters *a* and *b*.
- The *base field* – defined by one parameter *q* that can be one of two forms; a prime number or a power of two (binary).
- A point on the curve defined as the *generator* – parameter *G*, with its *order* – parameter *n*.

For any given *domain* a large number of key pairs can be created resulting in public keys *Q* and private keys *d*.

In a large multi-user system, the domain parameters could be individually selected by each user or could be shared by mutual agreement. Creating a domain with its own parameters is complex and therefore only done for research purposes, custom cryptographic applications, or to meet very specific security requirements.

Computational Enhancements

Computational efficiency is strongly influenced by parameter choice and curves defined over binary fields, including Koblitz curves, are sometimes selected since scalar multiplication (the dominant computational step) is efficient. Binary fields can be represented in two forms; polynomial basis or normal basis – the former being most efficient for software implementations and the latter for hardware. However, care should be taken with implementations using binary fields as they are more patented and may be more vulnerable to mathematical attacks. For this reason, standardised curves over prime fields are recommended.

In addition to standardised curves such as NIST P-256, there are now many new innovative techniques proposed for ECC cryptography such as FourQ [184] and curve25519 specified in IETF RFC 7748 [102] (the authors of which cite advantages in performance and security compared to NIST curves).

3.1.3.3 Comparison of RSA with ECC

With such speed-up efficiencies the signing performance for ECC can be several times faster than an RSA implementation, whereas the verification performance can be about twice as slow as RSA (using e=3).

Storage of an ECC private key takes a little over twice that of an RSA key. Certificate sizes are comparable, with RSA, assuming pre-agreed domain parameters, ECC private keys and certificates are shorter than RSA private keys and certificates.

For key generation, then for a known domain, ECC is much faster.

From a security perspective, ECC implementations are thought to be less susceptible to timing and power analysis attacks than equivalent RSA implementations, although not immune. Fault analysis attacks which leverage the Chinese Remainder technique used for RSA speed-up have no equivalent in ECC implementations, however the random numbers used by ECC implementations are potentially a target for side-channel attacks.

The state of the art in cryptanalysis is not so clear, however, the best-known algorithm to attack ECC systems (Pollard's rho[3]) is less efficient than the General Number Field Sieve (GNFS) algorithm used to factor RSA moduli - hence one reason why ECC keys can be shorter. A further advantage is that Pollard's rho has no constant in the run-time formula (unlike NFS) and thus the security level for ECC can be more accurately predicted and the algorithm cannot be "improved" by developments that lower the value of the constant (although increased parallelisation might). Koblitz curves are slightly more susceptible and keys should thus be a few bits longer than for generally selected curves.

In the table below, the results of the performance comparison between the different algorithms for usage with digital signatures are listed: '-' stands for a negative point, while '+' stands for positive point. For this table, RSA with 1024-bit keys is considered, and the security of the three algorithms is assumed to be the same.

	ECDSA	RSA	DSA
Complexity System Set-up	- (< hours)	+ (<Minutes)	+ (< Minutes)
Users computational capacity	+ (simpler chip)	- (more complex chip)	- (more complex chip)
Users storage capacity	+ (160 bits per key)	- (1024 bits per key)	- (1024 bits per key)
Creation time digital signature	+	- (6 times ECDSA)	- (4 times ECDSA)
Verification time digital signature	- (4 to 7 times RSA)	+	- (28 times RSA)
Size digital signature	+ (320 bits per sig.)	- (1024 bits per sig.)	+ (320 bits per sig.)

Table 3: Comparison of signature schemes

[3] Newer methods introduced by Antoine Joux against elliptic curves over binary fields are also important to note [181].

One can conclude that ECC signatures schemes such as ECDSA may have advantages over RSA and DSA in applications where the available computational or data storage capacity in the signer's hardware is limited, e.g. in smartcard applications. However, unless the system parameters (Base Field, Curve and Generator) are implicitly known by the verifier, then these also need to be stored and made available. Sharing and distribution amongst users depends on trust and risk issues and if none of the parameters were implicitly known it would add approximately an additional 1000 bits to the storage requirements for ECDSA.

In a similar vein, for maximum performance pre-computation can be used with a trade-off between the partial results to be stored and the amount of speed-up. For a twofold improvement approximately 1,300 bytes of additional storage are required.

3.1.3.4 Algorithm and Key length recommendations

It is difficult to state precise requirements on key lengths without precise details of usage and implementation. However, it is recommended:

EPC recommendation 3

- No longer use RSA keys of 768 bits and ECC keys of 130 bits, or less.
- Avoid using 1024-bit RSA keys and 160-bit ECC keys for new applications unless for short term low value protection (e.g. ephemeral authentication for single devices).
- Use at least 2048-bit RSA or 224-bit ECC for medium term (e.g. 10 year) protection (see [174]).
- For considerations regarding low exponent RSA, section 3.1.3 should be consulted.

Further details are provided in section 0.

3.1.4 Security levels

Cryptographic algorithms provide different “strengths” of security, depending on the algorithm and the key size used. The concept of algorithm "strength" expressed in "bits of security" is described in the NIST guideline on key management SP800-57 [129]. ECRYPT has published a yearly report on algorithms and key sizes [162] which addresses the same issue (the previous report was published by ENISA [174]). In contrast to the conclusions of these reports, the largest RSA key size that has been broken, is 829 bits.

Two algorithms are considered to be of equivalent strength for the given key sizes (*X* and *Y*) if the amount of work needed to “break the algorithms” or determine the keys (with the given key sizes) is approximately the same using a given resource. The strength of an algorithm (sometimes called the work factor) for a given key size is traditionally described in terms of the amount of work it takes to try all keys for a symmetric algorithm with a key size of "*X*" that has no short cut attacks (i.e., the most efficient attack is to try all possible keys). In this case, the best attack is said to be the exhaustion attack. An algorithm that has a "*Y*" bit key, but whose strength is equivalent to an "*X*" bit key of such a symmetric algorithm is said to provide “*X* bits of security” or to provide "*X* bits of strength". An algorithm that provides *X* bits of strength would, on average, take $2^{X-1}T$ to attack, where *T* is the amount of time that is required to perform one encryption of a plaintext value and comparison of the result against the corresponding ciphertext value.

Bits of security	Symmetric key algs.	Hash functions	RSA (key length in bits)	ECC (key length in bits)
80	2TDES (1)	SHA1	1024	160
112	3TDES	SHA-224 SHA3-224	2048	224
128	AES-128	SHA-256 SHA3-256	3072	256
192	AES-192	SHA-384 SHA3-384	7680	384
256	AES-256	SHA-512 SHA3-512	15360	512

Table 4: Comparable security strengths (adapted from section 5.6 of [129])

Notes:

1. For 2TDES the key strength depends on the number of plaintext and ciphertext pairs the attacker has available. In those cases where the number is limited, as is often the case for card payment systems, a 2TDES key can provide as much as 112 bits of security. The depicted strength of 80 bits shown in the first row of the table occurs only when the attacker has the possibility and the incentive to collect of the order of 2^{40} plaintext and ciphertext pairs. More generally when an attacker can collect of the order of 2^{t} plaintext and ciphertext pairs then they can find one of the keys used with effort 2^{120-t}. The most recent result related to this attack can be found in [179] in which the author shows that the plaintext and ciphertext pairs need not all be encrypted under the same key[4], and then the attacker will expect to determine the key used for at least one of the pairs and, moreover, will be able to do so even if only partial plaintexts are known.
2. There also exist theoretical attacks on 3TDES and AES that reduce the key strength but require special “unrealistic preconditions” to be feasible. For example:
    - [147] shows that with 2^{28} fixed known plaintexts encrypted using 3TDES under different keys then one of the keys can be recovered with effort 2^{84}.
    - [160] shows that with 2^{43} fixed known plaintexts encrypted using AES128 under different keys, one of the keys can be recovered with effort 2^{85}.

Section 3.4 of the NIST publication SP800-67 [130] makes the following observation on 2TDES and 3TDES[5]:

“The security of TDEA is affected by the number of blocks processed with one key bundle. One key bundle shall not be used to apply cryptographic protection (e.g., encrypt) to more than 2^{20} 64-bit data blocks.

Note that this limitation applies to a key bundle with three unique keys (i.e., 3TDEA); the use of TDEA with only two unique keys (i.e., 2TDEA) shall not be used to apply cryptographic protection (see section 3.1).

[4] It had previously been thought that the pairs had to have been created using the same key.

[5] Note that NIST Recommendation SP800-67 caters for the protection of any type of information, rather than specific payment transactions; it applies to “*all federal agencies, contractors of federal agencies, or other organizations that process information (using a computer or telecommunications system) on behalf of the Federal Government to accomplish a federal function*”.

Also, note that in the previous version of SP 800-67 (dated January 2012), 3TDEA was limited to processing 2^{32} 64-bit blocks, and 2TDEA was limited to 2^{20} blocks. These prior limitations should be considered when processing information protected using the 2012 or the original version of SP 800-67 (e.g., determining the risk of accepting the decrypted information when the limit provided in this revision for 3TDEA is exceeded or the information was encrypted using 2TDEA)."

Moreover in March 2019 NIST published a 2nd revision of NIST SP 800-131 [132] in which they disallow encryption of data using 3TDES after 2023. Similar recommendations have meanwhile been published by the German BSI [196] .

Consistent with the above, the EPC recommendation is to use AES rather than TDES for all new payment systems. EPC further recommends to only continue to use TDES in existing payment systems for which doing so is assessed as sufficiently secure.

When collision resistance is not required the security level of hash functions is doubled. The collision resistance strength of SHA-1 is about 63 bits.

- For DSA and DH the key sizes are comparable to RSA, with subgroup parameter comparable to ECC. For the purposes of these recommendations, RSA key lengths that, due to implementation constraints, are slightly less than 2048 bits (e.g. 1976 bits) have equivalent security to 2048-bit RSA.
- Some additional information on key strengths for AES and RSA and ECC is provided in [162], [167] and [168].

3.1.5 Quantum computing considerations

For protecting long-term secrets, including secret keys, it is useful to consider current developments in physics: there is a possibility that so called "quantum computers" could be built in the next decades on a scale that is relevant to breaking cryptography. Research is refocusing on error reduction with IBM's recent update of their quantum development roadmap delaying error correction delivery from 2026 to 2029, earliest[6].

Such devices could solve certain cryptographic problems with lower complexity than current computers and so if these computers would become reality then new cryptographic algorithms would be necessary: these are called "quantum resistant" algorithms or alternatively "post-quantum crypto" algorithms.

In order to better assess the probability and impact of the availability of quantum computers, some background facts and observations are summarized hereafter.

3.1.5.1 Impact on cryptographic primitives

The crypto-breaking algorithms that would run on a quantum computer are very different from the algorithms that attack today's crypto such as the General Number Field Sieve (GNFS). The two quantum computer algorithms relevant for cryptography are:

- *Shor's algorithm* reduces the complexity of solving both RSA and elliptic curve-based problems enormously. The key length increase needed to compensate for this reduction in complexity would result in key sizes that are not practical.

[6] See https://www.ibm.com/quantum/technology

- *Grover's algorithm* seemingly enables a search of the key space of secret key-based algorithms in a time proportional to the square root of the key space. Grover's algorithm can also be applied to hash functions.

However further analysis of Grover's algorithm reveals that pre-quantum security can be maintained without doubling key lengths:

1 Grover's algorithm does not benefit from linear speed-up through parallelisation. To obtain a 1,000-fold speed-up when using Grover's algorithm, a million quantum computers would be needed[7].

2 In combination with this, the quantum attack work function on AES-128 is estimated to be 2^{101} where 2^{64} operations are performed on a single quantum machine[8] and this indicates AES-128 remains fit for purpose in most contexts. See also [200].

In their FAQs https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/faqs (old Q8) NIST still states that when taking all considerations into account, "*it becomes quite likely that variants of Grover's algorithm will provide no advantage to an adversary wishing to perform a cryptanalytic attack that can be completed in a matter of years, or even decades*".

Assuming (very) optimistically that attackers would have access to quantum computers that have the required performance and error correction (see 3.1.5.2), the following changes would have to be made to current cryptographic primitives to stay at a comparable security level:

- Current public key cryptosystems will have to be replaced by a new class of "quantum resistant" public key cryptosystems that is currently under development. As of this writing, standardization of quantum resistant public key cryptosystems by ISO and NIST has notably progressed and some vendors and regulators have started experimenting with their use (for example, Google implemented a quantum secure key exchange algorithm for TLS 1.3 while a BDF-MAS initiative tested quantum secure e-mail exchange). NIST has also standardised stateful hash schemes that are quantum resistant [136].
- Secret key-based cryptosystems do not have to be replaced. Doubling of pre-quantum key lengths from 128-bits to 256-bits would maintain the pre-quantum security level against a quantum capable adversary with an 'idealised' Grover's algorithm; however considering the actual algorithm's details then such doubling appears unnecessary.
- Hash function-based systems that require one-wayness will have to use hash functions such as SHA-256 that are (pre-quantum) collision resistant[9], see [200].

On August 2024 NIST standardized first PQC algorithms derived from CRYSTALS-KYBER (key-establishment) and CRYSTALS-Dilithium and SPHINCS+ (digital signatures). An additional signature standard based on FALCON is in preparation (c.f. https://csrc.nist.gov/news/2024/postquantum-cryptography-fips-approved).

Moreover, the Post-Quantum Cryptography (PQC) standardization process is continuing into a fourth round with key establishment algorithms still under consideration. For certain applications, NIST may also be interested in signature schemes that have short signatures and fast verification. ISO/IEC 18033-2 is being amended to include PQC algorithms.

---

[7] Grover's quantum searching algorithm is optimal, Christof Zalka, https://arxiv.org/pdf/quant-ph/9711070.pdf

[8] Quantum Resource Estimation, Vlad Gheorghiu

[9] This is because Grover's algorithm implies that there is not much difference between second preimage resistance and one-wayness (as defined in 2.1.1.1) anymore.

The UK National Cyber Security Centre (NCSC) published in November 2023 guidance to help organisations and critical national infrastructure providers think about how to best prepare for the migration to post-quantum cryptography [208].

To ensure a smooth transition to PQC primitives, crypto agility - as recommended by NIST [204] and BSI [205] - should be integrated into the cryptographic services' pipeline. TLS ciphersuites are already being upgraded to use a hybrid key establishment mode, i.e., a mode wherein public key encryption uses both classical and post-quantum resistant techniques so that an eavesdropper would need to break both the classical and the post-quantum encryption. This is largely motivated by the 'harvest now decrypt later' threat urging for PQC adoption without allowing for a typical scrutiny period as generally advisable for new cryptographic primitives. Finally, it is worth noting here that with the integration of the new PQC standards in other standards, practical challenges may still surface, as evidenced by recent discussions on PQC private and public key data formats. As NIST did not standardize these data formats, standardization bodies deviating from the key representations in the NIST documents recently caused interoperability issues for prior generated early-mover key material.

