European
Payments Council

[X] Public – [ ] Internal Use – [ ] Confidential – [ ] Strictest Confidence
Distribution: Publicly available

GUIDELINES ON CRYPTOGRAPHIC ALGORITHMS USAGE AND KEY MANAGEMENT

Abstract	This document defines guidelines on cryptographic algorithms usage and key management.
Document Reference	EPC342-08
Issue	**Version 6.0**
Date of Issue	13 December 2016
Reason for Issue	Maintenance of document
Reviewed by	EPC
Produced by	EPC
Authorised by	EPC
Circulation	General Public

Document History

This document was first produced by ECBS as TR 406, with its latest ECBS version published in September 2005. The document has been handed over to the EPC which is responsible for its yearly maintenance.

TABLE OF CONTENTS

Management Summary

The purpose of this document is to provide guidance to the European payments industry, in the field of cryptographic algorithms and related key management issues.

It has been written mainly for payment service providers: security officers, risks managers, system engineers and systems designers. Although its reading does not assume expertise in cryptology, it contains some sections which require a basic mathematical background.

There is constant progress in the science of cryptology, either to create new algorithms or to break existing ones, such that any recommendation on cryptographic algorithms and key management is liable to become obsolete before it is published. 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, and some are subject to one or another type of attack, depending on how and for what they are used.

Consequently, the choice of a cryptographic technique to protect data should always be the result of a risk assessment process. This process should consider not only the potential loss in case the cryptographic technique fails to prevent an attack, but also the operational conditions that may allow some kinds of attacks and prevent others. For example, algorithms, which are subject to known plaintext/cipher text attacks when used in a certain way, may be strong enough if used in another way that does not allow the attacker to access an adequate sample of plaintexts and matching cipher texts.

As most algorithms are subject to attacks that will succeed given sufficient time, this risk assessment should consider the period during which data must be protected. As an example, an algorithm that may be broken within 24 hours would be safe enough to protect data for 10 minutes, but not for a week.

Using a cryptographic algorithm, which is considered as weak by the specialists may be a bad political decision as it may alter the reputation of a financial institution, although it may have no operational consequences for many systems.

A system using cryptography should always be designed with a possible algorithm migration in mind. Nevertheless, the cost and difficulty of migrating from one algorithm to another (or of changing the size of the keys) should not be underestimated.

This document specifies a number of recommendations on crypto algorithms, digital signatures, security protocols and cryptographic transformations which are summarised in Table 1 (section 1.3) and for which further detailed background information may be found in sections 2 to 4.

It further provides guidance on implementation best practices which are summarised in Table 2 (section 1.4) and for which further detailed background information may be found in sections 2 to 4.

Finally, when implementing and integrating cryptography into protocols or schemes, algorithms, key sizes and randomness of key material are essential, but other security aspects beyond the scope of this document become equally important such as side-channel countermeasures and protocol layer

interdependency checking.

With the publication of this new version of the document, the structure and layout have been updated to improve its readability. Newsworthy developments in cryptography since the publication of version 5.0 have also been integrated.

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 professional or national regulation 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.

1.2 Document structure

Section 1 specifies the scope of this document, contains the list of "Recommendations" and "Implementation best practices and describes the structure 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. 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.
Rec 2 **Symmetric algorithms**	- Single DES (56 bits) should be considered broken. - Using 2TDES may still be acceptable under specific conditions (3.1.4); plans should be made to migrate to larger block sizes. - Using 3TDES is still acceptable in use cases where there is no concern regarding reduced block sizes. - AES is the recommended standard for new systems. Further details are provided 3.1.4.
Rec 3 **Public Key** **algorithms**	- No longer use RSA keys of 768 bits and ECC keys of 130 bits, or less; - Use 1024-bit RSA keys and 160-bit ECC keys only for short term protection (e.g. ephemeral authentication for single devices); - Only use 1024-bit RSA keys and 160-bit ECC keys for decrypting previously-encrypted data or verifying cryptographic protection (e.g. signature verification) rather than for applying cryptographic protection (signature generation or data encryption); - Use at least 2048-bit RSA or 224-bit ECC for medium term protection (see [176]); - For considerations regarding low exponent RSA, section 3.1.3 should be consulted; - New systems should be designed with the agility to introduce PQC algorithms as and when they are standardised (see 3.1.5). Further details are provided in 3.1.4.
Post quantum computing	
Rec 4	In view of the deployment of viable quantum computing systems, if long-term confidentiality (several decades) is needed:

	- Do not use ciphers with small data block sizes (such as any form of TDES) - Use sufficiently long keys (at least 256 bits) - Use a sufficiently long hash function (at least 512 bits). Further details are provided in 3.1.5.
Digital signatures	
Rec 5	Except for systems where computational or data storage capacities are restricted, or where a strategic long-term migration is needed, there is no strong reason to move from RSA to ECC-based signature schemes.
Security protocols	
Rec 6	In designing a security protocol the following should be identified: - the security service provided, - the entities involved, - the security verification values (MACs, hashes, digital signatures, …) that will need to be generated and verified, - the algorithms and their modes of operation, - the key material to be generated, - the random number generators to be used (e.g. key generation, challenges, padding), - the key establishment material (this could be key names, encrypted session keys, public key certificates or certificate references, initialisation vectors if any, …), - unambiguous definition of the data used as input in the cryptographic mechanisms, - the formatting techniques used in the signature generation/verification process, - the transformation techniques used to represent binary data (known as filtering), if any.
Confidentiality and integrity protection	
Rec 7	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.
Key management	
Rec 8	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.
Rec 9	Symmetric keys should be dedicated to one usage (e.g. encryption or MAC computation, but not both).
Rec 10	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.
Rec 11	Keys should be generated inside a TRSM and private keys should never exist in clear text outside a TRSM.
Rec 12	Where appropriate public keys shall be distributed such that their integrity and the binding with the owner are preserved (e.g., by using certificates).
Rec 13	Usage of X 509, version 3, format certificates is recommended.
Rec 14	When verifying a digital signature, users should check that the certificate status (either valid, expired or revoked) when the signature was generated does not make it invalid. When a certification path is involved, this verification should be performed for every certificate of the path up to the root certificate. Efficient verification may require that the date and time when the signature was produced is unambiguously defined.
Rec 15	Whenever possible, trusted time sources should be used, in order to allow reliable verification of certificate validity.
Rec 16	An asymmetric key pair should be dedicated to one usage, for instance: one of entity authentication, non-repudiation of data, symmetric keys encryption.
Rec 17	Payment service providers shall not use key escrow, but key recovery is part of their due diligence and business continuity obligations.

Table 1: Recommendations

1.4 Implementation best practices

In addition to the recommendations listed in the previous section, the following best practices with respect to implementations of cryptographic techniques need to be emphasised.

Data confidentiality	
BP 1	For assuring confidentiality of data, symmetric block cipher algorithms are still the best choice. They are subject to attacks, made easier if the secret key is short. Any legacy system using 56-bit keys (i.e. single DES) should now have been replaced. 2TDES is still appropriate for legacy applications under certain circumstances (e.g. limited key use or data protection time). AES using 128-bit keys is today's recommended standard for new applications. Additionally, the use of three-key Triple DES is acceptable. In all cases, the secret key crypto period should be as short as possible and therefore usage of non-reusable session keys should always be preferred. See also Recommendations 2 and 4.
Data integrity	
BP 2	For assuring data integrity, although the SHA-1 hash function may still be used in certain legacy use cases[1], migration towards SHA-224, SHA-256, SHA-384 and SHA-512, (collectively known as the SHA-2 family), Whirlpool or to SHA-3 should be used. Alternatively, a symmetric algorithm MAC computed with a secret key may be used. And this may also provide data origin authentication. When a symmetric algorithm MAC is used, similar considerations on the length of the key as for confidentiality apply: use of TDES or AES should be preferred.
Data integrity and origin authentication	
BP 3	*For assuring data integrity and data origin authentication* (which are the bases to provide non repudiation of origin), an asymmetric algorithm and a hash function based digital signature is recommended. SHA-1 may still be used for legacy applications, but more recent hash functions such as SHA-2, Whirlpool or SHA-3 are preferred. Legacy systems that are architecturally restricted to using RSA moduli less than, or equal to 1024 should be rapidly phased out, with moduli longer than 1024 bits recommended for any new applications. When protection is needed for longer than 3 years then applications will need to support RSA digital signatures with 1536-bit or even 2048-bit moduli. Elliptic Curve DSA algorithms are now being adopted on the grounds of smaller key sizes and signing efficiency, but their use in Europe is not yet very common and may be subject to various patents.
Key management	
BP 4	Key management for asymmetric algorithms requires distribution of authenticated public keys and secured storage of private keys. The best solution to the first requirement is to use certificates, however this leads to the problems of certificate management and of establishment of Certification Authorities. For secured storage of private keys, different techniques are possible, the use of smart cards with RSA capability being one already widely adopted.

