ETSI TS 119 312 V1.1.1 (2014-11)

TECHNICAL SPECIFICATION

Electronic Signatures and Infrastructures (ESI);
Cryptographic Suites

Reference

DTS/ESI-0019312

Keywords

e-commerce, electronic signature, security, trust services

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI).

The present document replaces ETSI TS 102 176-1 (also known as "Algo Paper") [i.4].

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**may not**", "**need**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Selection of the cryptographic suites to apply for digital signatures is an important business parameter for products and services implementing digital signatures. The present document provides guidance on selection of cryptographic suites with particular emphasis on security. The present document identifies a range of different cryptographic suites that can be used corresponding to the appropriate level of security, which fulfils the security needs identified during the system design. There is no normative requirement on selection among the alternatives but for all alternatives, normative requirements apply to ensure security and interoperability.

The present document is based on various security recommendations given by other standardization bodies, security agencies and supervisory authorities of the Member States. National cryptographic recommendations including but not limited to French [i.2] and German [i.3] are considered in the present document.

1 Scope

The present document specifies cryptographic suites used for digital signature creation and verification algorithms.

The present document provides guidance on selection of cryptographic suites corresponding to the appropriate level of security, which fulfils the security needs identified during the system design. The present document identifies a range of alternative cryptographic suites. There is no normative requirement on selection among the alternatives but for all alternatives, normative requirements apply to ensure security and interoperability.

The present document also provides guidance on the hash functions, signature schemes and signature suites to be used with the data structures used in the context of digital signatures. For each data structure, the set of algorithms to be used is specified.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] FIPS Publication 180-4 (2012): "Secure Hash Standard (SHS)".

[2] FIPS Publication 186-4 (July 2013): "Digital Signature Standard (DSS)".

[3] IETF RFC 3447 (2003): "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1".

[4] ISO/IEC 14888-3 (2006): "Information technology - Security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms".

[5] IETF RFC 5639 (2010): "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation".

[6] ANSI X9.62 (2005): "Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)".

[7] IETF RFC 3279 (2002): "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

NOTE: Updated by RFC 4055, RFC 4491, RFC 5480, and RFC 5758.

[8] IETF RFC 4055 (2005): "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile".

[9] IETF RFC 5753: "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)".

[10] IETF RFC 6931 (2013): "Additional XML Security Uniform Resource Identifiers (URIs)".

[11] W3C Recommendation: "XML Encryption Syntax and Processing Version 1.1", April 2013.

NOTE: Available at http://www.w3.org/TR/2013/REC-xmlenc-core1-20130411.

[12] IETF RFC 3161 (2001): "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)".

NOTE: Updated by RFC 5816.

[13] IETF RFC 6960 (2013): "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

NOTE: Updates RFC 2560, RFC 6277.

[14] W3C Recommendation: "XML-Signature Syntax and Processing Version 1.1", April 2013.

NOTE: Available at http://www.w3.org/TR/2013/REC-xmldsig-core1-20130411.

[15] ETSI TS 101 861 (V1.4.1) (07-2011): "Electronic Signatures and Infrastructures (ESI); Time stamping profile".

[16] ISO/IEC 18031 (2011): "Information technology - Security techniques - Random bit generation".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ENISA: "Algorithms, Key Sizes and Parameters Report, 2013 recommendations, version 1.0" (2013-10).

NOTE: Available at http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report.

[i.2] Agence nationale de la sécurité des systèmes d'information, Référentiel Général de Sécurité version 2.0, 2014-06.

NOTE: Annex B1 (version 2.03 of 2014-02) is available at http://www.ssi.gouv.fr/IMG/pdf/RGS_v-2-0_B1.pdf.

[i.3] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, Übersicht über geeignete Algorithmen, 2014-01.

NOTE: Available at http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/2014Algorithmenkatalog.pd.

[i.4] ETSI TS 102 176-1 (V2.1.1) (07-2011): "Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures; Part 1: Hash functions and asymmetric algorithms".

NOTE: This reference is given only for informational purposes.

[i.5] ISO/IEC 18032 (2005): "Information technology - Security techniques - Prime number generation".

[i.6] ISO/IEC 10118-3 (2004): "Information technology - Security techniques - Hash functions - Part 3: Dedicated hash functions".

NOTE: This ISO Standard duplicates the standardization from FIPS Publication 180-4 [1].

[i.7] National Institute of Standards and Technology SHA-3 Competition (2007-2012).

NOTE: More details on winner selection are available at http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_selection_announcement.pdf.