3.1.5.2 Current progress in quantum computers

The most important measure for the performance of a quantum computer is the number of "qubits" it has. A qubit is a quantum bit in a yet unknown state of either 0 or 1, which will be revealed at the end of the computation. Gate time is also an important measure (risk assessments should anticipate increases).

For cryptographic purposes, the number of qubits needs to be at least as high as the number of key bits, but sometimes even much higher. This observation on its own suggests that ECC-based cryptosystems might be impacted before classical public key cryptosystems like RSA (with thousands of key bits).

Moreover, implementations of qubits suffer from "decoherence" (i.e. errors) where the information is lost to the environment. This problem can be addressed by combining a number of "physical" qubits into "logical" qubits, dramatically increasing however the number of required qubits (e.g. increasing the number of qubits needed for breaking today's cryptosystems from thousands to millions). Research in this space indicates possibly upcoming improvements, be it through new chip architectures or advances in materials research.

Both aspects eventually explain why current time estimates for the availability of quantum computers able to break specific cryptographic algorithms vary widely. In December 2023 IBM announced the implementation of a record-size laboratory-environment 1121 qubit quantum processor and at the same time explained they will refocus research on their modular 100+ qubit processor designs with error correction for scalable commercialization. Such general quantum computers, irrespective of their modular or non-modular design, must not be confused with a completely different type of quantum computer that solves a very different type of problem and that is already available with thousands of qubits (e.g. D-Wave 2000Q). D-Wave computers use quantum annealing, but there is no evidence that this technology can be used to perform general quantum computations as would be needed for breaking crypto primitives.

In trying to extrapolate what a concerted research program (similar to the Apollo or Manhattan programs) could reach within the foreseeable future, a recent BSI study estimates that, under the assumption that current technical challenges are met, a quantum computer that breaks 2048-bit RSA in a few hundred days appears possible. However a faster attack (in one day) would require new technological solutions to connect up to 1000 of such computers and would need roughly the full annual industrial demand of Helium 3, i.e., investments by far larger than current efforts in quantum computing. The same study noticeably also remarks that progress in materials research towards lower errors would bring these numbers down significantly, why one can be much less confident about the reliable mainstream thinking that steady progress towards cryptanalytic relevance will take at least one decade or more likely two (c.f. "Status of quantum computer development" [200]).

Irrespective of such analyses, many parties that invested in quantum computer research feel the need to publish about the research, even though not much progress is observed. This situation is not likely to change in the near future. For example, there are publications on factoring small numbers using a quantum computer that use Grover's algorithm rather than Shor's.

Note finally that quantum computers have nothing to do with "quantum cryptography" that uses physical principles to establish cryptographic keys, although the latter is sometimes proposed as an alternative for key establishment; these methods allow a different kind of protection that is outside the scope of this document. A good overview on Quantum Safe Cryptography, also covering quantum cryptography, can be found in a comprehensive publication by the BSI [206].

EPC recommendation 4

In view of the current published progress of quantum computing initiatives, and especially for example concerns regarding Harvest Now Decrypt Later, public key cryptography policy is significantly impacted; however, as noted by UK NCSC [208], policy related to symmetric cryptography and hash functions is not significantly impacted.

Where public key cryptographic primitives are used, crypto agility - as recommended by NIST and BSI [204], [205] - should be integrated into the cryptographic services' pipeline. Moreover, where practically feasible, hybrid key establishment and hybrid digital signature techniques, combining classical and post-quantum algorithms, are generally advisable as a transitional migration strategy.

3.2 Constructions

3.2.1 Symmetric Encryption

This report considers two types of symmetric encryption: block ciphers in a mode of operation and stream ciphers. Authenticated encryption is addressed in section 3.2.6.

3.2.1.1 Block Cipher Modes of Operation

A block cipher is a keyed function that encrypts an n-bit block of data, where n is a fixed characteristic of the cipher, or a chosen parameter. For the purposes of this report it is an example of a 'primitive' cryptographic function. Within the financial industry, TDES and AES are the most commonly used block ciphers. These ciphers have different block and key lengths as detailed below. Block ciphers are standardised in ISO/IEC 18033-3 [56].

Messages exceeding the block size are partitioned into blocks and are then processed using the block cipher[10] in one of its Modes of Operation (see ISO/IEC 10116 [19]). The Modes defined in [19] are:

[10] For messages shorter than the block size, one block is built using a suitable padding technique.

Electronic Code Book (ECB)

This is the simplest mode. Each block of data is encrypted independently using the block cipher. This method can be subject to cryptanalysis based on known ciphertext (or "dictionary attacks") and so is rarely used for messages longer than one block. The use of ECB is discouraged except for exceptional purposes, such as encrypting a randomized PIN block per ISO 9564.

Cipher Block Chaining (CBC)

This mode of operation is the most commonly used. It solves the problems with ECB mode by adding the last cipher block to the next plaintext block before encryption. Specifically each input block is first xor'ed with the encrypted text of the previous block (or an initialising vector for the first block in the sequence), and then the result of the xor'ing is enciphered with the block cipher.

The resulting cipher is secure under common definitions of security, but only if the initialisation vector added to the first block is chosen freshly random, independently from the entire message content.

Cipher Feedback (CFB) and Output Feedback (OFB) modes

In their simplest form, the input message is partitioned into a stream of r-bit blocks (r<=n) and each block of this input stream is encrypted to a cipher stream block by xor'ing it with a key stream block:

- in CFB each key stream block is created by truncating to r bits the encipherment of a block of a feedback buffer containing previous cipher stream blocks (or an Initialisation Vector (IV) block if first in sequence).
- in OFB each key stream block is created by truncating to r bits the encipherment of the previous key stream block (or an IV block if first in sequence).

It may be noted that CFB mode is self-synchronising (decryption can start in the middle of a message without having to know the exact position in the key stream).

Counter (CTR)

This mode operates like a stream cipher. The input message is partitioned into a stream of r-bit blocks (r<=n). Each block of this input stream is encrypted to a cipher stream block by xor'ing it with a key stream block. Each key stream block is generated by encrypting an incrementing counter block. The first counter block is initialized with a starting variable and it is important that this variable is such that the value of counter blocks never repeat during the lifetime of the key.

The five modes have different properties from one another (e.g. error propagation, self-synchronising) and security. They require specific techniques for partitioning into blocks (using padding bits and auxiliary data) and IV management. ISO/IEC 10116 [19] should be consulted for more information on these modes including their usage and security properties.

Other modes have been described in ISO TR 19038 [5]. These include pipelined and interleaved modes of CBC, CFB and OFB for TDES and Counter Mode.

Attacks on encryption in the last decade have led the industry to recognise that generic encryption should also be authenticated. Authenticated encryption modes are addressed in section 3.2.6.

3.2.1.2 Stream ciphers

A stream cipher is a symmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (keystream). In a stream cipher, each plaintext digit is encrypted one at a time with the corresponding digit of the keystream, to give a digit of the cipher text stream.

Stream ciphers may be found in ISO/IEC 18033-4 [57] and also include CHACHA20 as described by IETF in [108].

3.2.1.3 Format preserving encryption

Format Preserving Encryption (FPE) encrypts a piece of data into a form that is the same as the original format. For example, the most common application is to encrypt decimal numbers into a new number of the same length. FPE encryption of data allows the data to be processed using the same methods as the original data. Two potential applications of FPE are:

- Tokenization: for processing payment cards with encrypted card numbers (PAN), a token is generated by encrypting the card number with FPE. Where needed, the original card number can be obtained from the token by decryption.
- Generation of card numbers: simply encrypting a counter will generate all possible numbers in a random order, which can be used to make card numbers that are impossible to predict.

The first FPE algorithms were designed in 2006 and are now reaching the point where the algorithms are standardized.

NIST has standardized two different FPE functions ([127]), which are in draft at the time of this writing. Both functions use a regular block cipher as a building block, so that FPE can be used with regular AES keys.

3.2.2 Asymmetric Encryption

Due to performance limitations, asymmetric algorithms are not recommended for bulk data encryption but rather for encrypting symmetric keys and performing signatures. Therefore, the typical way of using asymmetric cryptography for encrypting data involves hybrid techniques (see next section).

3.2.3 Hybrid Encryption

Hybrid encryption combines the key management advantages of asymmetric schemes with the efficiency of symmetric schemes. Hybrid encryption can be understood as an asymmetric scheme based key establishment combined with a symmetric encryption scheme.

Hybrid encryption using RSA

To encrypt a message, a key for a symmetric encryption scheme (and optionally for a MAC) is chosen at random. The message is encrypted under the symmetric scheme (and optionally MACed). The symmetric keys are in turn encrypted with an asymmetric scheme under the receiver's public key. Both the symmetric ciphertext and the encrypted keys constitute the final ciphertext that is transmitted to the receiver. The receiver can recover the symmetric keys using their private key. They can then (optionally verify the MAC and) decrypt the message.

A good example of a hybrid encryption scheme is the RSAES-OAEP encryption scheme based on the Optimal Asymmetric Encryption Padding (OAEP) by Mihir Bellare and Philip Rogaway [146].

Hybrid encryption using Diffie-Hellman

To encrypt a message, the sender generates a random ephemeral key pair and derives a secret value using the ephemeral private key and the recipient’s public key. The secret value is then used to derive symmetric keys that are used to encrypt the message. The secret value can also be derived by the recipient using their private key and the sender’s ephemeral public key and they can then derive the same symmetric keys so as to decrypt the received message.

A good example of hybrid encryption using Diffie-Hellman is ECIES. See also section 4.2.5.

Hybrid encryption schemes are defined in PKCS#1 [112], in IEEE P1363-2000 [142] and in ISO/IEC 18033-2 [54].

3.2.4 MACs

If only data integrity or authentication is required for a message, a Message Authentication Code (MAC) can be appended to it. If in addition, confidentiality should be ensured, authenticated encryption should be applied, see section 3.2.6.

EPC recommendation 5

Although many legacy systems currently still use MACs based on DES or TDES, new systems should use CMAC based on AES, or HMAC.

3.2.4.1 CBC MACS

For CBC MACs, the MAC code is calculated as the final ciphertext block (often truncated to three or four bytes) of the CBC encrypted message. For computing MACs, the most commonly used block cipher algorithm is the TDES algorithm (more details on MAC computation may be found in ISO/IEC 9797-1 [23]).

Retail and wholesale requirements for MACs are defined in ISO 16609: “Banking - Requirements for Message Authentication using symmetric techniques” [4]. This standard, which combines and replaces two separate retail and wholesale MAC standards, identifies two approved implementations of CBC MACs that are defined in ISO/IEC 9797-1. Namely Algorithm 1 (plain CBC) using TDES as the block cipher and Algorithm 3 (CBC with two-key EDE on final block) using single DES as the block cipher. In regard to the security analyses Annex B of ISO/IEC 9797-1, the standard recommends that implementations of Algorithm 1 consider using Padding Method 3 (where the length is coded in the padding) and that implementations of Algorithm 3 consider using session keys. Furthermore, both implementations should consider using truncated MACs that are shorter than the block length (see also section 3.2.4.3).

The CMAC mode can be used to compute MAC values based on block ciphers like AES. See [125] and Algorithm 5 of [23].

3.2.4.2 Hash MACS

Applications may sometimes need to build keyed authentication functions from hash functions. This may either be because the resultant function is faster or more secure or because they happen to have access to the primitive hash function but not an encipherment function. HMAC defined in ISO/IEC 9797-2 [24] is an example of such a function. A NIST standard FIPS 198 "The keyed-hash Message Authentication Code (HMAC)" [85] which claims to be a generalisation of RFC 2104 [92] and of the withdrawn ANSI X 9.71 was published in 2002.

The MAC function described in ISO/IEC 9797-3 [25] use so-called universal hash-functions such as POLY1305.

3.2.4.3 Length of MACs

MAC algorithms often output a data block that is then truncated to form a MAC of shorter length (e.g. CBC MACs that truncate the output from 8 bytes down to 4 bytes or in some cases even 3 bytes). This approach clearly reduces the amount of data that needs to be sent to the recipient but in certain circumstances it can actually increase security as it provides a would-be attacker with less information to work with. In the case of CMAC, having a full MAC of a single block even allows to decrypt the input data, which may be undesirable.

However, an argument against the use of short MACs is that the chances of an attacker being able to guess a short MAC are greater than those for longer MACs. As a rule of thumb for estimating a minimum MAC length, the cost of guessing a MAC value must be balanced against the profit for guessing it correctly, from the viewpoint of an attacker.

For example, if a MAC is used to protect a payment of up to €100, and an attacker can get away with a few inconspicuous attempts per day at €.01 per attempt.

Then if the probability of guessing the MAC right is less than 1 in 10000, the expected gain of guessing is less than the cost of an attempt. The minimum MAC size is 14 bits in this case.

Annex B of [23] contains security analyses relating to length of MAC.

3.2.5 Digital Signatures

Digital Signature functions typically use a public key algorithm, the private key providing the signature function and the public key providing the verification function.

In most cases, a digital signature function is a complex process in which one or more transformations may be applied to the data to be signed before the cryptographic operation is performed. This process is sometimes called a digital signature scheme.

The ISO/IEC 9796 [21] and ISO/IEC 14888 [42] standards and IEEE P1363 [142] describe signature schemes using Integer Factorisation methods (i.e. RSA [137]) and Discrete Log methods. The PKCS#1 [112] specification describes schemes based on RSA.

3.2.5.1 Signatures with appendix and signatures with message recovery

The most general type of digital signature scheme is one that conforms to the hash and sign paradigm. In this case the signed message comprises the whole message in clear appended with a signature block. This type of scheme is known as signature with appendix.

If, however, the message to be signed is very small and bandwidth is limited then a signature scheme with "message recovery" may be attractive. In this case the signed message comprises part of the message in clear appended with a signature block, and that part of the message not sent in clear is recovered from the signature block during the signature verification process. The bandwidth saving (i.e. the amount of message that can be embedded in the signature and therefore recovered from it) is upper-bounded by the key size less the hash size. Thus, if the message is small compared to the key then the whole message might be recovered from the signature and no part of the message need be sent in clear.

ISO/IEC 14888 describes signature schemes with appendix and ISO/IEC 9796 describes signatures schemes with message recovery.

The EMV specifications [122] use the first of the three RSA-based mechanisms standardised in ISO/IEC 9796-2, but this mechanism is not recommended for new systems.

3.2.5.2 Length of Hash

The length of the hash is important because:

- The application of the signature scheme may require that it be collision-resistant and in this case 160-bit hashes (which provide at most an 80-bit security level) may not be suitable.

- It must be consistent with the algorithm and key size.

3.2.5.3 Efficiency

This sub-section provides a comparison between three digital signature schemes: RSA [21], DSA [78] and a digital signature scheme based on Elliptic Curves [83].