---

[1] E.g., integrity of data may be assured by protecting the hash of the data.

BP 5	Keys (either symmetric or asymmetric key pairs) should be dedicated to one purpose only. For instance, separate symmetric keys should be used for MAC computation and for confidentiality, and an asymmetric key pair should be dedicated to digital signature or for key encryption. Amongst the benefits of such a policy are: design flexibility, facilitation of exploiting audit trails, simplification of archiving, prevention of many attacks.

Table 2: Implementation best practices

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 we propose 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 a main part of this document, but discussed briefly where appropriate.
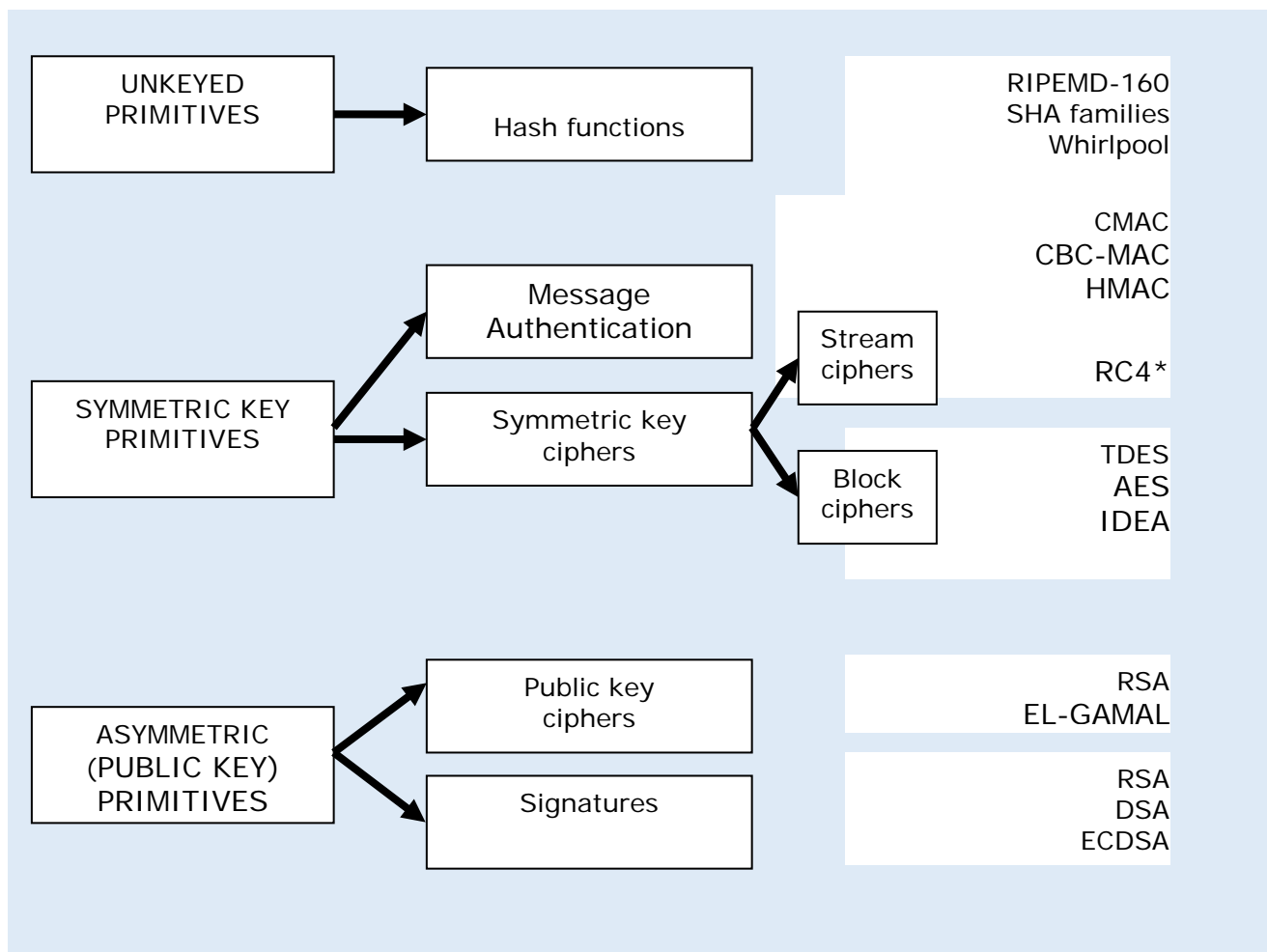
Figure 1: A technical taxonomy of cryptographic primitives and mechanisms

(*) Note that RC4 is no longer a recommended algorithm

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 and RIPEMD [37] and [96].There are also hash functions based on a modified block cipher, like Whirlpool [37] or on generic block ciphers [36], or functions based on modular arithmetic [38]. 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 [34].

A hash function is normally employed as a component of a data integrity mechanism[2]. It is generally a central component of a digital signature scheme.

Examples: SHA-1/-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, IDEA)

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

2.1.2 Elementary Constructions

In this section we review 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 in order 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 [104], ANSI X9.31 [72], ISO/IEC 18033-2 [57]), EL-GAMAL, ECIES (IEEE P1363 [132], ISO/IEC 18033-2 [57]), Paillier ([182]).

Typically a public key is used for encrypting an ephemeral symmetric key which is then used for encrypting the message. The recipient uses its private key to decrypt the ephemeral 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 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 [27]), CMAC (ISO/IEC 9797-1, mechanism 5 [26]) and CBC-MAC (ISO/IEC 9797-1, mechanism 1 [26])

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 [104], ISO/IEC 9796-2 [24]), DSA (FIPS PUB 186-4 [82], ISO/IEC 14888-3 [47]), ECDSA and similar schemes (ISO/IEC 14888-3) [47]

2.2 Typical Usage

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

- Confidentiality protection
    o Data confidentiality,
    o Key encapsulation,
    o Key establishment.
- Integrity protection
    o Data origin authentication,
    o Entity authentication,
    o 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	Yes (one to one)	No (1)	No	No
Key encapsulation	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 3: Matching of techniques and security functionalities

Notes:

(1) Usage is possible, 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. MACs offer better performance, but signatures allow for authentication towards multiple recipients (one-to-many), as the public verification key can be freely distributed.

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 Legal or Commercial Status

Primitives and elementary constructions may also be viewed according to their legal or commercial status, such as:

- De jure or de facto standard,
- Government/military specifications,
- Proprietary algorithm.

Some algorithms have been standardised by ISO (usually by ISO/IEC JTC1/SC27 or ISO TC68/SC2) or by national bodies such as ANSI or NIST and are de jure standards. Other algorithms have never been standardised, but are recognised by

the industry as de facto standards. For instance, RSA PKCS, ANSI X9.31 [72] and IETF PKIX (information at http://www.ietf.org) specifications may be considered as de facto 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 usage 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. An example of a proprietary algorithm which has been widely adopted is RC4 that was developed and made public by RSA Data Security, Inc. and has been used in SSL and S/MIME [90].

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.

---

3 Algorithm Related Design Issues

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 [81]) 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 [144] (which produces a 160-bit hash result) they are now also standardised in ISO/IEC 10118-3 [37] (the ISO standard does not include SHA-224). 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 [81], [85], [177]).