[i.8] ANSI X9.82 (2006): "Random Number Generation Part 1".

[i.9] AIS 20/31: "Application Notes and Interpretation of the Scheme: Functionality classes and evaluation methodology for deterministic random number generators", Version 2.

[i.10]	ANSI X9.17: "Pseudo Random Number Generator (RNG)".
[i.11]	NIST Special Publication SP 800-90A: "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", January 2012.
[i.12]	ETSI TS 101 733 (V2.2.1) (04-2013): "Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)".
[i.13]	ETSI TS 101 903 (V1.4.2) (12-2010): "Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)".
[i.14]	ETSI TS 102 778 (parts 1 to 6): "Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles".
[i.15]	IETF RFC 5280 (2008): "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".
[i.16]	ISO/IEC 14888-3:2006/Amd.1 (2010): "Elliptic Curve Russian Digital Signature Algorithm, Schnorr Digital Signature Algorithm, Elliptic Curve Schnorr Digital Signature Algorithm, and Elliptic Curve Full Schnorr Digital Signature Algorithm" and ISO/IEC 14888-3:2006/Amd.2 (2012): "Optimizing hash inputs".
[i.17]	W3C Recommendation: "Canonical XML Version 1.0" (omits comments).

NOTE: Available at http://www.w3.org/TR/2001/REC-xml-c14n-20010315.

[i.18]	W3C Recommendation: "Canonical XML Version 1.0" (with Comments).

NOTE: Available at http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

cryptographic suite: combination of a signature scheme with a padding method and a cryptographic hash function

(digital) signature: data associated to, including a cryptographic transformation of, a data unit that:

a) allows to prove the source and integrity of the data unit,

b) allows to protect the data unit against forgery and

c) allows to support signer non-repudiation of signing the data unit.

hash function: As defined in ISO/IEC 10118-3 [i.6].

signature policy: set of rules for the creation and validation of a signature, that defines the technical and procedural requirements for signature creation and validation, in order to meet a particular business need, and under which the signature can be determined to be valid

signature scheme: triplet of three algorithms composed of a signature creation algorithm, a signature verification algorithm and a key generation algorithm

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AC	Attribute Certificates
ANSI	American National Standards Institute
CA	Certification Authority

CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
DRG	Deterministic Random Generator
DRNG	Deterministic Random Number Generator
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECGDSA	Elliptic Curve German Digital Signature Algorithm
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
MGF	Mask Generation Function
NIST	National Institute of Standards and Technology
NRNG	Non-deterministic Random Number Generator
OCSP	Online Certificate Status Protocol
OID	Object Identifier
PKCS	Public-Key Cryptography Standards
PSS	Probabilistic Signature Scheme
RFC	Request for Comments
RNG	Random Number Generator
RSA	Rivest, Shamir and Adleman algorithm
SCDev	Signature Creation Device
SHA	Secure Hash Algorithm
SP	NIST Special Publication
TST	Time-Stamp Token
TSU	Time-Stamping Unit
URI	Uniform Resource Identifier
URN	Uniform Resource Number
XML	Extensible Markup Language

4 Maintenance of the document

As a response to relevant developments in the area of cryptography and technology, activities for the maintenance of the present document will enable dynamic updating of the lists of recommended algorithms. The current list of recommended cryptographic hash functions and signature algorithms is given in the present document.

The maintenance activities will adjust security parameters, will introduce new cryptographic hash functions and signature algorithms and/or will lead to remove cryptographic hash functions and signature algorithms from the list. These activities need to respond to the following situations:

1) The need to introduce new algorithms and relevant parameters calls for a mechanism that is rather dynamic. Since it is important to maintain interoperability, updates may result from the adoption or removal of an algorithm in a document on which another ETSI deliverable is based upon.

2) Advances in cryptography call for a phasing out of some algorithms or parameters. Such phasing out will normally be known well in advance.

3) In the case of new attacks, the immediate need to remove an algorithm could arise.

The maintenance activities for cases 1 and 2 allow for some transition period during which the algorithm/parameter values to be revised can be used in already issued certificates and signature products. However, the revised items could not be considered for use in certificates to be issued and new signature products. The transition period allow vendors using the revised items to alter their production process. If the security implications of a revision are considered very serious, the certificates and signature products using the revised item could be withdrawn before their planned expiry date and other measures, like archival time stamps could be undertaken to preserve the security status of signatures based on it.

In order to allow an easy follow up of the present document, a history of the tables provided in the main body of the document is maintained and kept as annexes.