Comparison criteria for digital signature schemes

The following aspects are of importance with respect to digital signature schemes:

- Security.
- The certainty that the digital signature is uniquely linked to the signatory and to the data being signed.
- Complexity of setting up the system.
- Here a distinction is made between the one-time generation of the system parameters (used by all users) and the generation of the key-material (user specific).
- Required computational capability of users' hardware.
- The complexity of the hardware required for users to employ the system (e.g. a simple smart card, or a Pentium processor).
- Required data storage capability of users' hardware.
- The required capability for storing the system parameters, public keys and private keys.
- Required time for creating a digital signature.
- Required time for verifying a digital signature.
- Required space for storing a digital signature (i.e. the size of a signature).

To employ digital signatures not only cryptographic techniques are required, but also other security techniques (e.g. hashing, time stamping, smart cards). Moreover, also non-technical measures (e.g. procedural, organisational, legal) and services (e.g. TTPs) are required. These are not discussed in this sub-section.

3.2.6 Authenticated Encryption

If a message requires both confidentiality and integrity protection, then one may use encryption with either a MAC or a signature. Whilst these operations can be combined in many ways, not all combinations of such mechanisms provide the same security guarantees. For this reason, special dedicated constructions have been designed and standardised in ISO/IEC 19772 [61]. These constructions can provide the optimum level of security and efficiency. They typically involve either a specified combination of a MAC computation and data encryption, or the use of an encryption algorithm in a special way such that both integrity and confidentiality protection are provided. The constructions are:

- AES Key Wrap (from NIST and RFC 3394)
- CCM (Counter with CBC-MAC)
- EAX (by Bellare, Rogaway and Wagner)
- Encrypt then MAC
- GCM (Galois/Counter Mode)

These methods can provide both confidentiality and authenticity/integrity protection under the same symmetric key and provide better performance and simplified key management as opposed to separate encryption and authentication.

Nowadays it is best practice to use an authenticated encryption algorithm even when confidentiality alone is needed because the authentication can prevent manipulation attacks that seek to undermine the confidentiality. On the other hand by using additional authenticated data most of these algorithms can be used to provide authentication of some of the data without it being encrypted.

Another construction is MAC then Encrypt where the MAC is used as an IV for the encryption. This kind of method has a proof of security in the form of SIV [166] and has been standardised in ISO 20038[11] *Banking and related financial services - Key Wra*p [10] and is used to protect keys[12] using AES.

Another popular method for authenticated encryption is the combination of CHACHA20 with POLY1305 as described in [106].

For further information on authenticated encryption see [162].

3.2.7 Homomorphic Encryption

Homomorphic encryption allows the processing of data while maintaining continuous data encryption, removing the need for encrypted data to be decrypted prior to processing.

The term 'homomorphic' means that the encryption preserves algebraic structure between plaintext and ciphertext spaces. So the encryption function Enc() is additively homomorphic if

$$\mathrm{Enc}(m_1+m_2) = \mathrm{Enc}(m_1) + \mathrm{Enc}(m_2)$$

and is multiplicatively homomorphic if

$$\mathrm{Enc}(m_1*m_2) = \mathrm{Enc}(m_1) * \mathrm{Enc}(m_2)$$

If the homomorphic property holds for just one of these, for example for + but not $*$, then it is said to be partially homomorphic encryption, see for example Paillier [180] and [59]. If the property holds for both, then it is said to be fully homomorphic encryption (FHE).

FHE could become a privacy preserving data protection technique for safeguarding both personal information and commercially sensitive data from unnecessary exposure. FHE notably reduces the risk of data exposure when processing data as part of large-scale distributed calculations with potentially untrusted partners.

Standardisation of FHE is very active, see for example ISO/IEC 28033, but performance is still not very good.

3.2.8 Distributed ledger technologies

3.2.8.1 Introduction

A growing number of financial institutions have invested in distributed ledger technology (DLT) projects. The stated objectives of these investments often relate to transaction cost savings,

[11] based on ANSI TR31

[12] sometimes referred to as key bundles or key blocks

business process efficiencies, quicker service deployments, increased service accessibility and immediate transaction settlement in the absence of a central counterparty.

Distributed ledgers refer to databases distributed across multiple sites (network nodes) – typically connected in a peer-to-peer network architecture – with each node sharing a consistent copy of the database. Dedicated distributed ledger network nodes confirm the integrity/ authenticity of proposed changes and proceed to update the ledger dataset through a decentralised consensus protocol. Consensus protocol types used comprise (i) Proof of Work, whereby computational power is used to solve a hard problem before the solution is submitted for validation to other network participants (ii) Round Robin, whereby nodes take turns to introduce changes to the ledger, (iii) Proof of Authority/Identity, whereby the ability of a node to introduce changes to the ledger is determined by the identity of the entity that operates the node and the reputation of the node among network participants; (iv) Proof of Elapsed Time, whereby nodes are assigned random (and verifiable) waiting times before they are allowed to introduce changes, (v) Voting protocols, whereby candidate database changes are submitted for approval to designated/approved network nodes that subsequently vote to commit a proposed change to the database using a number of protocols[13] that may incorporate protection against corrupted/malicious or unavailable voting nodes as detailed in [195].

Distributed ledgers that have no restrictions on which nodes can introduce changes to the ledger (permission-less) often use a “Proof of Work” based consensus protocol. A disadvantage of such a protocol is the large amount of computational effort and resources that is needed to obtain sufficient security and associated power usage and dedicated hardware costs.

Committed updates to a distributed ledger are subsequently broadcasted to as many network nodes as possible to deter attempts by malicious adversaries to promote tampered versions of the database. Users will typically retain updated partial copies of the distributed ledger in their own devices and seek access to multiple nodes that contain full copies of the distributed ledger before submitting a change to its contents.

3.2.8.2 Blockchain networks

A specific type of distributed ledger is a *blockchain* where all database records are cryptographically bound (“chained”) together in contiguous blocks. Thus, a blockchain allows users to verify the sequence of committed database changes and to review all previous changes to a database record. This process makes sure that older entries are progressively harder to forge, but intrinsically delays the attainment of consensus until a number of newer blocks have been produced.

Blockchains can be categorized based on their permission model, which determines who is authorised to update them (publish new blocks). Blockchains that allow anyone to publish new blocks are categorised as *permission-less*; blockchains that allow only particular, authorised users to publish new blocks are *permissioned*. Permission-less blockchain networks underpin the operation of major cryptocurrencies. Permissioned blockchain networks are typically deployed by groups of individuals or organisations that know each other to facilitate the secure exchange of assets.

The functional characteristics of a blockchain (decentralised operation, transaction integrity protection, usability in the absence of a central authority that ensures the integrity/authenticity of database changes) made it an attractive technology to use for the first widely-used cryptocurrency

[13] Random selection of staked users, leader-based, multi-round voting, coin-aging, delegate-based

implementation launched in 2008, Bitcoin (BTC). Cryptocurrencies are digital assets that act as a medium of exchange and use some type of blockchain to secure transactions, to control the creation of new units of currency and to verify the transfer of assets across parties in the absence of a central issuing authority.

A number of cryptocurrencies are now widely used (BTC, Ethereum, Tether, BNB, USDC, XRP) for payment and investment purposes. Cryptocurrencies have attracted a lot of support (and criticism) by retail users, technology providers, financial service industry stakeholders and by financial services' regulators. Additionally, a number of central banks are assessing the economic and societal benefits of issuing digital currencies (CBDCs) that may leverage blockchain technology and digital tokens to represent a centrally-controlled digital instance of an existing fiat currency. The operation of organised cryptocurrency exchanges across the EEA and in the UK is now being regulated for anti-money laundering (AML) purposes. Most regulated financial service providers do not use cryptocurrencies in payment interactions with retail or corporate customers.

Non-fungible tokens (NFTs) are blockchain-based tokens that each represent a unique asset (a piece of art, digital/media content, sports memorabilia). Unlike cryptocurrencies, NFTs are associated with unique underlying assets and therefore cannot be exchanged equivalently with other assets of the same kind (non-fungible). Each NFT is generated by digitally signing a transaction that details the fundamental token details and attaching it to a widely used blockchain to trigger a smart contract function[14] which creates the token and assigns it to its owner. NFTs are cryptographically verifiable and can be transferred in a manner that allows the validation of their provenance and authenticity.

3.2.8.3 Security considerations

By design, distributed ledgers replicate data across a wide array of sites (network nodes); additionally, in permission-less blockchains users can review the entire blockchain at any time. This can have a potentially negative impact on the confidentiality of data stored in the blockchain that should be considered by potential users.

Blockchains use cryptographic primitives to secure underlying operations. These comprise hashes (using algorithms like SHA-256, Keccak[15], RIPEMD-160) and asymmetric key cryptography primitives (ECDSA) to verify transactions and to derive user addresses.

In that respect, the statements and Recommendations that appear in sections 3.1.1, 3.1.3, 0 and 3.2.5 on the use of (and long-term security properties afforded by) such primitives should be considered by users attempting to assess the security of a blockchain.

Key management recommendations for asymmetric algorithms - related to key generation, usage and lifecycle management - detailed in section 4.2 should also be followed by distributed ledger users.

As detailed in *Section 3.1.5* of the report and in *EPC Recommendation 5*, users of blockchain architectures are advised to assess their crypto agility to allow for the transition to the use of standardised post-quantum asymmetric cryptographic primitives if the need arises.

---

[14] ERC- 1155: Multi-Token Smart Contract Standard

[15] Selected by NIST to as the winner of the SHA-3 competition

A challenge that is specific to blockchains relates to the hijack of the consensus protocol that is used to review proposed changes and commit these to the blockchain (51% attack)[16]. To carry out such an attack, the attacker must garner resources to outpace the block creation rate of the remaining nodes of the blockchain network (holding more than 51 % of the resources applied towards producing new blocks). Depending on the size of the blockchain network, this may be a prohibitively expensive attack that can only be carried out by state-level actors. The cost to perform this type of attack increases the further back in the blockchain the attacker wishes to make a change. This attack is not technically difficult; it just carries significant cost associated with obtaining the necessary computational power. The robustness of the consensus protocol and its ability to recover from error conditions (duplicate changes caused by network latency, malicious or adversarial validator nodes etc.) is an additional parameter that blockchain users should consider.

Some blockchains have experienced transaction malleability attacks because of the digital signature (ECDSA) implementations they have used[17]. *Section 3.2.5 of this Report* offers guidance on the correct use of cryptographic algorithms to generate and validate digital signatures to afford appropriate protection to signed data.

Finally, the scalability and performance constraints of some blockchain implementations makes their use problematic for certain financial service use cases. The need to store/update all data pertaining to a blockchain may create problem for individual users if the general ledger continues to grow at a rapid rate[18]. Additionally, the speed at which a given block update is processed or committed in some blockchain network implementations may not be sufficient or acceptable[19]for certain use cases.

EPC recommendation 6

Financial service providers that decide to deploy distributed ledger-based services or processes should:

- Confirm the security properties afforded by the distributed ledger to its users,
- Identify the cryptographic primitives used to validate and commit changes to the ledger and confirm the status of cryptanalytic attacks targeting these primitives.
- Identify the consensus protocol used to commit changes to the ledger and assess the feasibility of a successful consensus hijack attack.

Further information on distributed ledger technology may be found in [79], [186] through [191], [194] and [195]. Additionally, the work of international standardisation bodies on blockchain and distributed ledger technologies (e.g. ISO/TC 307) should be monitored.

3.3 Domain of Application

Algorithms may be selected according to the domain of application for which they were designed or for which they are best suited. Such domains may be:

---

[16] See "*Majority Attack*" in the Bitcoin Wiki, https://en.bitcoin.it/wiki/Majority_attack

[17]Transaction Malleability (Bitcoin Wiki)

[18] For example, the Bitcoin blockchain was 310Gb (in November 2020) and had grown 25% in size in the last 12 months.

[19] The Bitcoin blockchain is restricted to a sustained rate of 7tps, Ripple (XRP) can process 1700tps. The international payment card schemes can process over 20,000tps.

- Electronic and mobile banking,
- On-line transactions - retail (PIN generation or PIN verification),
- On-line transactions - host to host,
- Batch transfer of individual transactions,
- File transfer (EDI),
- Email,
- Electronic purse,
- Debit or credit cards,
- Secure back-up
- ...

When the domain of application is concerned, typical criteria that may be used to select the most appropriate algorithm and key management scheme may be:

- Expected performance (including response time requirements in interactive systems),
- Volume of data to protect,
- Key management suitable for an open or closed user group,
- Necessity to use a Trusted Third Party and acceptance of such a TTP by the users,
- Scalability,
- Costs (to set up the system and to keep it running),
- Perceived sensitivity of the data to protect,
- Requirements on how long time protection is needed
- ...

3.4 Implementation and interoperability issues

3.4.1 Security protocols

As soon as different applications using cryptographic techniques must interact many design issues must be addressed. The choice of appropriate algorithms and modes of operation is the most obvious decision to make but is far from being sufficient. Data formatting issues, padding or redundancy rules, filtering of binary results, interactions between compression, confidentiality and integrity techniques, scope of application of the algorithms and bit ordering issues are examples of typical sources of operational difficulties.

Solving such problems may be easy in a closed user application, but if interoperability within an open environment is required, then all these issues must be resolved, if not by a previous agreement then automatically set up by means of a security protocol.

SSL or S/MIME in the Internet world are examples of such security protocols.

This issue is not covered satisfactorily by any international standard. The standards from the set of ISO/IEC 10181 "framework standards" [12] define the conceptual models for authentication, access control, non-repudiation, integrity, and confidentiality, but are not aimed at being sufficient to define unambiguously a security protocol.

TLS 1.3 (see [107]) is a new version of the TLS secure messaging protocol improving security over the previous versions by:

- fixing flaws in the protocol;

- removing support for insecure cryptographic protocols and ciphers;
- introducing the use of "ephemeral key exchange" to provide security against adversaries that store communication to break it in the future when private keys are exposed;
- providing a mathematical proof that design requirements are satisfied by the reference implementation (see [197]).

Since the publication of TLS 1.3 on March 21, 2018, it is time to reconsider which versions of TLS are sufficiently secure. At least, TLS 1.0 and 1.1 should no longer be used. Since its introduction in August 2008, TLS 1.2 provides protection against many security problems of the earlier versions of the protocol. All earlier TLS versions were supposed to have been updated a few years ago.

The restriction of the use of earlier versions of TLS (TLS 1.0/TLS 1.1) will help address communication security flaws; the enforcement of such restrictions must be the highest priority for financial service providers in order to secure remote communication sessions with their customers, partners and 3rd party service providers. The large internet browser vendors (Google, Apple, Microsoft and Mozilla) have all removed TLS 1.0 and TLS 1.1 support from their respective browsers; all recent versions of mainstream browsers already support TLS 1.3 (see https://security.googleblog.com/2018/10/modernizing-transport-security.html). Enabling (and promoting) the use of TLS 1.3 further increases security for remote communication sessions. The continued usage of TLS 1.2 is still acceptable to ensure backwards compatibility with legacy systems that are less frequently updated, especially if used in conjunction with ephemeral DH/ECDH cipher suites or TRSM -protected Server keys.