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 a few collision search attacks on SHA-1 [158] (and more recently by Marc Stevens on the SHA-1 compression function). However, there is a new method developed by Marc Stevens [174] that allows to discover an attack attempt, given a message. The false positive rate of this method is negligible. If it is necessary for some reason (e.g., backward compatibility) to still support SHA-1, it is recommended to apply such a method.

Recently, a "freestart collision" for SHA-1 has been computed (see [180]) that gives the last blocks of a possible collision for SHA-1. As a result of this, many parties are now migrating away from SHA-1. Although there is still no known pair of different plaintexts that yield the same SHA-1 value, it is strongly recommended to move towards longer hashes.

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, [122] disallows SHA-1 for digital signature generation.

3.1.1.2   Modification Detection Code

A block cipher may be used as a kind of hash function to compute a Modification Detection Code. In this usage, the key required by the algorithm is no longer secret and may be replaced by data that is being protected. This technique is described in ISO/IEC 10118-2 [36] where a 128-bit hash result is produced. The DES algorithm may be used to compute such an MDC (known as DES in MDC mode). Since 2005, 128-bit hash results are considered to be too short for general usage, so ISO/IEC 10118-2 [36] should not be used with 64-bit block ciphers.

3.1.1.3   Length recommendations

In the long term, hash functions with less than 256 bits output length should not be used. We recommend the use of SHA-2 or Whirlpool and a migration to the SHA-3 standard in due course (see Table 4). For legacy systems SHA-1 may[3] still be acceptable, but, unless the lifetime of the system is limited, plans should be made to 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 [6]) on TDES modes of operation was published in 2005. This technical report specifies modes of operation consistent with ISO/IEC 10116 [22] and provides implementation guidelines specifically for TDES and the financial industry.

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 [83]), 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 [59].

In [115] NIST has defined modes of operation for AES consistent with ISO/IEC 10116 [22].

Further information on AES may be found at:

---

[3] For example, if the collision-resistant property of SHA-1 is not relied on.

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 [1]) 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 8-byte and 16-byte PIN blocks.

3.1.2.4   **Algorithm and Key length recommendations**

EPC recommendation 2

- *Single DES (56 bits) should be considered broken.*

- *Using 2TDES may still be acceptable under specific conditions (see 3.1.4); plans should be made to migrate to larger block sizes.*

- *Using 3TDES is still acceptable in use cases where there is no concern regarding reduced block sizes.*

- *AES is the recommended standard for new systems.*

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 3.1.4 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 [185] and [184].

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[4] (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.

Reblocking problem
It may be necessary to sign a message (with the sender's private RSA key) and then encrypt the resulting signature (with the receiver's public RSA key). One must be concerned about the relative sizes of the moduli involved when implementing this procedure. If the sender's modulus is greater than the receiver's there is a chance that the receiver cannot recover the signature.

There are different ways of solving this issue some of which are discussed in [130], for instance.

When requiring authenticity and confidentiality there are pros and cons as to whether messages should first be signed and then encrypted or vice versa. See section 3.4.3 for a discussion on this.

Random padding and redundancy
The basic transformations that implement RSA decryption and signature generation (complemented by encryption and verification, respectively) are essentially the same but have diametrically different requirements, the first for confidentiality and the second for authenticity and integrity. This leads to different processing requirements in order to defend against one of the most basic types of attack, namely guessing (followed by application of the public RSA function to corroborate the guess).

With confidentiality,[5] the designer must ensure that the chances of an attacker guessing the plaintext are sufficiently small. So for instance with PIN encryption, padding with an unpredictable value must be introduced. This is to prevent an attacker who has an encrypted PIN-message from being able to exhaustively encrypt every possible PIN-message until the correct one is found.

With authenticity, the designer must similarly ensure that the chances of an attacker guessing a useful signature are sufficiently small. This effectively requires that a specified function introducing redundancy be applied to messages before

---

[4] 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.

[5] This point is not RSA specific and may be pertinent for any encryption algorithm.

applying the RSA private function (thereby reducing the probability that the application of the RSA public function to a signature-guess will result in a valid message). Such a function may be implemented by specifying fixed message headers and trailers and also by padding the message with data derived from the actual message (e.g. a modified copy of the message or a hash of the message).

A function introducing redundancy will combine message information with the hash value in order to create a specially structured field. Doing so may prevent some published attacks (for instance some attacks based on the multiplicative property of RSA), and allow the verifier to check the structure of the signed data before verifying the hash itself. Thus it will be possible to distinguish erroneous signatures:

- produced with a wrong private key (the redundancy is not respected)
- produced with the good private key over corrupted data or on a false hash value (the redundancy is correct, but the hash is wrong)

Current Standards address this requirement.

The mathematics and usage of RSA can be subject to far subtler attacks and the straightforward use of random padding and redundancy functions may not always be sufficient. [133], [134] and [135] provide useful starting points for further reading on this subject, [147] and [129] may also be consulted about RSA padding.

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.

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 quicker.

Some attacks on RSA implementations using CRT have been published. These attacks involve the 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 [135], [140], and [141]). For example the use of exponent 3 immediately reveals the plaintext if used for message encryption to at least three recipients (see [135]).

It is also possible to choose small private exponents in case the signature generation or decryption device is resource constrained. However, there are attacks against this approach (see for example [135]) and so normally it is recommended instead to use more powerful devices that enable the completion of signature generation and encryption operations using private exponents that are the same length as the modulus.

3.1.3.2 Elliptic Curve Cryptosystem (ECC) specific issues

Domain Definition
ECC keys and certificates are claimed to be 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. If the parameters are not shared, then the *domain* choice becomes part of the public key. Typically the domain parameters are some three times the size of the basic public key.

Domain Sharing
If the domain parameters *a*, *b*, *q*, *G* & *n* are all shared, then for each user the public key consists only of *Q*. This results in small public keys and the possibility of optimisation of hardware and software to suite the domain parameters.

If only the base field *q* is shared, the result is that the public key size increases significantly (threefold or more), but there may be an advantage from a security point of view.

Finally, if no parameters are shared, but become part of each public key, then there is complete freedom of choice of the parameters to the user, but a resulting further increase in the public key size, plus any implementation optimisations will have to be customised for each domain.

It should be noted that sharing domain parameters could increase the prospect of an attack since the rewards could be perceived as being larger and it may become possible to design attacks that exploit specific properties of the shared domain parameters.

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 [186] and curve25519 specified in IETF RFC 7748 [101] (the authors of which cite advantages in performance and security compared to NIST curves).

If the *base field* is known in advance, then a speed-up can be obtained by pre-computing certain components of the calculation and storing as partial results – these are sometimes known as public key multiples. The technique is applicable for both signing and verification and can be extended if the public key is also known in advance.

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 public key including *domain* parameters and public key multiples can take of the order of ten times the space of an equivalent RSA key. Storage of an ECC private key takes a little over twice that of an RSA key. Certificate sizes are comparable, with RSA certificates being some 30% larger for equivalent key strength.

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 lever on 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[6]) is less efficient than the Number Field Sieve (NFS) 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 August 2015, the US National Security Agency (NSA) released a major policy statement on the need for post-quantum cryptography and their preliminary plans to transition to quantum resistant algorithms [178][170]. The NSA believes the time is right to make a major push to design public-key cryptographic protocols whose security depends on hard problems that cannot be solved efficiently by a quantum computer[7]. This announcement and more in particular certain peculiarities in the wording have puzzled many people and have given rise to much speculation concerning the NSA, elliptic curve cryptography and quantum-safe cryptography. A critical analysis on this has recently been published in [179]. See also section 3.1.5 of this report.

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;*

- *Use 1024-bit RSA keys and 160-bit ECC keys only for short term protection (e.g. ephemeral authentication for single devices);*

- *Only use 1024-bit RSA keys and 160-bit ECC keys for decrypting previously-encrypted data or verifying cryptographic protection (e.g. signature verification) rather than for applying cryptographic protection (signature generation or data encryption);*

- *Use at least 2048-bit RSA or 224-bit ECC for medium term protection (see [176]);*

- *For considerations regarding low exponent RSA, section 3.1.3 should be consulted;*

---

[6] Newer methods introduced by Antoine Joux against elliptic curves over binary fields are also important to note [183].