5 Hash functions

5.1 General

The list of hash functions in table 1 should be used.

Table 1: The list of recommended hash functions

Hash function entry index	Short hash function entry name	Adoption date	References
1.03	sha224	2004	FIPS Publication 180-4 [1]
1.04	sha256	2004	FIPS Publication 180-4 [1]
1.06	sha384	31.03.2007	FIPS Publication 180-4 [1]
1.07	sha512	31.03.2007	FIPS Publication 180-4 [1]
1.08	sha512/256	31.12.2013	FIPS Publication 180-4 [1]

NOTE 1: Each hash function has a unique entry index represented by a string beginning with "1." followed by a two-digit entry number.

NOTE 2: Hash functions can be used in a variety of cases, including the following:

- advanced electronic signatures TS 101 733 [i.12], TS 101 903 [i.13], TS 102 788 [i.14] include the identifier of the hash function used to compute the digital signature on the signed data;
- time-stamp tokens include the identifier of the hash algorithm used to compute the hash value for the time-stamped data; or
- public key certificates include the identifier of a signature suite which defines the hash function used to compute the digital signature on the certified data.

NOTE 3: A hash function takes as input a variable-length message and produces as output a fixed-length hash value.

NOTE 4: The Whirlpool algorithm defined in ISO/IEC 10118-3 [i.6] is removed from the corresponding table in ETSI TS 102 176-1 [i.4]. It is expected that the Keccak algorithm function, currently adopted as SHA-3, will be accepted as an alternative algorithm instead.

NOTE 5: The RIPEMD-160 algorithm defined in ISO/IEC 10118-3 [i.6] was already removed from the corresponding table in ETSI TS 102 176-1 [i.4] due to its low hash length and its minor dissemination.

5.2 Recommended hash functions

5.2.1 SHA-224

SHA-224 may be used to hash a message, *M*, having a length of up to 2^{64} - 1 bits. The output size is 224 bits.

SHA-224 should not be used, except for legacy applications, if SHA-256 can be used instead without truncation.

The SHA-224 algorithm shall be implemented as defined in FIPS Publication 180-4 [1].

NOTE: The SHA-224 function is identical to SHA-256, except that different initial values are used, and the final hash value is truncated to 224 bits.

5.2.2 SHA-256

SHA-256 may be used to hash a message, *M*, having a length of up to 2^{64} - 1 bits. The output size is 256 bits.

The SHA-256 algorithm shall be implemented as defined in FIPS Publication 180-4 [1].

5.2.3 SHA-384

SHA-384 may be used to hash a message, *M*, having a length of up to 2^{128} - 1 bits. The output size is 384 bits.

SHA-384 should not be used if SHA-512 can be used instead without truncation.

The SHA-384 algorithm shall be implemented as defined in FIPS Publication 180-4 [1].

NOTE: SHA-384 is identical to SHA-512, except that different initial values are used, and the final hash value is truncated to 384 bits.

5.2.4 SHA-512

SHA-512 may be used to hash a message, *M*, having a length of up to 2^{128} - 1 bits. The output size is 512 bits.

The SHA-512 algorithm shall be implemented as defined in FIPS Publication 180-4 [1].

5.2.5 SHA-512/256

SHA-512/256 may be used to hash a message, *M*, having a length of up to 2^{128} - 1 bits. The output size is 256 bits.

SHA-512/256 should not be used if SHA-512 can be used instead without truncation.

The SHA-512/256 algorithm shall be implemented as defined in FIPS Publication 180-4 [1].

NOTE: The difference to SHA-256 is the bigger inner state, which gives a better collision resistance.

5.3 Other hash functions

5.3.1 SHA-1 is no more recommended

SHA-1 should not be used.

SHA-1 may be used for certificate verification if the certificate body includes sufficient entropy (i.e. data bits neither known to nor predictable by the attacker prior to any data bits controllable by the attacker (see note 3)).

The SHA-1 algorithm shall be implemented as defined in FIPS Publication 180-4 [1] (see note 4).

NOTE 1: SHA-1 can be used to hash a message, *M*, having a length of up to 2^{64} - 1 bits and the output is a 160-bit message digest.

NOTE 2: Several attacks against SHA-1 have been discovered since February 2005. A collision of the full 80-round SHA-1 has not been published yet.