- The continued use of legacy systems that are still forced to use TLS 1.1 or TLS 1.0 for remote communications must be subjected to ongoing security risk assessment; those earlier TLS versions notably have known and easily exploitable vulnerabilities, under any of the following conditions:
  - The TLS connection is public, i.e. exposed to the Internet,
  - The TLS connection is used for browser-based access.

EPC recommendation 7

- Use TLS with secure cryptographic primitives and appropriate key sizes (c.f. 3.1.3.4)
- Enable TLS 1.3 support in all new systems (offers forward-secrecy by default)
- Enforce the use of TLS 1.2 or higher for all use cases (preferably with ephemeral cipher suites)
- Do not use TLS versions older than TLS 1.2 because of known and exploitable vulnerabilities (unless such use is approved in specific use cases through ongoing security risk assessment).

3.4.2 Data formatting issues

Whenever hash functions are used for data integrity, it is essential that the input data for the hash functions is exactly defined, such as padding, representation, field sizes and the use of binary data.

The evolution of so-called 'padding oracle attacks' continues. Such attacks are particularly relevant to decryption algorithms that reveal whether a possibly tampered ciphertext is correctly formatted when decrypted. These attacks are especially relevant to PKCS#1v1.5 [112], PKCS#11 [119] and CBC encryption and may be avoided by the detection of such tampering (e.g. by using authenticated encryption).

3.4.3 Implementation rules

As soon as different transformations must be applied to the data, the order in which these transformations are performed may be important. The transformations and their purpose are as follows:

- Encryption: to protect confidentiality - this may be absolute (such as for cryptographic keys, not to be seen as plaintext outside a TRSM) or for transmission or storage
- Data compression: to reduce message or storage size
- Digital signature or MAC: to give origin authentication (asymmetric cryptography only) and data integrity - also for use in authentication
- Legal signature (asymmetric digital signature): to achieve the equivalent of a handwritten signature - which may achieve non-repudiation in the legal sense of providing evidence to a third party or a court as to intent (also known as electronic signature).

The following rationale should be considered in conjunction with the recommendations given below:

- Sending encrypted data that is not integrity protected enables attacks on encryption keys and plaintext through error analysis: it is better to encrypt first and then sign or MAC or to use a combined mechanism as set out in section 3.2.6, so that the receiver can verify the integrity of the ciphertext. But note: some signature algorithms are weaker when signing unintelligible data.
- Signing the data in plain text format is necessary for legal signature, but not for other purposes.
- Performance of encryption is better if there is less data to encrypt.
- Data compression of encrypted data has no effect.

The following basic principles should be respected:

- Separate keys should be used for each transformation (where relevant).
- Keys should be reserved for specific associated functions, and this association should be protected. For example, keys used for legal signature might be associated with that purpose through a public key certificate.

EPC recommendation 8

To achieve confidentiality and integrity protection:

- For the originator, sign the plaintext data first for legal signature (if required), then compress (if required), then apply authenticated encryption with the non-confidential data being treated as associated data (not encrypted).
- For the recipient, perform the steps in the reverse order. Verify that signatures are from an authentic source.
- The encryption and signature/MAC can be performed as separate steps or can be achieved by use of authenticated encryption (which also allows the use of traditional encrypt-then-MAC) or signcryption.

3.4.4 Key management impact on interoperability

As long as cryptographic techniques are used within closed user groups, any technically suitable and sufficiently secure key management scheme may be used.

When cryptographic techniques are used in an open environment, key management issues may become the major obstacle to interoperability.

It is commonly agreed that public key technology is better suited to open environment than symmetric algorithm technology. If symmetric algorithms are required and secret keys must be exchanged then using a mix of public key for key exchanges and secret keys for encryption is a good solution.

As long as the application concerned may be considered as "low-risk”, use of commercial CA services may be an acceptable way of solving this interoperability issue. One danger is to use the root keys of too many commercial CAs simultaneously (such as is the case with current internet browsers). The security of such a system is as strong as the weakest of the CAs in the list.

3.4.5 Implementation quality and side-channel attacks

A bug in an algorithm implementation can transform a good algorithm into a bad one. Therefore, quality control during the development process and testing against other implementations of the same algorithms are crucial.

Implementation of protocols is equally important for the security of the protocol, and needs as much care as the implementation of the underlying protocols.

For high risk applications usage of evaluation criteria (Common Criteria or FIPS 140-3 - [81]) and of formal description languages may be considered.

Any component of a protocol using secret data is a potential target for side-channel attacks (e.g. timing attacks, glitch attacks, DPA). Generally, side-channel attacks need special equipment – for example malicious software on POS devices cannot be used for side-channel attacks.

It should be noted that timing attacks can be attempted at long distances, for example [153] describes timing attacks on SSL implementations performed over the Internet.

3.4.6 Algorithm OIDs

OIDs or Object Identifiers are a mode of identification of objects originally defined in ASN 1 syntax, which is widely used today. For example, X 509 certificates make a very extensive usage of OIDs.

International standardisation bodies (e.g., ISO, IEC, ITU) have defined a hierarchical way of assigning OIDs, such that the same OID may not be assigned twice to two different objects.

On the other hand, nothing prevents standards writers or users to define new OIDs for an object, which has been already given one elsewhere.

So when designing a security protocol, or when creating a certificate request one must consider which OIDs will be used to identify the algorithms and algorithm parameters.

Choosing the correct OID is not always straight forward and implementers should refer to the relevant standard for the OID of the algorithm they are implementing.

4 Key Management Issues

This section focuses on issues related to key management. Section 4.1 addresses the management of keys used with symmetric algorithms (i.e. symmetric keys) and Section 4.2 addresses the management of keys used with asymmetric (public key) algorithms (i.e. asymmetric keys).

These Guidelines do not address any particular commercial products for key management. Specific key management techniques may be imposed by the device being used and in this case the device vendor's instructions should be followed.

4.1 Symmetric algorithms

Many systems - especially those providing general security protection - are implemented using symmetric block ciphers with symmetric keys of various lengths, for example TDES with double-length 112-bit keys or triple-length 168-bit keys or AES with 128-bit, 192-bit or 256-bit keys.

Many systems – especially those using asymmetric keys to encrypt data – will generate ephemeral random or derived symmetric keys that are used once and then can be erased. Because of the short-lived nature of these symmetric keys, their management is not the main concern of this section (but see section 5).

Symmetric cryptographic keys, whether long-lived or short-lived, must be managed and protected throughout their lifecycle – from generation to active use to de-activation and destruction. Such key protection includes both key confidentiality and key integrity, often with separate "superior" keys providing various levels of protection. This section of the Guidelines primarily discusses key confidentiality for symmetric keys, however the requirements for key integrity are similar and may be accomplished by a key check value or MAC or by using an authenticated encryption algorithm.

4.1.1 Key generation and derivation

Keys should be generated in a TRSM (Tamper-Resistant Security Module) – as described in standard FIPS PUB 140-2 [81] and ISO 13491. The TRSM should have a master key that is generated inside the TRSM and never exists in clear text outside the TRSM.

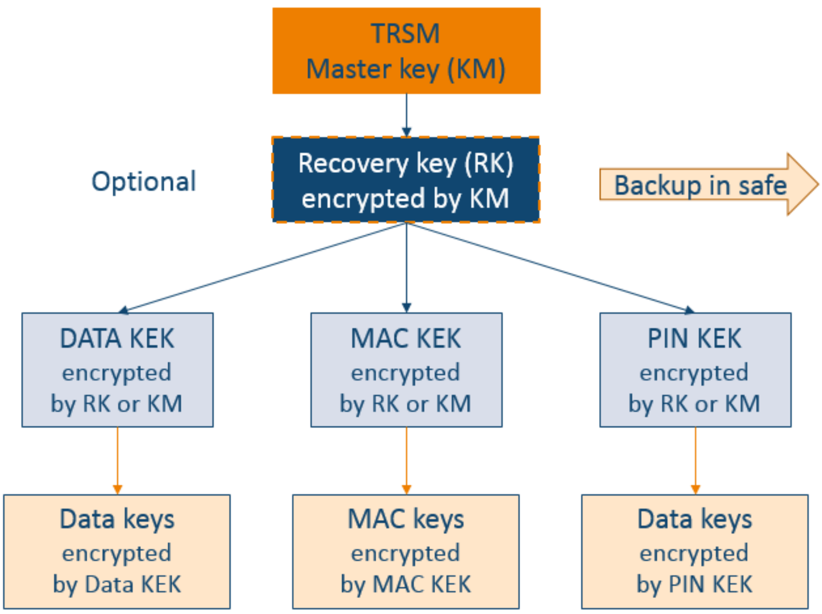

Figure 2: Example of key hierarchy for symmetric keys

Key encrypting keys are also generated in the TRSM but may be exported for backup/storage encrypted under the master key (or a dedicated recovery key that is encrypted under the master key). Master key generation is therefore the most sensitive as this key encrypts all other key encrypting keys.

Access control mechanisms should be implemented that provide Dual Control over the TRSM. This prevents a single person from generating or accessing the master key on his or her own.

- Access control mechanisms are used to control access to the systems operations. Typically, the users must log on to the system with a password or a PIN and a smart card to get access to the operations. In key management, dedicated smart cards are often used because they are more secure than passwords[20].
- Dual control means that the operation needs (at least) two people. Typically, in the case of master key generation the first person logs on to the system and generates the first key component in the TRSM (which in the case of TDES is a full-length 128 bits or 192 bits component with parity bits). Then the second person logs on to the system and generates the second key component in the TRSM. The two key components are now combined (e.g. xor'ed together) and the result is the new master key. This scheme also supports backup of a master key whereby each component is securely retained by the person that generated it so that no single person knows the master key. More complex secret sharing systems such as Shamir's threshold scheme may be used (described in [141] and now standardised in ISO/IEC 19592). When generating a new master key for a TRSM all keys protected by the old master key must be decrypted and re-encrypted under the new master key. This is an example of **key translation** and specially designed secure functionality must be supported in the TRSM for this purpose (see the conference paper in "The unbearable lightness of PIN cracking" https://www.researchgate.net/publication/220797019_The_Unbearable_Lightness_of_PIN_Crackingin [198].

For key generation, it is critical that the TRSM use a good random source. Many implementations of cryptosystems have been broken on account of an inadequate source of randomness or 'pseudo-randomness'. This topic is discussed in [141] where an extensive bibliography on randomness may be found. See section 5 of these Guidelines.

Key derivation techniques are used when deriving keys from other keys (for example, card keys derived from issuer keys, session keys derived from fixed keys and symmetric keys derived during asymmetric key agreement) and from passwords or other secret data. Techniques for deriving keys from passwords such as PBKDF2 are standardised in NIST SP 800-132 [134] and PKCS#5 (IETF RFC 8018) [114]. Other techniques include SCRYPT (IETF RFC 7914) [103] and Argon2[21].

4.1.2 Key backup and storage

Subsidiary symmetric keys (i.e. those other than the master key) can be stored inside the hardware or encrypted outside. The advantages and disadvantages are comparable. Multiple copies of encrypted keys may be made in case a single instance is destroyed, but an intact master key will still be needed to decrypt it within the TRSM.

---

[20] A PIN usually protects usage of smart cards, thus access control based on smart cards requires both possession of the smart card and knowledge of the PIN.

[21] https://password-hashing.net/argon2-specs.pdf

EPC recommendation 9

If a master key needs to be backed-up outside of a TRSM then it should be split into key components in a secure and resilient manner. Secret sharing techniques should be used to allow recovery of the master key from all or a fixed number (threshold) of the master key components. The security level of the storage of the master key components should be commensurate with the protection afforded to the operational master key itself. Unless derived, subsidiary keys should be backed-up encrypted under a master (or a dedicated recovery key).

4.1.3 Key distribution

In some hybrid encryption methods (ones using both symmetric and asymmetric cryptography) data is protected using a symmetric key, and the symmetric key is distributed encrypted under the recipient's public key. Assuring the authenticity and integrity of this public key is paramount and techniques addressed in Section 4.2 may be used.

When using only symmetric cryptography, key-encrypting keys (KEKs) must be distributed or a hierarchic key derivation scheme may be used. KEKs are used to encrypt other keys. After the first KEK is interchanged between the parties, this KEK can typically be used to encrypt the successor KEK and so on. The first KEK distribution is then called the initialisation exchange. The security of this exchange must be very high, this means physical attendance of bank personnel who are responsible for the security of the system in question. If any of the KEKs is compromised then all its successor KEKs are assumed compromised. In this case a re-initialisation exchange is needed, and the old keys must be deleted.

The most difficult part of key distribution is the initialisation phase. It may require manual procedures such as:

- key components in sealed envelopes or smart cards
- face-to-face identification
- key components to be distributed by different channels
- installation of a security module with pre-installed keys.

For other symmetric key distribution schemes refer to ISO/IEC 11770-2 [37] and other references provided at the end of this section.

4.1.4 Key installation

Often symmetric keys are generated in one place and are used somewhere else. This could be two different machines inside the same organisation or it could be two parties in a communication, e.g. two payment service providers.

There are several ways to install these keys:

- Key split into key components: the key is split into components that must be combined (e.g. xor'ed together) to form the actual key. At the most elementary level the key components may be securely printed in plaintext on different pieces of paper (key component letters). A separate trusted person holds each key component letter thereby providing split knowledge and dual control. This way of installing keys can be used when installing initial keys, e.g. initial KEK's between parties with different master keys.
- Smart cards: The key component letter installation method can be done more securely using smart cards (chip cards) with PIN. Each smart card contains a component of the key.

When installing the key the person holding the card enters the PIN on the smart card reader and installs the component of the key.

- Encrypted installation: The key which is going to be installed is encrypted by a key-encrypting key (KEK). The sender's KEK must already be installed in the TRSM of the recipient and so no additional dual control is needed. This method is the simplest and most often used – the use of fragmented keys is only necessary to initialise or reinitialise a secure communication between parties.

When installing a key, especially if manual methods are used, its integrity should be verified. This can either be done using a MAC (using a pre-established Key MAC'ing Key) or by a key check value, which is calculated when the key is generated. This value is distributed with the key and the receiver of the key makes the same calculation and compares his result with the key check value. If they are equal the key is valid.

A common way to calculate a key check value for verification is to encrypt a fixed string (e.g. zeros or a pre-selected number) with the key, and part of this result, typically the first six positions, is then the key check value. It must be noted that these key verification techniques are intended only to allow error detection during the key installation process.

4.1.5 Key usage and key separation

To increase the level of security, one key should be dedicated to only one usage. Furthermore, the keys and their intended usage should be connected in a reliable way (key wrapping). Key usage information tells the system, what the key can (and cannot) be used for. This is sometimes called the key control vector (CV). All cryptographic operations other than those specified in the CV will fail. The control vector method only works if all parties involved use compatible CV's with the same certified TRSMs that enforce the key usage policy.

Key wrapping methods using AES and Triple DES are standardised in ISO 20038 [10] and X9.143 [80].