[7] For many years it has been known that both the integer factorisation problem (upon which RSA is based), and the elliptic curve discrete logarithm problem (upon which ECC is based), can be solved in polynomial time by quantum computers.

> - *New systems should be designed with the agility to introduce PQC algorithms as and when they are standardised (see 3.1.5).*

See section 3.1.4 for more details.

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 [120]. ECRYPT used to publish a yearly report on algorithms and key sizes [164] which addresses the same issue. For more recent recommendations see the ENISA report [176]. The following text and table is adapted from the NIST guideline.

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	SHA-1	1024	160
112	3TDES (2TDES)	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: Equivalent strengths (adapted from [120])

Notes:

- 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 will often be the case for payment systems, 2TDES equates in strength to 3TDES. The lower strength of 80 bits occurs when the attacker has of the order of 2^{40} plaintext and ciphertext pairs (see reference to SC27 SD12 below)[8].
- When collision resistance is not required the security level of hash functions is doubled.
- For DSA and DH the key sizes are comparable to RSA, with subgroup parameter comparable to ECC.
- Additional information on key strengths for AES and RSA and ECC are provided in [169] and [170].

To quote ISO/IEC JTC1 SC27 SD12:

http://www.jtc1sc27.din.de/sixcms_upload/media/3031/ISO-IECJTC1-SC27_N9780_SD12_20110415.pdf

"The security level of two-key Triple-DES may be assessed at less than the number of bits in the keys (112 bits) because of a "meet-in-the-middle" attack on the Triple-DES construction. An attacker with access to 2^{40} plaintext/ciphertext pairs encrypted under the same key will benefit from an exhaustive key search equivalent to 80 bit key search according to the attack described in [9]. The implication of this is that the effective key-length of two-key Triple-DES in specific applications can only be regarded as 80 bits (instead of 112 bits).

For many practical applications this degradability of the effective key-length is not necessarily a problem as access to 2^{40} plaintext/ciphertext pairs encrypted under a single key is quite unlikely. However, conservative system security design will not encrypt a too huge amount of data under the same key before re-keying."

The ECRYPT report on Algorithms and Keysizes [164] assesses that 2TDES restricted to a million plaintext/ciphertext pairs might provide 10 years protection at a 96-bit legacy standard security level. See also the more recent ENISA report [176]. It should be noted that a paper [181] published in 2016 has refined the above analysis and shows that the attack is effective even if the plaintext/ciphertext pairs are not all encrypted under the same key. In particular in this case the attacker will expect to determine the key used for at least one of the pairs and, moreover, will be able to do so when even only partial plaintexts are known.

3.1.5 Quantum computing considerations

For long-kept secrets or signatures, it might be useful to consider current developments in physics: there is a possibility[9] that a so called "quantum

---

[8] It is this consideration primarily that distinguishes the recommendations in the current report from those of NIST.

[9] The likelihood that a quantum computer is going to be built is outside the scope of this document.

computer" could be built in the next decades.

Such a device could solve certain cryptographic problems with lower complexity than current computers.

Two current quantum algorithms are relevant for cryptography:

- *Shor's algorithm* allows to reduce the complexity of solving both RSA and elliptic curve based problems enormously. The key length increase needed to compensate for this results in key sizes that are not practical.
- *Grover's algorithm* allows the execution of searches for key space of secret key based algorithms and hash functions in a time proportional to the square root of the key space (i.e. the effective length of the key is halved). To compensate for this, doubling of the key space is enough (e.g. using AES-256 instead of AES-128). This implies that there is not much difference between second preimage resistance and one-wayness (as defined in 2.1.1) anymore.

Assuming (very) optimistically that an attacker would have access to a quantum computer that has a performance comparable to current computers, the following changes would have to be made to current cryptographic primitives to stay at a comparable security level:

- Secret key based cryptosystems will have to double their key length.
- Hash function based systems that assume one-wayness will have to use hash functions that are collision resistant.
- Although no quantum computing attacks on the data block size of block ciphers are known, Grover's algorithm could reduce the effort for dictionary based attacks; therefore, block ciphers with small data block size (smaller than 128 bits) are to be avoided.
- Current public key cryptosystems will have to be replaced by a new class of "quantum resistant" public key cryptosystems that are currently in development. As of this writing, no standard efficient quantum resistant public key cryptosystems are available.

EPC recommendation 4
In view of quantum computing, if long term confidentiality (that is, for several decades) is needed:
- Do not use ciphers with small data block size (such as TDES)
- Use sufficiently long keys (at least 256 bits)
- Use a sufficiently long hash function (at least 512 bits).

3.1.6 ISO Recommendation for Financial Services

In the ISO Technical Report [2], ISO TC68 provides a list of recommended ISO cryptographic algorithms for use within applicable ISO TC68 – Financial Services standards. It also provides strategic guidance on key lengths and associated parameters and usage dates. It should be noted that this ISO report was published in 2010 and is currently being revised.

The focus is on algorithms rather than protocols, and protocols are in general not included in this document. However, in some cases, for example for some key

agreement and some authentication protocols, there is no "underlying" algorithm, but the protocol in a sense *is* the algorithm. In this case the mechanisms are included, in particular where they have security parameters that can be adjusted for higher or lower security.

Algorithmic vulnerabilities or cryptographic keys of inadequate lengths are less often the cause of security compromises in the financial industry than inadequate key management or other procedural flaws, or mistakes in the *implementation* of cryptographic algorithms or the protocols that use them. However, compromises caused by algorithmic vulnerabilities are more systemic and harder to recover from than other kinds of compromises.

This document deals primarily with recommendations regarding algorithms and key lengths. For standards on key management, see ISO/IEC 11568 [7].

The categories of algorithms covered are:

- Block ciphers
- Stream ciphers
- Hash functions
- Message authentication codes (MACs)
- Asymmetric algorithms
- Digital signature schemes giving message recovery
- Digital signatures with appendix
- Asymmetric ciphers
- Authentication mechanisms
- Key establishment and agreement mechanisms
- Key transport mechanisms.

This document does not define any cryptographic algorithms, however the standards to which this document refers may contain necessary implementation information as well as more detailed guidance regarding choice of security parameters, security analysis, and other implementation considerations.

3.1.7   Patents on cryptographic techniques

Many cryptographic techniques have been patented, many in the United States, but also in some European countries.

Before designing a system, users should consider whether the cryptographic techniques involved are currently protected by an unexpired patent valid for the countries where the system is to be used. Cryptographic algorithms and mechanisms that have been standardised may have a statement in the standard if the contributor participated in the standardisation process. This will indicate that licenses will be available on "reasonable and non-discriminatory" terms, but such statements should not be relied upon as they do not address the whole picture.

[130] contains an extensive discussion on cryptographic patents.

Table 3 contains information about some well-known patented cryptographic primitives. As most patents are filed in the US, US patent numbers are given. A US patent is only valid in the United States. But in some cases, a technique covered by a US patent may also have been patented under European, or other international patent organisations. In some cases, patents are for efficient implementation

techniques, rather than a complete algorithm.

US patents are valid either for 17 years after the date of issue, or 20 years from the date a patent application was filed, thus most of the patents listed in Table 5 have already expired (for instance, the RSA patent expired on September 20th, 2000).

Inventors	US Patent #	Issue date	Major claim or area
Rivest-Shamir-Adleman	4,405,829	September 20 1983	RSA system
Hellman-Diffie-Merkle	4,200,770	April 29 1980	Diffie-Hellman agreement
Massey-Lai	5,214,703	May 25 1993	IDEA cipher
Kravitz	5,231,668	July 27 1993	DSA signatures
Collins-Hopkins-Langford-Sabin	5,848,159	December 8 1998	Multiprime RSA

Table 5: Patent information

More information on US or international patents may be obtained from: http://www.micropatent.com and ISO also provides information on its standards that are known to be subject to patents

When using Elliptic Curve Cryptography (ECC) it is generally advisable to clarify potential patent issues, as some standards and implementation techniques are known to be covered (e.g.,

http://www.secg.org/index.php?action=secg,docs_member_patents).