NOTE 3: All known collision attacks on SHA-1 require full control of certain substrings within the data to be hashed and knowledge of the data bits prior to these strings. This is considered as a realistic attack scenario for documents signed by signers (in particular, when a kind of "active" program element can be hidden in the document). On the other hand for X.509 certificates such attacks can be prevented by the CA by including at least 20 bits of entropy (i.e. data bits neither known to nor predictable by the attacker) in the certificate string prior to any data bits controllable by the attacker. This method leads to a considerably higher resistance of certificates against collision attacks which is not available for document signatures. SHA-1 remains resistant against second pre-image attacks.

NOTE 4: ISO/IEC 10118-3 [i.6] duplicates description of FIPS Publication 180-4 [1].

5.3.2 WHIRLPOOL is no more recommended

WHIRLPOOL, defined in ISO/IEC 10118-3 [i.6], should not be used.

NOTE 1: WHIRLPOOL is a hash function defined in ISO/IEC 10118-3 [i.6] that operates on messages less than 2^{256} - 1 bits in length, and produces a message digest of 512 bits. Whirlpool was defined as an alternative to the SHA-2 group after the collision weakness of the MD5, SHA-1 algorithms showed up.

NOTE 2: The recently NIST adopted SHA-3 Keccak function is expected to be used in the future as a replacement for SHA-2 algorithms (see clause 5.3.3).

5.3.3 SHA-3

NOTE: NIST announced Keccak as the winner of the SHA-3 Cryptographic Hash Algorithm Competition on October 2, 2012 [i.7]. NIST chose Keccak for its elegant design, large security margin, good general performance, excellent efficiency in hardware implementations, and for its flexibility. Keccak uses a new "sponge construction" chaining mode, based on a fixed permutation, that can readily be adjusted to trade generic security strength for throughput, and can generate larger or smaller hash outputs as required. Despite that SHA-2 is now widely implemented, and will continue to be used for the foreseeable future, it is expected that Keccak will serve as an alternative hash function. It seems very unlikely that a single new cryptanalytic attack or approach could threaten both algorithms.

6 Signature schemes

NOTE: A signature scheme consists of three algorithms: a key generation algorithm, a signature creation algorithm and a signature verification algorithm. The two latter are identified hereafter as a pair of algorithms. Each pair has its own name.

6.1 Signature algorithms

6.1.1 General

The list of signature algorithms given in table 2 should be used.

Table 2: The list of recommended signature algorithms

Signature algorithm entry index	Short signature algorithm entry name	Key and Parameter generation algorithms	References
2.01	rsa	rsagen1	RFC 3447 [3]
2.02	dsa	dsagen1	FIPS Publication 186-4 [2], ISO/IEC 14888-3 [4]
2.03	ecdsa-Fp	ecgen1	FIPS Publication 186-4 [2]
2.04	ecdsa-F2m	ecgen2	FIPS Publication 186-4 [2]
2.05	ecgdsa-Fp	ecgen1	ISO/IEC 14888-3 [4]
2.06	ecgdsa-F2m	ecgen2	ISO/IEC 14888-3 [4]

NOTE: Each signature algorithm has a unique entry index represented by a string beginning with "2." followed by a two-digit entry number.

The following clauses describe the parameters and key generation algorithms for the signature algorithms listed in table 2.

6.1.2 Recommended signature algorithms

6.1.2.1 RSA

NOTE 1: The RSA algorithm's security is based on the difficulty of factoring large integers. The private key consists of a positive integer d (the private exponent) and the modulus n. The public key consists of a positive integer e (the public exponent) and the modulus n.

NOTE 2: In the following the bit length of an integer p is an integer r such that $2^{r-1} \leq p < 2^r$.

The RSA algorithm shall be implemented as described in RFC 3447 [3].

To generate the key pair, two prime numbers, p and q, shall be generated randomly and independently, satisfying the following requirements:

- The bit length of the modulus $n = p\,q$ shall be at least MinModLen (see table 7).
- p and q should be selected uniformly such that either :
  - $0{,}1 < | \log_2 p - \log_2 q | < 30$; or
  - $\sqrt{2}\cdot 2^{n/2-1} \leq p \leq 2^{n/2}-1$, $\sqrt{2}\cdot 2^{n/2-1} \leq q \leq 2^{n/2}-1$, and $|p-q| > 2^{n/2-100}$.

For RSA signatures, a padding method shall also be selected (see clause 7.2).

NOTE 3: Depending on the choice of the padding (probabilistic or deterministic) the signature will be unique or not.

6.1.2.2 DSA

NOTE 1: The security of this signature algorithm, referred to as dsa, is based on the difficulty of computing the discrete logarithm in the multiplicative group of a prime field Fp.