The PCI SSC has mandates on the use of key wrapping in the form of "key blocks":

- https://www.pcisecuritystandards.org/documents/Cryptographic_Key_Blocks_Information_Supplement_June_2017.pdf
- https://www.pcisecuritystandards.org/documents/PIN_Security_Rqmt_18-3_Key_Blocks_2019.pdf
- https://docs-prv.pcisecuritystandards.org/PIN/Supporting%20Document/PIN_Security_Rqmt_18-3_Key_Blocks_2022_v1.1.pdf
- https://www.pcisecuritystandards.org/pdfs/PCI_SSC_Bulletin_on_Key_Block_Equivalents_Final.pdf

Use of TRSMs that make use of control vectors and key wrapping is recommended so that at least within a system the keys are used only for their intended purposes. See also NIST SP800-38F [126] [166].

Irrespective of the key wrapping technique used, key separation should be implemented.

EPC recommendation 10

Symmetric keys should be dedicated to one usage (e.g. encryption or MAC computation, but not both).

EPC recommendation 11

Key usage controls (e.g., making use of control vectors, key wrapping) which bind the key to key control information in a secure way should be employed.

4.1.6 Key deletion

Keys should be deleted when they are compromised or expired. Key component letters should be destroyed as well. Storage media with old keys should be destroyed magnetically. EPROM should be destroyed physically.

4.1.7 Key cryptoperiod

General principles of key cryptoperiod management should be based on ISO 11568 [6]**Error! Reference source not found.** and NIST SP 800 57 [129].

Whenever possible, session keys should be used. This means that instead of using the same key for many "sessions", different keys are used for different sessions. These session keys are typically derived (securely) from a parent key so it is infeasible to determine the parent key from the derived key. For example, in the case of DUKPT XX9.24-3 [69] each session key is derived from its predecessor using a 1-way function.

KEKs are less exposed than session keys and consequently their cryptoperiod may be longer. The precise frequency at which KEKs are changed will vary according to the level of risk – for low risk applications years may be enough, whilst for higher risk applications changing the KEK after several hours or days may be needed.

Master keys are even less exposed (but of course are even more valuable) and so may be kept for longer periods (many months or even years depending on the risk exposure and type of application). In determining the cryptoperiod for master keys, the procedural risks surrounding over-frequent initialisations should also be considered, against infrequently invoked, perhaps forgotten, procedures.

4.2 Asymmetric algorithms

This section describes key management for asymmetric algorithms, i.e. public-key cryptography such as RSA or ECC. For other key distribution schemes refer to the Diffie-Hellman protocol [113], [142], ISO 11568 [6] and ISO/IEC 11770-3 [38].

Asymmetric algorithms use key pairs comprising a public key and a private key. A key pair has several phases in its lifetime:

1. Generation phase. The key pair is generated and waiting to be activated.
2. Usage phase. The private and public keys are used.
3. Verification-only phase. The private key is no longer used (it may be archived or terminated) but the public key is still used e.g. for verification of issued certificates or digital signatures. This phase will last as long as the issued certificates or digital signatures are valid. This phase doesn’t apply for key pairs used for data or key encryption.

4. Decryption-only phase. The public key is no longer used (for example because the public key's certificate has expired) but the private key may still be used for decrypting previously encrypted data or keys. This phase will last as long as the scheme requires. This phase doesn't apply for key pairs used for digital signatures.
5. Archival phase. The private key and/or the public key are archived from normal operational use. This phase could be omitted.
6. Termination phase. The private key is deleted, the public key certificates are revoked if not already expired.

4.2.1 Key generation

Keys should be generated in a dedicated hardware security device.

The hardware should contain a TRSM (Tamper-Resistant Security Module) – described in standard FIPS PUB 140-3 [81], or in ISO 13491 [7]. The keys should be generated inside the TRSM and the private key should never exist in clear text outside the TRSM.

EPC recommendation 12

Keys should be generated inside the TRSM and the private key should never exist in clear text outside the TRSM.

As with the generation of symmetric keys, it is critical to use a good random number generator. Refer to section 5 for information on random number generation.

The generation of asymmetric keys will often imply usage of prime numbers and recommendations may be found to use 'strong' primes. Strong primes are prime numbers generated to have a particular structure that makes them more impervious to particular cryptanalytic attacks. Nowadays it is less necessary to explicitly require that prime numbers be strong, this is for two reasons: because all randomly generated primes of the size used in modern cryptographic systems will be strong with overwhelming probability anyway and, to a lesser extent, because new attacks such as elliptic curve factorisation are 'immune' to weak primes.

Algorithms used for generating primes for RSA should have been publicly scrutinized (see Recommendation 1) and should not be susceptible to Coppersmith's attack[22] (see for example the security flaw in the Estonian ID card[23] [192]). ISO/IEC 18032 [53] specifies algorithms for generating primes numbers that can be used for creating RSA moduli.

4.2.2 Example of a hybrid key architecture

Private and public keys may be deployed within a fixed hybrid key hierarchy, for instance with the following keys, as shown in Figure 3:

- Master key: stored inside TRSM. Typically, a symmetric key – e.g. double- or triple length DES key or AES key.
- Key-encrypting key (KEK) – optional. Typically, a symmetric key – e.g. double- or triple length DES key or AES key. Encrypted by the master key.

---

[22] see for example https://crocs.fi.muni.cz/public/papers/rsa_ccs17

[23] https://www.schneier.com/blog/archives/2017/09/security_flaw_i.html

- Private key: e.g. 2048-bit RSA key – with corresponding public key. The private keys are encrypted by the master key or a key-encrypting key when outside the TRSM.
- Public key (corresponding to a private key) authenticity may be protected with a certificate created by a Certification Authority signature. Certificate Management and Certification Authorities are complex subjects warranting separate discussion which is outside the scope of this document (**Error! Reference source not found.** and [11]).
- Session key: symmetric key used in a protocol between nodes in a network. The session key may be randomly generated and encrypted with the correspondent parties public key or it may be derived as a function of the correspondent parties keys (see section 4.2.5).

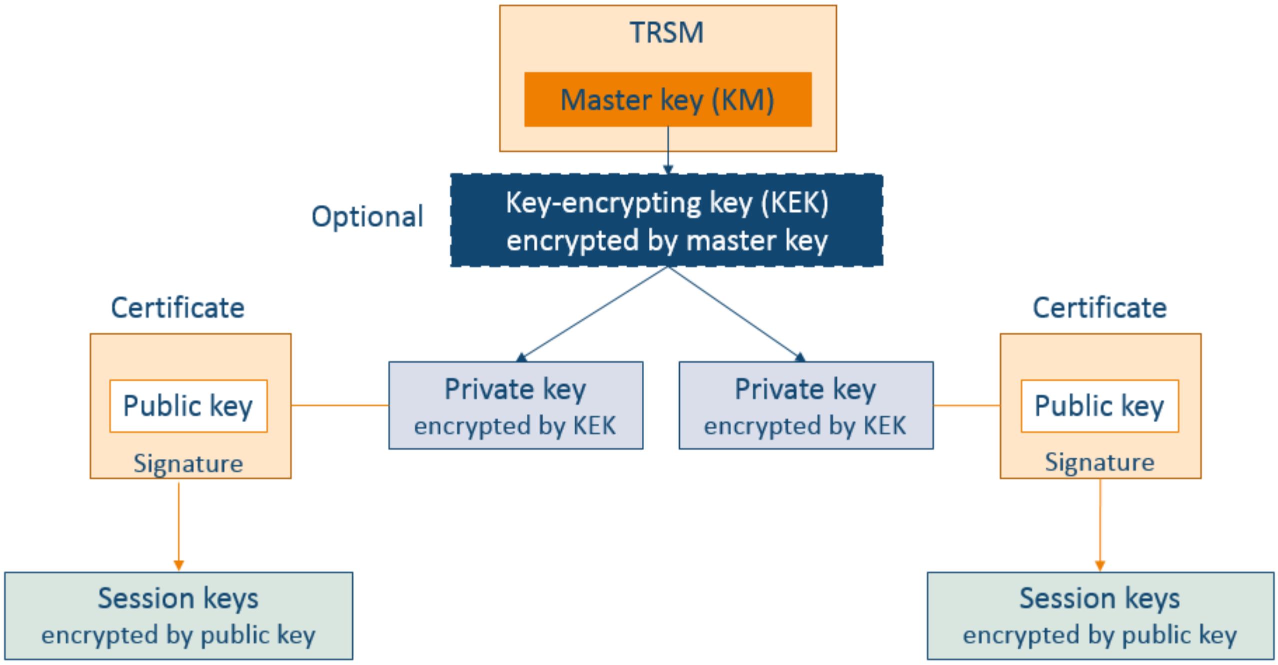

Figure 3: Example of a hybrid key hierarchy with asymmetric and symmetric keys (for data confidentiality)

4.2.3 Key backup and storage

Public keys may be stored in plaintext in certificates outside the cryptographic hardware. Private keys can be stored in various ways, for example:

- Inside one piece of cryptographic hardware in plaintext (or even encrypted under a master key).
- Outside but encrypted by e.g. a TDES or AES key (KEK). This KEK key must be stored inside the hardware or stored outside encrypted under the master key or another KEK. When a private key is needed it is taken into the cryptographic hardware and decrypted by the KEK inside the TRSM.
- Inside many pieces of hardware but in plaintext fragments (or even encrypted under a master key) - securely and resiliently fragmented in such a way as to require co-operation of a threshold number of fragments in order to operate the private key.

The first two ways both have a very high level of security. The advantage of the first one is performance, but there is no backup of the keys. The second one has an easy way to make backup and the scalability is better, because only one key is needed inside the cryptographic hardware. The disadvantage is that the security of the KEK must be very high, a compromise of this key would lead to disclosure of all other keys and destruction of this key (and any backups it may have) would effectively destroy the private key that it protects.

The third approach is described in ISO 21188 [11] and can provide the resilience of the second approach with the security of the first, however the fragments must be managed carefully. This approach can be modified to provide a secure procedure for backup (as opposed to operational storage) of private keys. With this modification a backup copy of the private key is fragmented using techniques similar to those used for symmetric keys – a threshold number of key fragments are needed to reconstitute the key (and only then can the reconstituted key be operated).

4.2.4 Key distribution

The private key usually remains on the hardware that generated the key pair or within a secure storage environment (e.g. smart cards). However, the public key must be distributed to the parties that are intended to communicate.

Distribution of the public key is a very important security issue. The recipient of the public key must be absolutely sure of the authenticity and integrity of the delivered public key, i.e. she must be sure that the key belongs to the declared owner of the key and that the key has not been modified after its creation. If for example, the original public key is replaced with a public key from a malicious person or organisation, then the recipient will be misled to accept false signatures on transactions, or send data encrypted with the false public key, which can be decrypted by the malicious entity. So, the risk of public key substitution during its distribution requires the deployment of appropriate security controls.

There are various methods to distribute or validate public keys. For example:

- Electronically (e.g. download from a web site).
- Delivered on a physical storage medium.
- Public key in a certificate.

In the first method, the integrity and authenticity of the distribution channel has to be ensured e.g. through the use of Transport Layer Security v1.2 or higher ([101], [107]) or by a multiplicity of controls (e.g. key hashes/checksums published on a website or sending key hash information over a different channel than the one used to distribute the public key). The disadvantage of the second “physical” method is that the key has to be delivered in person, which is almost always impractical in large user communities, or by registered post where the recipient must be sure that the key is delivered in an untampered envelope. Neither of these methods preserves the integrity of the public key after installation on the recipient’s device unless a hash or a MAC is applied. Where the operation of a Certification Authority (CA) is practical and appropriate the use of public key certificates to distribute public keys is recommended.

EPC recommendation 13

Public keys should be distributed in a manner that preserves their integrity and the binding with the legitimate key owner (e.g. by using verifiable public key certificates or an out-of-band method).

The widely used standard format for certificates is X.509 [18].

EPC recommendation 14

Usage of X 509, version 3, format certificates is recommended.

Some other techniques related to public key authenticity and certificates are:

- CRL (Certificate Revocation Lists) and OCSP (Online Certificate Status Protocol) – see section 4.2.7.
- Certificate Pinning (see [199]).
- OCSP stapling (https://en.wikipedia.org/wiki/OCSP_stapling).
- Certificate transparency (https://en.wikipedia.org/wiki/Certificate_Transparency).
- Cascading bloom filter (currently used in browsers, replacing OCSP). This method appears to be the new standard for browsers at the time of this writing.

Often there is confusion between public key certificates and attribute certificates. A public key certificate binds the identity of a person or an organisation to a specific public key. In an attribute certificate different privileges of a particular user are bound to the identity of the user or the serial number of their public key certificate. The lifetime of an attribute certificate is foreseen to be shorter than the lifetime of a public key certificate.

4.2.5 Key agreement and forward secrecy

If the requirement is to establish a shared secret key (which might then be used for sending keys or messages from one party to another protected using symmetric techniques) then it is important to consider Diffie-Hellman key agreement and its EC-based variant ECDH (see ISO/IEC 11770-3 [38]) in conjunction with key derivation techniques (see ISO/IEC 11770-6 [41],NIST SP800-56c [128] and NIST SP 800-108 Revision 1 **Error! Reference source not found.**). If ephemeral EC keys a re used in the Diffie-Hellman key agreement then it is possible to achieve forward secrecy whereby an attacker (who subsequently compromises the sender's or recipient's key store) is unable to determine the shared secret key that was established and the payload it protected. This property is enforced by TLS v1.3 (see [107]) and IPsec.

4.2.6 Public Key installation

Before an installation of a public key can take place, the validity of the public key and related parameters must be verified. This is done either as part of the distribution process or, if certificates are used, by using the public key of the CA which issued the certificate. The process of verifying certificates in a chain back to the trusted root public key is called validating the certification path.

4.2.7 Certificate revocation and expiry

If a private key is compromised then the associated public key certificate must be revoked. The CA must keep updated information about certificate revocations. When verifying a certificate, users must have access to the most recent information about the certificate status. This may be provided by the CA distributing regularly Certificate Revocation Lists (CRL) or giving interactive access to a certificate status database through an on-line protocol (OCSP).

A key pair and the related certificate have a lifetime that must be indicated in the certificate by a validity period. As soon as this validity period expires the public key must not be used any longer. Copies of expired certificates must be kept as long as there may be a need to verify signatures generated by the corresponding private key (the use of a trusted archive or time stamping techniques may be needed in this case – see ISO/IEC 18014 [48]).

When validating a certification path, it should be checked that no certificate in the path is present in a Certificate Revocation List (CRL), nor expired.

EPC recommendation 15

When verifying a digital signature, users should check that the certificate status (either Valid, Expired or Revoked) at the time the signature was generated does not render the signature invalid. When a certification path is involved, this verification should be performed for every certificate in the path up to the root certificate. Efficient verification may require that the date and time when the signature was produced is unambiguously defined.