3.2 Constructions

3.2.1   Symmetric Key Encryption

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 [59].

3.2.1.1   Modes of Operation

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

---

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

10116 [22]). The Modes defined in [22] are:

- Electronic Code Book (ECB)
  - o 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.

- Cipher Block Chaining (CBC)
  - o 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.
  - o 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
  - o 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:
  - o 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).
  - o 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).
  - o 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)
  - o This mode operates like a stream cipher.
  - o 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 [22] should be consulted for more information on these modes including their usage and security properties.

Other modes have been described in ISO TR 19038 [6]. 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.

3.2.2 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 exchange combined with a symmetric encryption scheme.

To encrypt a message, a key for a symmetric encryption schemes (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 his private key. He 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 [137].

Hybrid encryption schemes are defined in PKCS#1 [105] and in IEEE P1363-2000 [132] and in ISO/IEC 18033-2 [57].

3.2.4 MACs

3.2.4.1 CBC MACS

Two parties can encrypt messages to one another using a block cipher in some mode of operation. However if all that is required is data integrity or authentication rather than confidentiality then the message can be sent in clear but with a Message Authentication Code appended to it. For CBC MACs this 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 [26]).

Retail and wholesale requirements for MACs are defined in ISO 16609: "Banking - Requirements for Message Authentication using symmetric techniques" [5]. 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 5.1.2.5).

The CMAC mode of NIST can be used to compute MAC values based on AES. See [117] and Algorithm 5 of [26].

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 [27] is an example of such a function. A NIST standard FIPS 198 "The keyed-hash Message Authentication Code (HMAC)" [84] which claims to be a generalisation of RFC 2104 [91] and of the withdrawn ANSI X 9.71 was published in 2002.

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.

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 [26] 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 [24] and ISO/IEC 14888 [45] standards and IEEE P1363 [132] describe signature schemes using Integer Factorisation methods (i.e. RSA [124]) and Discrete Log methods. The PKCS#1 [104], [105] 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 [114] 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 [24], DSA [79] and a digital signature scheme based on Elliptic Curves [82].

Comparison criteria for digital signature schemes

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

- Security
  - o The certainty that the digital signature is uniquely linked to the signatory and to the data being signed
- Complexity of setting up the system
  - o 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
  - o 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
  - o 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.

In the table below, the results of the comparison are listed: τ stands for a negative point, while σ 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	τ (< hours)	σ (<Minutes)	σ (< Minutes)
Users computational capacity	σ (simpler chip)	τ (more complex chip)	τ (more complex chip)
Users storage capacity	σ (160 bits per key) (*)	τ (1024 bits per key)	τ (1024 bits per key)
Creation time digital signature	σ	τ (6 times ECDSA)	τ (4 times ECDSA)
Verification time digital signature	τ (4 to 7 times RSA)	σ	τ (28 times RSA)
Size digital signature	σ (320 bits per sig.)	τ (1024 bits per sig.)	σ (320 bits per sig.)

Table 6: Comparison of signature schemes

(*) Note that system parameters and pre-computation storage may be significant and may need to be taken into account.

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.

EPC recommendation 5
Except for systems where computational or data storage capacities are restricted, or where a strategic long-term migration is needed, there is no strong reason to move from RSA to ECC- based signature schemes.

3.2.6   Authenticated Encryption

If a message requires both confidentiality and integrity protection then one may use both encryption and a MAC or 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 [62]. 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:

- OCB (Offset CodeBook)[11]
- Key Wrap
- 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.

---

[11] OCB patent status can be found at http://www.cs.ucdavis.edu/~rogaway/ocb/

Another construction is MAC then Encrypt where the MAC is used as an IV for the encryption. This method has proofs of security in the form of SIV [168] and a variant of this is being standardised in ISO 20038 Banking and related financial services - Key Wrap.

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:

- 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" [15] 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.

EPC recommendation 6
In designing a security protocol the following should be identified:
- *the security service provided,*
- *the entities involved,*
- *the security verification values (MACs, hashes, digital signatures, …) that will need to be generated and verified,*
- *the algorithms and their modes of operation,*
- *the key material to be generated,*
- *the random number generators to be used (e.g. key generation, challenges, padding),*
- *the key establishment material (this could be key names, encrypted session keys, public key certificates or certificate references, initialisation vectors if any, …),*
- *unambiguous definition of the data used as input in the cryptographic mechanisms,*
- *the formatting techniques used in the signature generation/verification process,*
- *the transformation techniques used to represent binary data (known as filtering), if any.*

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, PKCS#11 and CBC encryption and may be avoided by the detection of such tampering (e.g. by using authenticated encryption). See also EPC Recommendation 6 above.

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 an HSM) 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, so that the receiver can first 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 7
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 140-2[12]) 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 [149] 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.

---

[12] A new version of FIPS 140-3 is currently under development and should be officially published in 2009.

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 algorithms (i.e. asymmetric keys).

More information on this topic may be found, for instance in: HAC [130], ISO 11568-1 [7], ISO/IEC 11770-1 [39] , ISO/IEC 9594-8 [21], Davies and Price [128].

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

This section describes key management for symmetric algorithms, and in particular block ciphers, covered by ISO 11568-2 [8] and ISO/IEC 11770-2 [40]. For many years TDES has been the most commonly used block cipher in the financial industry typically using double-length (112 bits) keys. However, many new systems - especially those providing general security protection - are implemented using TDES with triple-length (168 bits) keys or the AES block cipher with 128, 192 or 256 bit keys.

Key management involves the managing and protection of cryptographic keys 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 (e.g. Key MAC'ing Keys instead of Key Encrypting Keys).

4.1.1   Key generation

Keys should be generated in a TRSM (Tamper-Resistant Security Module) – as described in standard FIPS PUB 140-2 [80] 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.
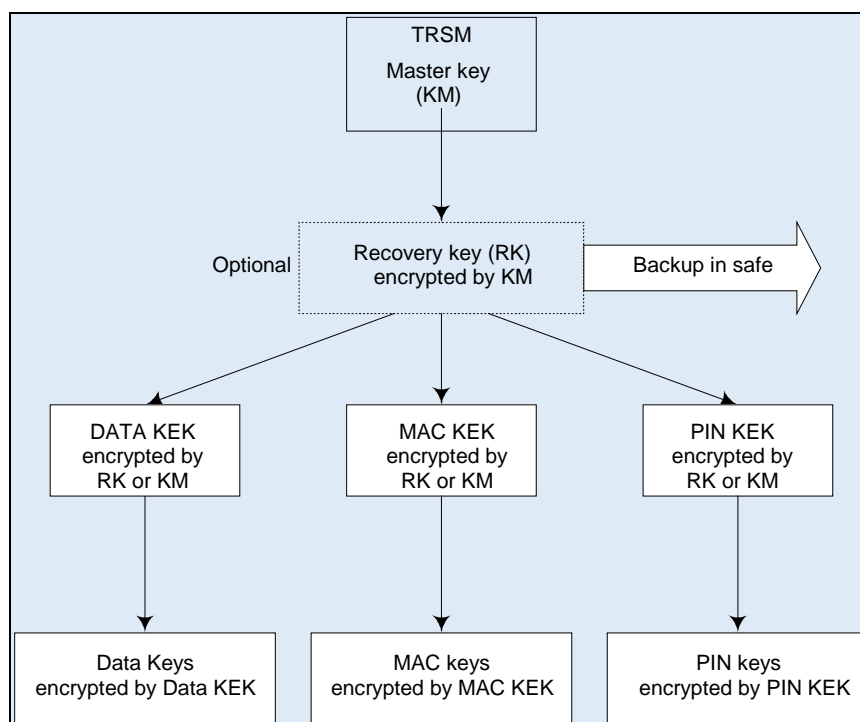
Figure 2: Example of key hierarchy for symmetric keys

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, smart cards are often used because they are more secure than passwords[13], and may have many other applications.