If certificates are not distributed along with the signed message, then users should either have online access to up-to-date certificates or they should keep track of certificates in local databases and update the database when new certificates are issued.

A timestamp may be included in the signed message so that the signer can attest to the time the signature was created. An agreed time source may be enough for some implementations, however there are also certified time sources available, i.e. Time Stamping and Notary Services, which are being supplied by trusted third parties. Trusted timestamps allow the user to be sure of when the message was signed (in case they do not trust the signing entity for such things).

Similarly, a user must have access to a trusted time source when verifying signatures and certificates containing dates and times.

EPC recommendation 16

Whenever possible, trusted time sources should be used, in order to allow reliable verification of certificate validity.

4.2.8 Key usage and key separation

In contrast to symmetric keys, public and private keys have different usage. For example:

Public key usage:

- key and data encryption,
- digital signature verification.

Private key usage:

- key and data decryption,
- digital signature creation,
- certificate signing for issuing certificates,
- CRL (Certificate Revocation List) signing.

Furthermore, an entity with a private key can authenticate itself to an entity that has the corresponding public key, and key pairs can also be used for establishing shared secret keys.

It is good security practice to have separate key pairs for different purposes, e.g. a separate key pair reserved for signing and another for encryption. In that way key recovery can more easily be implemented, as the private decryption key can be stored at a trusted place apart from the private signing key. Another advantage of key separation is that a large key is often required for signing, whereas shorter keys may be sufficient for short-term encryption, and different certification policies may apply to encryption and signature keys. X 509 certificates [18] contain an extension which indicates the purposes for which the certified public key is used.

EPC recommendation 17

An asymmetric key pair should be dedicated to one usage, for instance: one of entity authentication, data integrity, symmetric keys' encryption.

4.2.9 Key deletion and archiving

If a private key is suspected of being compromised the private key should no longer be used by the owner. Then the owner should delete any copy of it and request the prescribed entity (e.g., the CA) to revoke the relevant certificate. However, it may be necessary to archive the certificate as long as verification of signatures produced with the private key may be required.

4.2.10 Key crypto period

The crypto period of an asymmetric key pair is related to the size of the key and therefore to its cryptographic strength. A typical crypto period for an RSA key pair will be measured in years. Because ECC keys are much easier to generate their crypto periods can be much shorter, indeed the keys can be ephemeral (see section 3.2.3). Considerations on key length and crypto period may be found in **Error! Reference source not found.**, [96], [140], [148], and [149].

4.3 Key recovery and key escrow

Sections 4.1.2 and 4.2.3 noted the importance of key backup to handle the situation where an operational key is accidentally destroyed or becomes unusable, for example in case of a natural disaster. This is referred to as key recovery.

An organisation may also need to make copies of secret keys available to law enforcement so as to meet national requirements on lawful access to data (see [131] and **Error! Reference source not found.**). In such cases the keys should only be made available to appropriately authorised entities and with active participation of the organisation. This is referred to as key escrow.

EPC recommendation 18

Where possible, payment service providers should avoid the use of key escrow, but key recovery is part of their due diligence and business continuity obligations.

4.4 Additional information and ISO standards

More information on this topic may be found, for instance in: HAC [141], ISO 11568 [6], ISO/IEC 11770-1 [36] , ISO/IEC 9594-8 [18], Davies and Price [139].

ISO 11568 [6]is the ISO standard for retail financial services key management.

Regarding asymmetric algorithms:

- On Certification Authorities and public key certificate management, the reader is referred toand ISO 21188 [11].
- Some PKCS specifications cover certificate management: certificate format in PKCS #7 [116] and certificate requests in PKCS #10 [118].

4.4.1 ISO 11568 Financial services – Key management (retail)

A new edition of ISO 11568 [6] has now been published that replaces the old multi-part Standard.

The standard describes the management of symmetric and asymmetric cryptographic keys that can be used to protect sensitive information in financial services related to retail payments. It covers all aspects of retail financial services, including connections between a card-accepting device and an Acquirer, between an Acquirer and a card Issuer, and between an ICC and a card-accepting device.

It covers all phases of the key life cycle, including the generation, distribution, utilization, archiving, replacement and destruction of the keying material. It covers manual and automated management of keying material, and any combination thereof, used for retail financial services. It includes guidance and requirements related to key separation, substitution prevention, identification, synchronization, integrity, confidentiality and compromise, as well as logging and auditing of key management events.

Requirements associated with hardware used to manage keys have also been included.

It does not specifically address internet banking services offered by an Issuer to their own customers through that financial ’institution's website or applications.

Nor does it address using asymmetric keys to encrypt the Personal Identification Number (PIN) or any other data, or asymmetric keys managed with asymmetric keys.

It does not intend to apply to the management of the keys installed in an ICC during manufacturing or the initial key established in an ICC during card personalization.

5 Random Numbers

The production of high-quality random numbers is critical for cryptographic applications, and indeed some implementations of cryptosystems have been broken on account of an inadequate source of randomness or “pseudo-randomness”. An adequate source of randomness makes sure the generated numbers are not predictable.

The importance of good random number generation has received wide attention, especially in the area of prime selection for RSA key generation (see 4.2.1) where some internet research (see [173]) has revealed that a significant proportion (e.g. thousands) of TLS/PGP public keys shared common factors (and so could be broken). The cause of this is bad RSA key generation probably resulting from the use of bad random number generators or poorly seeded random number generators.

Random numbers can be generated using true hardware RNGs or using deterministic algorithmic methods, preferably both. If a deterministic random number generator is used, it must be seeded with an unpredictable source of information. Recommendations and requirements for generating random numbers can be found in the following standards:

- ISO/IEC 18031 and ISO/IEC18031:2011 Amd1:2017: Information Technology – Security Techniques – Random bit generation [52]
- ANSI X9.82: Financial Services – Random Number Generation [77]

  Part 1: Overview

  Part 2: Entropy Sources

  Part 3: Deterministic Random Bit Generators

  Part 4: Random Bit Generator Construction
- NIST SP 800-90A Rev. 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators [131]
- NIST SP 800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications [123]

Most modern mainstream crypto libraries now include satisfactory functions for random number generation, however caution is recommended with older libraries. Examples of random number generators are recent versions of the following environments:

Programming environment	Random number source
Unix	/dev/urandom
Java	java.security.SecureRandom
.net	System.Security.Cryptography.RandomNumberGenerator
Python	secrets.randbits
C	libsodium (third-party library)

See the respective environments’ documentation for details on how to use these crypto random generators.

Some applications require random numbers or random bit strings in order to ensure freshness of a cryptogram (e.g. in a challenge-response protocol) or to protect the private key in an elliptic curve signing process. A standard approach for a device to derive such random values is to use a one-way function of an internal device state (at least 100 bits) where the internal device state is updated as a one-way function of a secret seed (at least 100 bits), time-varying data (e.g. a counter) and hardware entropy (if available).

Note that due to concerns regarding the security of the Dual Elliptic Curve Deterministic Random Bit Generator (Dual_EC_DRBG) this method has been officially removed from both updated NIST [131] and ISO/IEC18031:2011 Amd1:2017 [52] standards. Users of this random generator are recommended to switch to one of the other random number generators in these standards.

When using simpler hardware configurations, particularly without a decent built-in source of random generation, it can be challenging to achieve a high-quality production of random numbers. This particularly applies for appliances and Internet of Things (IoT) devices where the entropy can be very limited based on often-used sources of randomness, such as CPU cycles or a clock. In addition, fixed seeds across identical devices will contribute to poorer random number generation. Some alleviation to this challenge can be given by also considering other sources of randomness, such as timing of network packet reception or input from attached sensors.

Caution is also advised in virtualized environments. The quality of the randomness in these situations can be implementation or hardware dependent and is often very difficult to verify. In particular, the period of time immediately after the startup of a system can be susceptible to generate randomness of poor quality.

6 ANNEX I: Terminology

Definitions

Whenever a definition is copied without alteration from an International Standard, the reference of the standard is given.

Asymmetric algorithm: A cryptographic algorithm employing a public key and a private key. Together these form an asymmetric key set.

Block cipher: A cryptographic algorithm, which maps n-bit plaintext blocks to n-bit ciphertext blocks. “n” is called the block length or block size (both terms are used here).

Confidentiality: The property that information is not made available or disclosed to unauthorised individuals, entities or processes. (ISO 7498-2 [1])

Cryptography: The discipline, which embodies principles, means, and methods for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorised use. (ISO 7498-2 [1])

Cryptoperiod: See key cryptoperiod.

Data integrity: The property that data has not been altered or destroyed in an unauthorised manner. (ISO 7498-2 [1])

Data origin authentication: The corroboration that the source of data received is as claimed. (ISO 7498-2 [1])

Decipherment: The reversal of a corresponding reversible encipherment. (ISO 7498-2 [1])

Decryption: See decipherment. (ISO 7498-2 [1])

Digital signature: Data appended to, or a cryptographic transformation (see cryptography) of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient. (ISO 7498-2 [1])

Encipherment: The cryptographic transformation of data (see cryptography) to produce ciphertext. (ISO 7498-2 [1])

Ephemeral: A cryptographic key that is generated for each execution of a cryptographic process (e.g., key establishment) and that meets other requirements of the key type (e.g., unique to each message or session). (NIST [129])

Encryption: See encipherment. (ISO 7498-2 [1])

Hash function: A (mathematical) function, which maps values from a large (possibly very large) domain into a smaller range. A ‘good’ hash function is such that the results of applying the function to a (large) set of values in the domain will be evenly distributed (and apparently at random) over the range. (ISO 9594-8 [18])

Key: A sequence of symbols that controls the operations of encipherment and decipherment. (ISO 7498-2 [1])

Key cryptoperiod: The time period over which a key is valid for use by legitimate parties.

Key encapsulation: a class of encryption techniques designed to secure symmetric cryptographic key material for transmission using asymmetric (public-key) algorithms.

Key establishment: A process whereby a shared secret key becomes available to two or more parties, for subsequent cryptographic use.

Message Authentication Code: A data item derived from a message using cryptographic techniques to provide message integrity and authenticity.

Private key: (In a public key cryptosystem) that key of a ’user's key pair which is known only by that user. (ISO 9594-8 [18])

Public key: (In a public key cryptosystem) that key of a ’user's key pair which is publicly known. (ISO 9594-8 [18])

Repudiation: Denial by one of the entities involved in a communication of having participated in all or part of the communication. (ISO 7498-2 [1])

Secret key: A key used with symmetric cryptographic techniques and usable only by a set of specified entities. (ISO/IEC 11770-1 [36])

Signcryption: A public-key primitive that simultaneously performs the functions of both digital signature and encryption.

Stream cipher: A symmetric encryption system with the property that the encryption algorithm involves combining a sequence of plaintext symbols with a sequence of keystream symbols one symbol at a time, using an invertible function (ISO/IEC 18033-1 [54]).

Symmetric algorithm: A cryptographic algorithm employing the same value of key for both enciphering and deciphering or for both authentication and validation.

Unkeyed: A cryptographic algorithm that only uses a message as parameter, in other words, no cryptographic key is involved.

Abbreviations

2TDES	Two-key Triple DES
3TDES	Three-key Triple DES
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
ATM	Automated Teller Machine
CA	Certification Authority
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CRL	Certificate Revocation List
CRT	Chinese Remainder Theorem
CV	Control Vector
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
DH	Diffie-Hellman
DLT	Distributed Ledger Technology
DSA	Digital Signature Algorithm

DSS	Digital Signature Standard
DUKPT	Derived Unique Key Per Transaction
ECB	Electronic Code Book
ECBS	European Committee for Banking Standards
ECC	Elliptic Curve Cryptosystem
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
EESSI	European Electronic Signature Standardisation Initiative
EMV	Europay MasterCard Visa
EPC	European Payments Council
EtM	Encrypt then MAC
ETSI	European Telecommunication Standards Institute
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
HMAC	Keyed-Hash Message Authentication Code
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IFES	Integer Factorization Encryption Scheme
ISO	International Organisation for Standardisation
ITSEC	Information Technology Security Evaluation Criteria
IV	Initialisation Vector
KDF	Key Derivation Function
KEK	Key Encrypting Key
MAC	Message Authentication Code
MDn	Message Digest n
MQV	Menezes Qu Vanstone
NESSIE	New European Schemes for Signatures, Integrity, and Encryption
NFT	Non-Fungible Token
NIST	National Institute of Standards and Technology
OAEP	Optimal Asymmetric Encryption Padding
OCB	Offset Codebook Mode
OCSP	On-line Certificate Status Protocol

OFB	Output Feedback
PGP	Pretty Good Privacy
PIN	Personal Identification Number
PKCS	Public Key Cryptography Standards
PKIX	Internet X.509 Public Key Infrastructure
PQC	Post Quantum Cryptography
PSS	Probabilistic Signature Scheme
PSSG	Payment Security Support Group
RFC	Request For Comments
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest Shamir Adleman
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TC	Technical Committee
TDES	Triple Data Encryption Standard
TLS	Transport Layer Security
TRSM	Tamper-Resistant Security Module
TTP	Trusted Third Party
XML	Extensible Markup Language

Table 5: Abbreviations

7 ANNEX II: Bibliography

7.1 ISO Standards

ISO standards are available from ISO at http://www.iso.org, or from the National Standardisation bodies (such as AFNOR, BSI, DIN,...).

[1] ISO 7498-2, "Information processing systems - Open Systems Interconnection - Basic reference model - Part 2: Security architecture" (equivalent to ITU-T Rec X 800)

Technical Committee TC 68 Financial Services, Sub-Committee SC2 Security

[2] ISO 9564-2, " Banking and related financial services - Personal Identification Number management and security - Part 2: Approved algorithms for PIN encipherment"

[3] ISO TR 14742, "Banking and related financial services - Recommendations on cryptographic algorithms and their use"

[4] ISO 16609, "Banking – Requirements for message authentication using symmetric techniques"

[5] ISO TR 19038"Banking and related financial services - Triple DEA - Modes of Operation - Implementation Guidelines"

[6] ISO 11568, "Financial services - Key management (retail)"

[7] ISO 13491-1, "Banking and related financial services - Secure cryptographic devices (retail) - Part 1: Concepts, requirements and evaluation methods"

[8] ISO 13491-2, "Banking and related financial services - Secure cryptographic devices (retail) - Part 2: Security compliance checklists for devices used in financial transactions"

[9] ISO 13492, "Banking and related financial services - Key management related data element - Application and usage of ISO 8583 data elements 53 and 96"

[10] ISO 20038, "Banking and related financial services – Key wrap using AES".

[11] ISO 21188, "Public key infrastructure for financial services -- Practices and policy framework"

Joint Technical Committee ISO/IEC JTC 1 Information Technology

[12] ISO/IEC 10181-1, "Information technology - Open Systems Interconnection - Security frameworks for open systems - Part 1: Overview" (equivalent to ITU-T Rec X 810)

[13] ISO/IEC 10181-2, "Information technology - Open Systems Interconnection - Security frameworks for open systems - Part 2: Authentication framework" (equivalent to ITU-T Rec X 811)

[14] ISO/IEC 10181-3, "Information technology - Open Systems Interconnection - Security frameworks for open systems - Part 3: Access control framework"

[15] ISO/IEC 10181-4, "Information technology - Open Systems Interconnection - Security frameworks for open systems - Part 4: Non-repudiation framework"

[16] ISO/IEC 10181-5, "Information technology - Open Systems Interconnection - Security frameworks for open systems - Part 5: Confidentiality framework"

[17] ISO/IEC 10181-6, "Information technology - Open Systems Interconnection - Security frameworks for open systems - Part 6: Integrity framework"

Joint Technical Committee ISO/IEC JTC 1 Information Technology, Sub-Committee SC 6, Telecommunications and information exchange between systems

[18] ISO/IEC 9594-8, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks" (equivalent to ITU-T Recommendation X 509)

Joint Technical Committee ISO/IEC JTC 1 Information Technology, Sub-Committee SC 27, IT Security Techniques

[19] ISO/IEC 10116, "Information processing - Modes of operation for an n-bit block cipher algorithm"

[20] ISO/IEC 9796-1, "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 1: Mechanisms using redundancy", replaced by 14888

[21] ISO/IEC 9796-2, "Information technology - Security techniques - Digital signature scheme giving message recovery - Part 2: Integer factorisation based mechanisms"

[22] ISO/IEC 9796-3, "Information technology -- Security techniques -- Digital signature schemes giving message recovery -- Part 3: Discrete logarithm based mechanisms"

[23] ISO/IEC 9797-1, "Information technology - Security techniques – Message Authentication Codes (MACs) Part 1: Mechanisms using a block cipher"

[24] ISO/IEC 9797-2, "Information technology - Security techniques – Message Authentication Codes (MACs), Part 2: Mechanisms using a dedicated hash function"

[25] ISO/IEC 9797-3, "Information technology - Security techniques – Message Authentication Codes (MACs), Part 3: Mechanisms using a universal hash function"

[26] ISO/IEC 9798-1"Information technology -- Security techniques -- Entity authentication Part 1: General"

[27] ISO/IEC 9798-2"Information technology -- Security techniques -- Entity authentication Part 2: Mechanisms using symmetric encipherment algorithms"

[28] ISO/IEC 9798-3"Information technology -- Security techniques -- Entity authentication Part 3: Mechanisms using digital signature techniques",

[29] ISO/IEC 9798-4"Information technology -- Security techniques -- Entity authentication Part 4: Mechanisms using a cryptographic check function"

[30] ISO/IEC 9798-5"Information technology -- Security techniques -- Entity authentication Part 5: Mechanisms using zero knowledge techniques"

[31] ISO/IEC 9798-6"Information technology -- Security techniques -- Entity authentication Part 6: Mechanisms using manual data transfer"

[32] ISO/IEC 10118-1, "Information technology - Security techniques – Hash-functions - Part 1: General"

[33] ISO/IEC 10118-2, "Information technology - Security techniques – Hash-functions - Part 2: Hash functions using an n-bit block cipher algorithm"

[34] ISO/IEC 10118-3, "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions"

[35] ISO/IEC 10118-4, "Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic"

[36] ISO/IEC 11770-1, "Information technology - Security techniques - Key management - Part 1: Framework"

[37] ISO/IEC 11770-2, "Information technology - Security techniques - Key management - Part 2: Mechanisms using symmetric techniques"

[38] ISO/IEC 11770-3, "Information technology - Security techniques - Key management - Part 3: Mechanisms using asymmetric techniques"

[39] ISO/IEC 11770-4, "Information technology - Security techniques - Key management - Part 4: Mechanisms based on weak secrets"

[40] ISO/IEC 11770-5, "Information technology - Security techniques - Key management - Part 5: Group key management"

[41] ISO/IEC 11770-6, "Information technology - Security techniques - Key management - Part 6: Key derivation"

[42] ISO/IEC 11770-7, "Information technology - Security techniques - Key management - Part 7: Cross-domain password-based authenticated key exchange"ISO/IEC 14888-1, "Information technology - Security techniques - Digital signatures with appendix - Part 1: General"

[43] ISO/IEC 14888-2, "Information technology - Security techniques - Digital signatures with appendix - Part 2: Integer factorisation based mechanisms"

[44] ISO/IEC 14888-3, "Information technology - Security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms"

[45] ISO/IEC 14888-4, "Information technology - Security techniques - Digital signatures with appendix - Part 4: Stateful hash-based mechanisms"

[46] ISO/IEC 15946-1, "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 1: General"

[47] ISO/IEC 15946-5, "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 5: Elliptic curve generation"

[48] ISO/IEC 18014-1, "Information technology - Security techniques – Time-stamping services - Part 1: Framework"

[49] ISO/IEC 18014-2, "Information technology - Security techniques – Time-stamping services - Part 2: Mechanisms producing independent tokens"

[50] ISO/IEC 18014-3, "Information technology - Security techniques – Time-stamping services - Part 3: Mechanisms producing linked tokens"

[51] ISO/IEC 18014-4, "Information technology - Security techniques – Time-stamping services - Part 4: Traceability of time sources"

[52] ISO/IEC 18031 and ISO/IEC18031:2011 Amd1:2017, "Information technology - Security techniques - Random bit generation"

[53] ISO/IEC 18032, "Information technology - Security techniques - Prime number generation"

[54] ISO/IEC 18033-1, "Information technology - Security techniques - Encryption algorithms - Part 1: General"

[55] ISO/IEC 18033-2, "Information technology - Security techniques - Encryption algorithms - Part 2: Asymmetric ciphers"

[56] ISO/IEC 18033-3, "Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers"

[57] ISO/IEC 18033-4, "Information technology - Security techniques - Encryption algorithms - Part 4: Stream ciphers"

[58] ISO/IEC 18033-5, "Information technology - Security techniques - Encryption algorithms - Part 5: Identity-based cipher"

[59] ISO/IEC 18033-6, "Information technology - Security techniques - Encryption algorithms - Part 6: (partial) Homomorphic encryption"

[60] ISO/IEC 18033-7, “Information technology - Security techniques - Encryption algorithms - Part 7: Tweakable block ciphers”

[61] ISO/IEC 19772, “Information technology - Security techniques - Authenticated encryption”

[62] ISO/IEC 29150, Information technology - Security techniques - Signcryption”

7.2 ETSI Standards

ETSI standards are available from https://www.etsi.org

[63] ETSI TS 101 733, “Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (cAdES)”

[64] ETSI TS 101 903, “XML Advanced Signatures (xAdES)”

[65] ETSI TR 102 176-1, “Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures – Part 1: Hash functions and asymmetric algorithms”

[66] ETSI TR 102 176-2, “Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures – Part 2: Secure channel protocols and algorithms for signature creation devices”

[67] ETSI TS 119 312, “Electronic Signatures and Infrastructures (ESI); Cryptographic suites”

7.3 ANSI Standards

ANSI standards are available from https://www.ansi.org/

ANSI X 9 standards are available from https://x9.org/

[68] ANSI X3.106, “American National Standard for Information Systems- Data Encryption Algorithm - Modes of Operation”, 1983

[69] ANSI X9.24-3, “American National Standard - Retail Financial Services Symmetric Key Management - Part 3: Derived Unique Key Per Transaction”

[70] ANSI X9.24-2, “American National Standard - Financial Industry Standards – Retail Financial Services Symmetric Key Management - Part 2: Using Asymmetric Techniques for the Distribution of Symmetric Keys”

[71] ANSI X9.31, “American National Standard - Financial Industry Standards - Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry”

[72] ANSI X9.42, “American National Standard - Financial Industry Standards - Public Key Cryptography for the Financial Services Industry: Agreement of symmetric keys using Discrete Logarithm Cryptography”

[73] ANSI X9.44, “Key Establishment Using Integer Factorization Cryptography”

[74] ANSI X9.62, “American National Standard - Financial Industry Standards - Public Key Cryptography For The Financial Services Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)”

[75] ANSI X9.63, “American National Standard - Financial Industry Standards - Public Key Cryptography For The Financial Services Industry – Key Agreement and Key Transport Using Elliptic Curve Cryptography”

[76] ANSI X9.80, “American National Standard - Financial Industry Standards - Prime Number Generation Primality Testing, and Primality Certificates”

[77] ANSI X9.82-1, “Random Number Generation, Part 1: Overview and Basic Principles”

[78] ANSI X9.102, “Symmetric Key Cryptography For the Financial Services Industry - Wrapping of Keys and Associated Data”

[79] ASC X9, “Study Group Report Distributed Ledger and Blockchain Technology Study Group”, April 2018.

[80] ASC X9.143-2021 Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms (the standard is formally known as TR31).

7.4 NIST FIPS standards

All FIPS standards are available at https://csrc.nist.gov/publications/fips

[81] FIPS 140-3, “Security requirements for cryptographic modules”, Federal Information Processing Standards Publication, US National Institute of Standards and Technology (supersedes FIPS PUB 140-2).

[82] FIPS 180-4, “Secure Hash Standard (SHS)”, Federal Information Processing Standards Publication, US National Institute of Standards and Technology.

[83] FIPS 186-5, “Digital Signature Standard”, Federal Information Processing Standards Publication, US National Institute of Standards and Technology.

[84] FIPS 197, “Advanced Encryption Standard (AES)”, Federal Information Processing Standards Publication, US National Institute of Standards and Technology.

[85] FIPS 198-1, “The Keyed-hash Message Authentication Code (HMAC)”, Federal Information Processing Standards Publication, US National Institute of Standards and Technology.

[86] FIPS 202, “SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions”, Federal Information Processing Standards Publication, US National Institute of Standards and Technology.

7.5 Internet drafts, “standard” and RFCs

Internet standards and RFCs are available from: http://www.ietf.org.

[87] RFC 1421, “Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures”, Internet Request for Comments 1421; J. Linn.

[88] RFC 1422, “Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management”, Internet Request for Comments 1422; S. Kent.

[89] RFC 1423, “Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers”, Internet Request for Comments 1423; D. Balenson.

[90] RFC 1424, “Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services”, Internet Request for Comments 1424; B. Kaliski.

[91] RFC 1847“Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted”, Internet Request for Comments 1847; J. Galvin, S. Murphy, S. Crocker and N. Freed.

[92] RFC 2104“HMAC: keyed-hashing for Message Authentication”, Internet Request for Comments 2104; H. Krawczyk, M. Bellare, and R. Canetti, updated by RFC 6151.

[93] RFC 3279“Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, Internet Request for Comments 3279; L. Bassham, W. Polk, R. Housley, + updates (see web site).

[94] RFC 3370“Cryptographic Message Syntax (CMS) Algorithms”, Internet Request for Comments 3370; R. Housley, + updates (see web site).

[95] RFC 3447"Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", Internet Request for Comments 3447; J. Jonsson, B. Kaliski.

[96] RFC 3766"Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", Internet Request for Comments 3766; H. Orman, P. Hoffman.

[97] RFC 3874"A 224-bit One-way Hash Function: SHA-224", Internet Request for Comments 3874; R. Housley.

[98] RFC 4055"Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Internet Request for Comments 4055; J. Schaad, B. Kaliski, R. Housley, + Errata (see web site) + RFC 5756 Updates for RSAES-OAEP and RSASSA-PSS Algorithm Parameters.

[99] RFC 4056"Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS", Internet Request for Comments 4056; J. Schaad.

[100] RFC 4301"Security Architecture for the Internet Protocol", Internet Request for Comments 4301; S. Kent, K. Seo, + Errata.

[101] RFC 5246"The Transport Layer Security (TLS) Protocol Version 1.2" (https://datatracker.ietf.org/doc/rfc5246/), updated by RFC 8446.

[102] RFC 7748 "Elliptic curves for security", Internet Request for Comments 7724, A. Langley, M. Hamburg, S. Turner.

[103] RFC 7914"The scrypt Password-Based Key Derivation Function", Internet Request for Comments 7914, C. Persival and S. Josefsson. August 2016.

[104] RFC 8017"Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2", Internet Request for Comments 8017; J. Jonsson, B. Kaliski. November 2016.

[105] RFC 8018"PKCS #5: Password-Based Cryptography Specification, Version 2.1", Internet Request for Comments 8018; K. Moriarty, B. Kaliski, A. Rusch. January 2017.

[106] RFC 8439 "ChaCha20 and Poly1305 for IETF Protocols" (https://tools.ietf.org/html/rfc8439)

[107] RFC 8446 "The Transport Layer Security (TLS) Protocol Version 1.3", - final version (https://datatracker.ietf.org/doc/rfc8446/).

7.6 W3C Recommendations

[108] XML Signature Syntax and Processing (Second Edition), W3C recommendation 2008: http://www.w3.org/TR/2008/PER-xmldsig-core-20080326/.

[109] XML Encryption Syntax and Processing Version 1.1, W3C Candidate Recommendation 2011: https://www.w3.org/TR/2011/CR-xmlenc-core1-20110303/

7.7 PKCS "standard"

PKCS standards are available from http://www.rsa.com

Note that PKCS #1 is now instead maintained as RFC 3447 [95].

[110] PKCS #1, "The Public key cryptography standards - Part 1: RSA encryption standard", version 1.5, 1993. Obsoleted by v2.1.

[111] PKCS #1, "The Public key cryptography standards - Part 1: RSA cryptography standard", version 2.1, 2002. Also RFC 3447 [95]. Obsoleted by v2.2.

[112] PKCS #1, "The Public key cryptography standards - Part 1: RSA cryptography standard", version 2.2, 2016. Also RFC 8017 [104].

[113] PKCS #3, “The Public key cryptography standards - Part 3: Diffie-Hellman key-agreement standard”, version 1.4, 1993.

[114] PKCS #5, Password-Based Cryptography Specification, Version 2.1, 2017. Also RFC 8018 [105].

[115] PKCS #6, “The Public key cryptography standards - Part 6: Extended-Certificate Syntax Standard”, version 1.5, 1993 (superseded by x.509 v3).

[116] PKCS #7, “The Public key cryptography standards - Part 7: Cryptographic message syntax standard”, version 1.5, 1993 (see S/MIME RFC 5652).

[117] PKCS #9, “The Public key cryptography standards - Part 9: Selected Object Classes and Attribute Types”, version 2.0, 2000 (now maintained as RFC 2985).

[118] PKCS #10, “The Public key cryptography standards – Part 10: Cryptographic request syntax standard”, version 1.7, 2000 (now maintained as RFC 2986).

[119] PKCS #11, “The Public key cryptography standards – Part 11: Cryptographic token interface standard”, version 2.30, 2009.

[120] PKCS #13, “The Public key cryptography standards – Part 13: Elliptic Curve Cryptography standard”, under development.

[121] PKCS #15, “The Public key cryptography standards – Part 15: Cryptographic Token Information Format Standard”, version 1.1, 2000 (see ISO/IEC 7816-15).