- 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

---

[13] 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.

master key. More complex secret sharing systems such as Shamir's threshold scheme may be used (described in [130]). 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 paper "The unbearable lightness of PIN cracking" at

http://www.arx.com/files/Documents/The_Unbearable_Lightness_of_PIN_Cr acking.pdf)).

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 [130] where an extensive bibliography on randomness may be found.

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.

EPC recommendation 8
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.

4.1.3   Key distribution

In hybrid crypto systems (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 7.2 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 most often 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 [40] and other references provided at the beginning 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 the result 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. Although there is still no standard on control vectors and key wrapping, 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 NIST SP800-38F [118], ANSI x9.102 [79] and TR 19038 [6]. See also [168].

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

EPC recommendation 9
Symmetric keys should be dedicated to one usage (e.g. encryption or MAC computation, but not both).

EPC recommendation 10
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 after use. 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 11568-2 [8] and NIST SP 800 57 [120].

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 single common parent key but may, as is the case with DUKPT x9.24 [70], be a sequence of keys where each 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 more valuable) and so may be kept for longer period (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, which is the most commonly used. For other key distribution schemes refer to the Diffie-Hellman protocol [106], [132], ISO 11568-4 [9] and ISO/IEC 11770-3 [41].

On Certification Authorities and public key certificate management, the reader is

referred to ISO 15782 and ISO 21188 [13].

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 140-2 [80], or in ISO 13491 [10]. The keys should be generated inside the TRSM and the private key should never exist in clear text outside the TRSM.

EPC recommendation 11
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.

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.

### 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:

- 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.
- 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.
- Session key: key used in a protocol between nodes in a network. Typically a symmetric key, e.g. single or double length DES key or AES key.
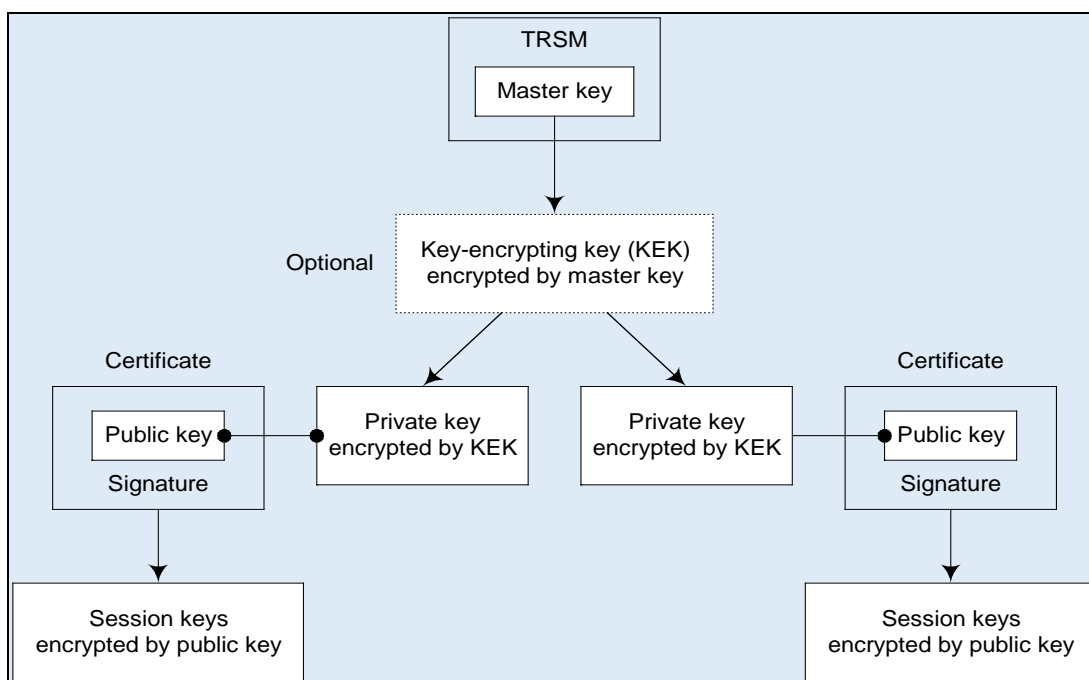
Figure 3: A hybrid key hierarchy with asymmetric and symmetric keys (for data confidentiality)

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 as ISO 15782-1 [3] 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 back up (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 environment concerned with production (e.g. smart cards). However the public key must be distributed to the parties to whom communication will take place.

Distribution of the public key is a very important security issue. The receiver of the public key must be absolutely sure of the authenticity and integrity of the delivered and stored public key, i.e. he must be sure that the key belongs to the declared owner of the key and that the key has not been modified at a later point in time. If for example the original public key is replaced with a public key from a malicious person or organisation, then the receiver 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 person. So, the risk of public key substitution during its distribution requires appropriate security controls.

There are various ways to distribute public keys. For example:

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

In the first method the integrity and authenticity of the distribution channel has to be provided e.g. by Transport Layer Security v1.2 or higher [100] or by a multiplicity of sources (e.g. hashes published in national newspapers and website). The disadvantage of the second 'disk' method is that the distribution has to be performed either personally, which almost always is impractical in large user communities, or by post where the user must be sure that it is a genuine and unbroken envelope. Neither of these methods preserves the integrity of the public key after installation on the user's computer unless a hash or a MAC is applied. Where the operation of a certification authority is practical and appropriate the third method is therefore recommended.

EPC recommendation 12
Where appropriate public keys shall be distributed such that their integrity and the binding with the owner are preserved (e.g. by using certificates).

A trusted third party called a Certification Authority (CA) must sign certificates. Certificate Management and Certification Authorities are complex subjects warranting separate discussion outside of this document.

For certificate management ISO TC 68 published two standards: 15782 [3] and 21188 [13] .

Some PKCS specifications cover certificate management: certificate format in PKCS #7 [108] and certificate requests in PKCS #10 [110].

The widely used standard format for certificates is the X.509 [21].

EPC recommendation 13
Usage of X 509, version 3, format certificates is recommended.

Often there is confusion between public key certificate 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 from one party to another using symmetric techniques) then it is important to consider Diffie-Hellman key agreement and EC-based variants of it (see [41]) in conjunction with key derivation techniques (see ISO/IEC 11770-6 and [119].If ephemeral or randomised keys are also used in Diffie-Hellman then it is possible to achieve forward secrecy whereby an attacker (who subsequently compromises one of the Diffie-Hellman key establishment keys) is unable to determine the key that was established and the payload it protected. This property is considered to be very important for the new TLS 1.3.

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 [51]).

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

EPC recommendation 14
When verifying a digital signature, users should check that the certificate status (either valid, expired or revoked) when the signature was generated does not make it invalid. When a certification path is involved, this verification should be performed for every certificate of 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 sufficient for some implementations, however there are also certified time sources available, i.e. Time Stamping and Notary Services, that 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.

4.2.8   Key usage and key separation

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 encryption 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 [21] contain an extension which indicates the purposes for which the certified public key is used.

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 RSA key pair will be measured in years. Considerations on key length and crypto period may be found in [8], [95], [129], [138], [139], and [144].

4.3 Key Escrow and Recovery

4.3.1 Key recovery

Key recovery and key escrow are two terms that are often used as synonyms. A key recovery system is defined [131] as an encryption with a backup decryption capability that allows authorised persons to decrypt cipher text with the help of information supplied by one or more trusted third parties who hold special data recovery keys. More specifically, a key escrow system is a system in which copies of private or secret keys are escrowed by a Third Party, which will be allowed to use them under legally specified circumstances.

Key recovery is used to ensure that stored, enciphered data can be retrieved and deciphered even if the ordinary way of deciphering data is not possible. This may be an essential part in disaster recovery planning and in the sections below, key recovery will usually be discussed in the context of governmental or other external agencies wanting to get access to information." Key recovery can also be essential for companies to retrieve information enciphered by employees who have met with misfortune, or who have left the company.