7.8 EMV specifications

EMV specifications are available from EMVCo: http://www.emvco.com/

[122] EMV 4.4, “Integrated Circuit Card Specifications for Payment Systems: Book 1 Application Independent ICC to Terminal Interface Requirements, Book 2 Security and Key Management, Book 3 Application Specification, Book 4 Cardholder, Attendant and Acquirer Interface Requirements”, Version 4.4, EMVCo, 2022.

7.9 NIST Special Publications

All NIST Special Publications on cryptography are available from http://csrc.nist.gov/groups/ST/:

[123] NIST SP800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications”

[124] NIST SP 800-38A, “Recommendation for Block Cipher Modes of Operation - Methods and Technique”, + Addendum.

[125] NIST SP 800-38B, “Recommendation for block cipher modes of operation: The CMAC mode for authentication”.

[126] NIST SP 800-38F, “Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping”.

[127] NIST SP 800-38G, “Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption”.

[128] NIST SP 800-56c, “Recommendation for Key Derivation through Extraction-then-Expansion”.

[129] NIST SP 800-57 Revision 5, “Recommendation for key management Part 1: General guideline”.

[130] NIST SP 800-67 Revision 2, “Recommendation for Triple Data Encryption Algorithm (TDEA) Block Cipher”.

[131] NIST SP 800-90A Rev. 1, “Recommendations for Random Number Generation Using Deterministic Random Bit Generators”.

[132] NIST SP 800-108 Revision 1, “Recommendation for Key Derivation Using Pseudorandom Functions”.

[133] NIST SP 800-131A Revision 2, “Recommendation for Transitioning the use of Cryptographic Algorithms and Key Length”.

[134] NIST SP 800-132, “Recommendation for Password-Based Key Derivation”.

[135] NIST SP 800-133, “Recommendation for Cryptographic Key Generation”.

[136] NIST SP 800-208,”Recommendation for Stateful Hash-Based Signature Schemes".

7.10 Other references

[137] R. RIVEST, A. SHAMIR and L. ADLEMAN, “A Method for obtaining Digital Signatures and Public Key Cryptosystemn”. Communications of the ACM, Vol.21(2), pp 120-126 (1978).

[138] H. DOBBERTIN, A. BOSSELAERS and B. PRENEEL, “RIPEMD-160: A strengthened version of RIPEMD” In “Fast Software Encryption, Proc. third International Workshop, Cambridge, UK February 21 – 23, 1996, pp. 71-82, D. Gollman editor”, Lecture Notes in Computer Science No. 1039. Springer-Verlag, Berlin 1996.

[139] D.W. DAVIES and W.L. PRICE, “Security for computer networks”, John Wiley & Sons, New York, 2nd edition 1989.

[140] B. SCHNEIER, “Applied cryptography: Protocols, Algorithms and Source code in C”, John Wiley & Sons, New York, 2nd edition, 1996.

[141] A. J. MENEZES, P.C. VAN OORSCHOT and S. A. VANSTONE, “Handbook of applied Cryptography”, CRC Press, Boca Raton 1997, https://cacr.uwaterloo.ca/hac/.

[142] IEEE P1363-2000: Standard Specifications For Public Key Cryptography.

[143] R. ANDERSON and R. NEEDHAM “Robustness principles for public key protocols”, Cambridge University Computer Laboratory, (rja14@cl.cam.ac.uk).

[144] D. BONEH, “Twenty years of attacks on the RSA cryptosystem”, (dabo@cs.stanford.edu).

[145] M. BELLARE and P. ROGAWAY, “The exact security of digital signatures: How to sign with RSA and Rabin”. In: U.M. Maurer (editor), Advances in Cryptology – Eurocrypt ’96, Lecture Notes in Computer Science 1070 (1996), Springer-Verlag, pp.399-416.

[146] M. BELLARE and P. ROGAWAY, “Optimal asymmetric encryption – How to encrypt with RSA”. In: A. De Santis (editor), Advances in Cryptology – Eurocrypt ’94, Lecture Notes in Computer Science 950 (1995), Springer-Verlag, pp.92-111.

[147] E. Biham, “How to Forge DES-Encrypted Messages in 2^{28} Steps”, http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi?1996/CS/CS0884

[148] A. LENSTRA and E. VERHEUL, “Selecting cryptographic key sizes”, 1999.

[149] R. SILVERMAN, “A cost based security analysis of symmetric and asymmetric key lengths”, RSA laboratories bulletin, number 13, April 2000.

[150] J-S. CORON, D. NACCACHE and J.STERN, “On the security of RSA Padding”, Crypto '99, Lecture Notes in Computer Science Volume 1666, pages 1-18, Springer-Verlag, Berlin 1999.

[151] M. BLAZE, W. DIFFIE, R. RIVEST, B. SCHNEIER, T. SHIMOMURA, E. THOMPSON and M. WIENER, “Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security”, January 1996.

[152] J.-S. CORON, M. JOYE, D. NACCACHE and P. PAILLIER, “Universal padding schemes for RSA”. In M. Yung, editor, Advances in Cryptology – Crypto 2002, volume 2442 of Lecture Notes in Computer Science. Springer Verlag, 2002.

[153] B. CANVEL, A. HILTGEN, S. VAUDENAY and M. VUAGNOUX, “Password Interception in a SSL/TLS Channel”. In D. Boneh, editor, Advances in Cryptology - Crypto 2003, volume 2729 of Lecture Notes in Computer Science. Springer-Verlag, 2003.

[154] A. SHAMIR and E. TROMER, “On the Cost of Factoring RSA-102”, Weizmann Institute of Science, https://www.researchgate.net/publication/2930418_On_the_Cost_of_Factoring_RSA-1024.

[155] X. WANG, D. FENG, X. LAI and H. YU, “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD”, https://eprint.iacr.org/2004/199.pdf.

[156] X. WANG, Y.L. YIN and H. YU, “Collision Search Attacks on SHA”, February 13, 2005 https://www.c4i.org/erehwon/shanote.pdf

[157] A. LENSTRA, X. WANG, B. DE WEGER.“Colliding X.509 Certificate”, version 1.0, 1st March 2005, http://eprint.iacr.org/2005/067.

[158] K.G. PATERSON and A. YAU, “Padding Oracle Attacks on the ISO CBC Mode Encryption Standard”. in T. Okamoto (ed.), Proc. CT-RSA04, Lecture Notes in Computer Science Vol. 2964, pp. 305-323, Springer-Verlag, Berlin, 2004.

[159] A. YAU, K.G. PATERSON and C.J. MITCHELL, “Padding oracle attacks on CBC-mode encryption with random and secret iVs”, Proceedings FSE 2005, Springer LNCS.

[160] A. BIRYUKOV, “Some Thoughts on Time-Memory-Data Tradeoffs”, https://eprint.iacr.org/2005/207

[161] ECRYPT, “Recent Collision Attacks on Hash Functions: ECRYPT Position Paper”, Revision 1.1, 17 February 2005 https://web.archive.org/web/20220301204947/https://www.ecrypt.eu.org/.

[162] ECRYPT CSA, “Algorithms, Key Size and Protocols Report (2018”, Editor Nigel Smart, Revision 1.0, 28th February 2018 https://web.archive.org/web/20220301204947/https://www.ecrypt.eu.org/.

[163] S. CONTINI and Y.L. YIN, “Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions”, ASIACRYPT 2006, pp. 37-53.

[164] X. WANG, Y.L. YIN and H. YU, “Finding Collisions in the full SHA-1”, in V Shoup ed. Advances in Cryptology - CRYPTO 2005 Lecture Notes in Computer Science Vol. 3621, pp 17-36, Springer-Verlag, Berlin, 2005.

[165] D. GLIGOROSKI, S. ANDOVA and S.J. KNAPSKOG, “On the Importance of the Key Separation Principle for Different Modes of Operation”, proceedings of ISPEC 2008, Lecture Notes in Computer Science Vol. 4991, pp 404-418, Springer-Verlag, Berlin, 2008.

[166] P. ROGAWAY and T. SHRIMPTON, “The SIV Mode of Operation for Deterministic Authenticated-Encryption (Key Wrap) and Misuse-Resistant Nonce-Based Authenticated-Encryption”, http://www.cs.ucdavis.edu/~rogaway/papers/.

[167] A. BOGDANOV, M. KHOVRATOVICH and C. RECHBERGER, “Biclique Cryptanalysis of the Full AES”, http://eprint.iacr.org/2011/449.

[168] J. W. BOS, M. E. KAIHARA, T. KLEINJUNG, A. K. LENSTRA and P. L. MONTGOMERY, “On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography”, version 1, July 14, 2009 https://document,s.epfl.ch/users/l/le/lenstra/public/papers/ecdl.pdf.

[169] G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, “The Keccak Sponge Family Function”, June 2010 https://keccak.team/obsolete/Keccak-main-2.1.pdf.

[170] G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, “The Keccak SHA-3 submission”, January 14 2011 https://keccak.team/files/Keccak-submission-3.pdf.

[171] G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, “The Keccak Reference”, January 14 2011 https://keccak.team/files/Keccak-reference-3.0.pdf.

[172] M. STEVENS, “Attacks on Hash Functions and applications”, Ph. D. Thesis, http://www.cwi.nl/system/files/PhD-Thesis-Marc-Stevens-Attacks-on-Hash-Functions-and-Applications.pdf.

[173] A.K. LENSTRA, J.P. HUGHES, M. AUGIER, J.W. BOS, T. KLEINJUNG and C. WACHTER, “Ron was wrong, Whit is right”, http://eprint.iacr.org/2012/064.pdf.

[174] ENISA, “Algorithms, Key Sizes and Parameters Report, 2013 recommendations”, version 1.0, October 2014, www.enisa.europa.eu.

[175] NIST, ITL Bulletin, “Additional secure hash algorithms offer new opportunities for data protection”, Sept 2015 https://csrc.nist.gov/csrc/media/publications/shared/documents/itl-bulletin/itlbul2015-09.pdf.

[176] NSA, “Cryptography Today”, [192] NSA, “Cryptography Today”, https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

[177] N. KOBLITZ and A.J. MENEZES, “A riddle wrapped in an enigma”, https://eprint.iacr.org/2015/1018.pdf.

[178] M. STEVENS, https://marc-stevens.nl/research/papers/KPS_freestart80.pdf

[179] C. MITCHELL, On the security of 2-key Triple DES https://www.chrismitchell.net/Papers/otso2t2.pdf

[180] P. PAILLIER, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Advances in Cryptology, EUROCRYPT 1999, LNCS 1592, Springer-Verlag, 1999, pp. 223-238.

[181] A. JOUX, A new index calculus algorithm with complexity *L(1/4+o(1))* in very small characteristic, https://eprint.iacr.org/2013/095.pdf.

[182] K BHARGAVAN, G. LEURENT, On the Practical (In-)Security of 64-bit Block Ciphers — Collision Attacks on HTTP over TLS and OpenVPN, ACM CCS 2016.

[183] B. PRENEEL and P. C. VAN OORSCHOT, A key recovery attack on the ANSI X9.19 retail MAC. Electronics Letters, 32:1568-1569, 1996.

[184] C. COSTELLO and P. LONGA, FourQ: “four-dimensional decompositions on a Q-curve over the Mersenne prime, Advances in Cryptology”, ASIACRYPT 2015.

[185] M. Stevens, E Bursztein, P. Karpman, A. Albertini and Y. Markov, “The first collision for full SHA-1”, https://shattered.io/static/shattered.pdf.

[186] S.NAKAMOTO, “Bitcoin: A peer-to-peer electronic cash system”, https://bitcoin.org/bitcoin.pdf, 2008.

[187] I.GIECHASKIEL, C. CREMERS, K.B. RASMUSSEN, “On bitcoin security in the presence of broken crypto primitives”, 2016.

[188] EUROPEAN UNION AGENCY FOR NETWORK AND INFORMATION SECURITY (ENISA) “Distributed Ledger Technology & Cybersecurity”, December 2016, https://www.enisa.europa.eu/publications/blockchain-security.

[189] D.ROMANO, G.SCHMID, “Beyond Bitcoin: A critical look at blockchain-based systems”, September 2017.

[190] UK GOVERNMENT OFFICE FOR SCIENCE, “Distributed ledger technology: beyond blockchain”, January 2016.

[191] J.A GARAY, A. KIAYIAS, G. PANAGIOTAKOS, “Proofs of Work for blockchain protocols”, 2017.

[192] M. NAMEC, M. SYS, P. SVENDA, D. KLUSEC, V. MATYAS, “The Return of Coppersmith’s Attack: Practical Factorization of Widely Used RSA Moduli”, ACM CCS 2017.

[193] Gaetan Laurence and Thomas Peyrin“SHA-1 is a Shambles First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust”, January 2020, https://eprint.iacr.org/2020/014.pdf

[194] NIST Blockchain Technology Overview, October 2018, https://www.nist.gov/publications/blockchain-technology-overview.

[195] M. BAUDET, A. CHING, A. CHURSIN, G. DANEZIS, F. GARILLOT, Z. LI, D. MALKHI, O. NAOR, D. PERELMAN, A. SONNINO “State Machine Replication in the Libra Blockchain”, May 2020

[196] BSI TR-02102-1:“Cryptographic Mechanisms: Recommendations and Key Length” Version: 2020-1: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.html

[197] SoK: Computer-Aided Cryptography, 2019, https://eprint.iacr.org/2019/1393.pdf

[198] “The unbearable lightness of PIN cracking” (conference paper): https://www.researchgate.net/publication/220797019_The_Unbearable_Lightness_of_PIN_Cracking

[199] OWASP: “Certificate and Public Key Pinning” Certificate and Public Key Pinning | OWASP Foundation

7.11 Quantum computing

[200] S. FLUHRER “Reassessing ’Grover's Algorithm”, 2017, https://eprint.iacr.org/2017/811

[201] M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent and J. Schanck, “Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3”, https://eprint.iacr.org/2016/992.pdf.

[202] Status of quantum computer development, v2.0, 2023: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Quantencomputer/Entwicklungstand_QC_V_2_0.html.

[203] US “National Quantum Initiative”: https://www.quantum.gov/

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

[205] BSI recommendations paper, August 2020: "Migration zu Post-Quanten-Kryptografie": https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Krypto/Post-Quanten-Kryptografie.pdf

[206] BSI, May 2022 “Quantum-safe cryptography – fundamentals, current developments and recommendations”,

https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.html

[207] Quantum Computing and the Financial Services Industry https://committee.iso.org/sites/tc68/home/articles/content-left-area/articles/quantum-computing-in-the-financi.html

[208] Guidance by NCSC, UK, on "Next steps in preparing for post-quantum cryptography", 14 August 2024, https://www.ncsc.gov.uk/whitepaper/next-steps-preparing-for-post-quantum-cryptography#section_6

[209] EMVCo security position statement on quantum computing and EMV chip cryptography, September 2024, https://www.emvco.com/resources/security-position-statement-quantum-computing-and-emv-chip-cryptography-2/