Key recovery was proposed as a technique for complying with due diligence to provide law enforcement and national security agencies with legal access to encrypted information. In this context key escrow is the term used when the keys for access are held by a third party. However in practice this objective is nowadays achieved through normal key recovery allowing for providing the data itself.

It is possible to build key recovery either into the key management or into the cryptographic algorithm itself. However, introducing a secret trapdoor into an algorithm contravenes the good cryptographic design principle, which states that only keying material should require secrecy protection.

Requirements to apply key recovery may restrict the choice of key management schemes and algorithms. Key recovery requirements might add increased cost to the solutions, both because trusted third parties representing the key recovery authorities will be needed and because complexity will be added to the technical solutions.

EPC recommendation 17

Payment service providers shall not use key escrow, but key recovery is part of their due diligence and business continuity obligations.

---

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".

The importance of good random number generation has received wide attention, especially in the area of prime selection for RSA key generation where some internet research (see [175]) 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. Recommendations and requirements for generating random numbers can be found in the following standards:

ISO/IEC 18031: Information Technology – Security Techniques – Random bit generation [55];

ANSI X9.82: Financial Services – Random Number Generation [78];

        Part 1: Overview
        Part 2: Entropy Sources
        Part 3: Deterministic Random Bit Generators
        Part 4: Random Bit Generator Construction

NIST SP 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators [121];

NIST SP 800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications [115].

Note that concerns have been expressed regarding the security of the Dual Elliptic Curve Deterministic Random Bit Generator (Dual_EC_DRBG) and so it has been officially withdrawn from both NIST [121] and ISO 18031 [55] standards. Users of this random generator are recommended to switch to one of the other random number generators in these standards.

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 [14])

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 [14])

Cryptoperiod: See key cryptoperiod.

Data integrity: The property that data has not been altered or destroyed in an unauthorised manner. (ISO 7498-2 [14])

Data origin authentication: The corroboration that the source of data received is as claimed. (ISO 7498-2 [14])

Decipherment: The reversal of a corresponding reversible encipherment. (ISO 7498-2 [14])

Decryption: See decipherment. (ISO 7498-2 [14])

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 [14])

Encipherment: The cryptographic transformation of data (see cryptography) to produce ciphertext. (ISO 7498-2 [14])

Encryption: See encipherment. (ISO 7498-2 [14])

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 [21])

Key: A sequence of symbols that controls the operations of encipherment and decipherment. (ISO 7498-2 [14])

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.

Modification Detection Code: A data item derived from a message using an

unkeyed hash function to provide message integrity.

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 [21])

Public key: (In a public key cryptosystem) that key of a user's key pair which is publicly known. (ISO 9594-8 [21])

Repudiation: Denial by one of the entities involved in a communication of having participated in all or part of the communication. (ISO 7498-2 [14])

Secret key: A key used with symmetric cryptographic techniques and usable only by a set of specified entities. (ISO/IEC 11770-1 [39])

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 [57]).

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

Acronyms

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
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
ECDSA	Elliptic Curve Digital Signature Algorithm
EESSI	European Electronic Signature Standardisation Initiative
EMV	Europay MasterCard Visa
EPC	European Payments Council
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
KEK   Key Encrypting Key
MAC   Message Authentication Code
MDC   Modification Detection Code
MDn   Message Digest n
MQV   Menezes Qu Vanstone
NESSIE   New European Schemes for Signatures, Integrity, and Encryption
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

Annex II: Bibliography

ISO Standards

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

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

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

[2] ISO TR 14742, "Financial services -- Recommendations on cryptographic algorithms and their use"

[3] ISO 15782-1, "Certificate management for financial services - Part 1: Public key certificates"

[4] ISO 15782-2, "Banking - Certificate management - Part 2: Certificate extensions"

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

[6] ISO TR 19038 "Triple DEA — Modes of Operation — Implementation Guidelines"

[7] ISO 11568-1, "Banking - Key management (retail) - Part 1: Principles"

[8] ISO 11568-2, "Banking - Key management (retail) - Part 2: Symmetric ciphers, their key management and life cycle"

[9] ISO 11568-4, "Banking - Key management (retail) - Part 4: Asymmetric cryptosystems – Key management and life cycle"

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

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

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

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

Joint Technical Committee ISO/IEC JTC 1 Information Technology

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

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

[16] 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)

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

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

framework"

[19] ISO/IEC 10181-5, "Information technology - Open Systems Interconnection - Security frameworks for open systems - Part 5: Confidentiality framework"
[20] 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

[21] 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

[22] ISO/IEC 10116, "Information processing - Modes of operation for an n-bit block cipher algorithm"
[23] ISO/IEC 9796-1, "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 1: Mechanisms using redundancy", replaced by 14888
[24] ISO/IEC 9796-2, "Information technology - Security techniques - Digital signature scheme giving message recovery - Part 2: Integer factorisation based mechanisms"
[25] ISO/IEC 9796-3, "Information technology -- Security techniques -- Digital signature schemes giving message recovery -- Part 3: Discrete logarithm based mechanisms"
[26] ISO/IEC 9797-1, "Information technology - Security techniques – Message Authentication Codes (MACs) Part 1: Mechanisms using a block cipher"
[27] ISO/IEC 9797-2, "Information technology - Security techniques – Message Authentication Codes (MACs), Part 2: Mechanisms using a dedicated hash function"
[28] ISO/IEC 9797-3, "Information technology - Security techniques – Message Authentication Codes (MACs), Part 3: Mechanisms using a universal hash function"
[29] ISO/IEC 9798-1 "Information technology -- Security techniques -- Entity authentication Part 1: General"
[30] ISO/IEC 9798-2 "Information technology -- Security techniques -- Entity authentication Part 2: Mechanisms using symmetric encipherment algorithms"
[31] ISO/IEC 9798-3 "Information technology -- Security techniques -- Entity authentication Part 3: Mechanisms using digital signature techniques",
[32] ISO/IEC 9798-4 "Information technology -- Security techniques -- Entity authentication Part 4: Mechanisms using a cryptographic check function"
[33] ISO/IEC 9798-5 "Information technology -- Security techniques -- Entity authentication Part 5: Mechanisms using zero knowledge techniques"
[34] ISO/IEC 9798-6 "Information technology -- Security techniques -- Entity authentication Part 6: Mechanisms using manual data transfer"
[35] ISO/IEC 10118-1, "Information technology - Security techniques – Hash-

functions - Part 1: General"

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

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

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

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

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

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

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

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

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

[45] ISO/IEC 14888-1, "Information technology - Security techniques - Digital signatures with appendix - Part 1: General"

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

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

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

[49] ISO/IEC 15946-2, "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures", withdrawn

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

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

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

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

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

[55] ISO/IEC 18031, "Information technology - Security techniques - Random bit generation"

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

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

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

[59] ISO/IEC 18033-3, "Information technology - Security techniques -

Encryption algorithms - Part 3: Block ciphers"

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

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

[62] ISO/IEC 19772, "Information technology - Security techniques - Authenticated encryption"

[63] ISO/IEC 29150, Information technology - Security techniques - Signcryption"

ETSI Standards

ETSI standards available from http://www.etsi.org

[64] ETSI TS 101 733, "Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)"

[65] ETSI TS 101 903, "XML Advanced Signatures (XAdES)"

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

[67] 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"

[68] ETSI TS 119 312, "Electronic Signatures and Infrastructures (ESI); Cryptographic suites"

ANSI Standards

ANSI standards are available from: http://web.ansi.org

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

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

[70] ANSI X9.24-1, "American National Standard - Financial Industry Standards – Retail Financial Services Symmetric Key Management - Part 1: Using Symmetric Techniques"

[71] 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"

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

[73] 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"

[74] ANSI X9.44, "Key Establishment Using Integer Factorization Cryptography"

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

[76] 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"

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

[78]ANSI X9.82-1, "Random Number Generation, Part 1: Overview and Basic Principles"

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

NIST FIPS standards

All FIPS standards are available at: http://www.itl.nist.gov/fipspubs/

[80]FIPS 140-2, "Security requirements for cryptographic modules", Federal Information Processing Standards Publication, US National Institute of Standards and Technology (supersedes FIPS PUB 140-1).

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

[82]FIPS 186-4, "Digital signature standard", Federal Information Processing Standards Publication, US National Institute of Standards and Technology.

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

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

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

Internet drafts, "standards" and RFCs

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

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

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

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

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

[90]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.

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

[92]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).

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

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

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

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

[97] 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.

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

[99] RFC 4301 "Security Architecture for the Internet Protocol", Internet Request for Comments 2401; S. Kent, K. Seo, + Errata.

[100] RFC 5246 "The Transport Layer Security (TLS) Protocol Version 1.2", Internet Request for Comments 5246, T. Dierks, E. Rescorla, + updates (see web site).

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

W3C Recommendations

[102] XML Signature Syntax and Processing (Second Edition), W3C recommendation 2008: http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/http://www.w3.org/TR/2008/PER-xmldsig-core-20080326/.

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

PKCS "standards"

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

Note that PKCS #1 is now instead maintained as RFC 3447 [94].

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

[105] PKCS #1, "The Public key cryptography standards - Part 1: RSA cryptography standard", version 2.1, 2002.

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

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

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

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

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

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

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

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

EMV specifications

EMV specifications are available from EMVCo, www.emvco.com:

[114]   EMV 4.3, "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 4.3, EMVCo, 2011.

NIST Special Publications

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

[115]   NIST SP800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications."

[116]   NIST SP 800-38A, "Recommendation for Block Cipher Modes of Operation - Methods and Techniques", + Addendum.

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

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

[119]   NIST SP 800-56c, "Recommendation for Key Derivation through Extraction-then-Expansion".

[120]   NIST SP 800-57 Revision 4, "Recommendation for key management Part 1: General guideline".

[121]   NIST SP800-90A, "Recommendations for Random Number Generation Using Deterministic Random Bit Generators".

[122]   NIST SP 800-131A Revision 1, "Recommendation for Transitioning the use of Cryptographic Algorithms and Key Lengths".

[123]   NIST SP 800-133, "Recommendation for Cryptographic Key Generation".

Other references

[124]   R. RIVEST, A. SHAMIR and L. ADLEMAN, "A Method for obtaining Digital Signatures and Public Key Cryptosystems". Communications of the

ACM, Vol.21(2), pp 120-126 (1978).

[125]   J. DAEMEN and V. RIJMEN, "AES proposal: Rijndael", AES Algorithm Submission, September 3 1999, http://www.nist.gov/aes/.

[126]   J.L. MASSEY "SAFER K-64: A byte-oriented block-ciphering algorithm". In "Fast Software Encryption", R. Anderson, editor, Lecture Notes in Computer Science No. 809, p.1-17. Springer, 1994.

[127]   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.

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

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

[130]   A. J. MENEZES, P.C. VAN OORSCHOT and S. A. VANSTONE, "Handbook of applied cryptography", CRC Press, Boca Raton 1997, http://cacr.math.uwaterloo.ca/hac/.

[131]   D.E. DENNING and D.K. BRANSTAD, "A taxonomy for key escrow encryption systems", Communications of the ACM, 39 (February 1996), 34-40.

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

[133]   L. GUILLOU, J.J. QUISQUATER, M. WALKER, P. LANDROCK and C. SHAER "Precautions taken against various potential attacks in ISO/IEC DIS 9796 Digital Signature Scheme giving Message Recovery", Proceedings of Eurocrypt'90, Springer-Verlag, 1991.

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

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

[136]   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.

[137]   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.

[138]   A. LENSTRA and E. VERHEUL, "Selecting cryptographic key sizes", 1999.

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

[140]   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.

[141]   D. COPPERSMITH, S. HALEVI and C. JUTLA, "ISO 9796-1 and the new forgery strategy", Technical report, Research Contribution to P1363, 1999.

[142] D.J. BERNSTEIN, "Circuits for integer factorization", 2002.

[143] A.K. LENSTRA, A. SHAMIR, J. TOMLINSON and E. TROMER, "Analysis of Bernstein's factorization circuit", 2002, http://tau.ac.il/~tromer/papers/meshc.pdf.

[144] 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.

[145] S. MURPH and M.J.B. ROBSHAW, "Comments on the security of the AES and the XSL Technique", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.4866&rep=rep1&type=pdf

[146] N. COURTOIS and J. PIEPRZYK, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations", Advances in Cryptology – Asiacrypt '02, Lecture Notes in Computer Science 2501 (2002), Springer-Verlag, p.267 ff.

[147] S. HABER and B. PINKAS, "Combining Public-Key Cryptosystems", Proceedings of the ACM Computer and Security Conference, November 2001.

[148] 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.

[149] 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.

[150] B. KALISKI, "TWIRL and RSA Key Size. RSA Laboratories" Technical Note. May 6, 2003, www.rsasecurity.com.

[151] A. SHAMIR and E. TROMER, "Factoring large numbers with the TWIRL device", proc. Crypto 2003, LNCS 2729, Springer-Verlag, 2003.

[152] A. SHAMIR and E. TROMER , "On the Cost of Factoring RSA-1024", Weizmann Institute of Science, http://www.madchat.org/crypto/cbtwirl.pdf#search='RSA%201024'.

[153] NESSIE consortium, "Security Report" Version 2.0, February 19, 2003.

[154] NESSIE consortium, "Performance Report" Version 2.0, February 19, 2003.

[155] NESSIE consortium "Portfolio of recommended cryptographic primitives", February 27, 2003. Available via http://www.cryptonessie.org

[156] D.GALINDO, S.MARTIN and J.L.VILLAR, "Evaluating elliptic curve based KEMs in the light of pairings", available at: http://eprint.iacr.org/2004/084.

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

[158] X. WANG, Y.L. YIN and H. YU, "Collision Search Attacks on SHA1", February 13, 2005, http://theory.csail.mit.edu/~yiqun/shanote.pdf.

[159] J.L. MASSEY, "SAFER-K64: one year later," In B. Preneel, editor Fast Software Encryption, LNCS 1008, pp. 212–241, Springer-Verlag, 1995.

[160] A. LENSTRA, X. WANG, B. DE WEGER. "Colliding X.509 Certificates", version 1.0, 1st March 2005, http://eprint.iacr.org/2005/067.

[161] 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.

[162]   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.

[163]   ECRYPT, "Recent Collision Attacks on Hash Functions: ECRYPT Position Paper", Revision 1.1, 17 February 2005, www.ecrypt.eu.org.

[164]   ECRYPT, "ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012)", Editor Nigel Smart, Revision 1.0, 30th September 2012, www.ecrypt.eu.org.

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

[166]   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.

[167]   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.

[168]   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/.

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

[170]   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://documents.epfl.ch/users/l/le/lenstra/public/papers/ecdl.pdf.

[171]   G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, "The Keccak Sponge Family Function", June 2010, http://keccak.noekeon.org/Keccak-main-2.1.pdf.

[172]   G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, "The Keccak SHA-3 submission, January 14 2011, http://keccak.noekeon.org/Keccak-submission-3.pdf.

[173]   G. BERTONI, J. DAEMEN, M. PEETERS and G. VAN ASSCHE, "The Keccak Reference", January 14 2011, http://keccak.noekeon.org/Keccak-reference-3.0.pdf.

[174]   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.

[175]   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.

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

[177]   NIST, ITL Bulletin, "Additional secure hash algorithms offer new opportunities for data protection", Sept 2015, http://csrc.nist.gov/publications/nistbul/itlbul2015_09.pdf.

[178]   NSA, "Cryptography Today",
 https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

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

[180]   M. STEVENS, https://marc-stevens.nl/research/

[181]   C. MITCHELL, On the security of 2-key Triple DES arXiv:1602.06229v1 [cs.CR]

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

[183]   A. JOUX, A new index calculus algorithm with complexity $L(1/4+o(1))$ in very small characteristic, iacr e-archive 2013.

[184]   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.

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

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