The DSA algorithm shall be implemented as defined in clause 4 of FIPS Publication 186-4 [2] with the amendments defined in this clause.

NOTE 2: The same algorithm is also specified in ISO/IEC 14888-3 [4].

The bit length L of the prime modulus p shall be at least pMinLen bits long (see table 9).

The bit length N of q, which is a prime divisor of (p-1), shall be at least qMinLen bits long (see table 9).

NOTE 3: The dissemination of DSA is low. Therefore it is suggested to use other more widely deployed algorithms unless it is the only alternative for interoperability.

ISO/IEC 14888-3 [4] provides additionally the combinations $L = 7\,680$, $N = 384$ and $L = 15\,360$, $N = 512$ which may be used.

If it is not required by a signature length restriction, parameters with $N = 256$ should be used instead of parameter with $N = 224$.

NOTE 4: SHA-224 does not provide security advantages over SHA-256.

6.1.2.3 Elliptic curve analogue of DSA based on a group $E(F_p)$

NOTE 1: The security of this signature algorithm, referred to as ecdsa-Fp, is based on the difficulty of computing the elliptic curve discrete logarithm.

For the SHA hash functions, the algorithm shall be implemented as specified in clause 6 of FIPS 186-4 [2].

For non-SHA hash functions, the analogous algorithm specified in ISO/IEC 14888-3 [4] may be used.

The public parameters meet the following requirements:

- p shall be a prime number;
- n shall be a large prime number of at least nMinLen bits long (see table 10);
- p and n shall be different $(p \neq n)$;
- the order of the group $E(F_p)$ of points of the elliptic curve E over a finite field F_p, shall be divisible by n; and
- G point on the group $E(F_p)$ shall be of order n.

The quotient *h* (cofactor) of the group order divided by *n*, considered as a security parameter too, shall be less or equal to 4.

The class number of the maximal order of the endomorphism ring of *E* shall be of at least MinClass (see table 10).

The value r_0: = min (r: q divides p^r-1) shall be greater than r0Min = 10^4.

NOTE 2: FIPS Publication 186-4 [2] recommends five elliptic curves over a prime field. The RFC 5639 [5] contains an alternative set of curves over prime fields with 160 bits, 192 bits, 224 bits, 256 bits, 320 bits, 384 bits and 512 bits. All these curves fulfil the above requirements.

6.1.2.4 Elliptic curve analogue of DSA based on a group $E(F_{2^m})$

NOTE 1: The security of this signature algorithm, referred to as ecdsa-F2m, is based on the difficulty of computing the elliptic curve discrete logarithm.

The algorithm shall be implemented as specified in clause 6 of FIPS 186-4 [2].

NOTE 2: The same algorithm is also specified in ISO/IEC 14888-3 [4], which can be used for information too.

The public parameters meet the following requirements:

- *m* shall be a prime number;
- *n* shall be a large prime number of at least nMinLen bits long (see table 11);
- the order of the group $E(F_{2^m})$ of points of the elliptic curve *E* over a finite field F_{2^m}, shall be divisible by *n*;
- it shall not be possible to define *E* over $F_{2;}$
- *G* point on $E(F_{2^m})$ shall be of order *n*; and
- *h,* the order of $E(F_{2^m})$ divided by *n*, shall be less or equal to *4.*

The class number of the maximal order of the endomorphism ring of *E* shall be at least MinClass as defined in table 11.

The value r_0:=min(r: q divides 2^{mr}-1) shall be greater than r0Min = 10^4.

NOTE 3: In FIPS Publication 186-4 [2] five pseudo-randomly generated curves over F_{2^m} are defined. All these curves satisfy the above requirements.

6.1.2.5 EC-GDSA based on a group $E(F_p)$

NOTE 1: The security of this signature algorithm, referred to as ecgdsa-Fp, is based on the difficulty of computing the elliptic curve discrete logarithm.

The algorithm shall be implemented as specified in ISO/IEC 14888-3 [4].

NOTE 2: The ecgdsa-Fp algorithm is a variant of the ecdsa-Fp algorithm with a modified signature creation equation and verification method.

The parameters are the same as for ecdsa-Fp and therefore shall satisfy all the constraints given in clause 6.1.2.3.

NOTE 3: The advantage of EC-GDSA over ECDSA is that signature creation does not need an integer inversion. Under certain circumstances this can be important for the design and performance of the SCDev.

NOTE 4: The dissemination of EC-GDSA is low. Therefore it is suggested to use other more widely deployed algorithms unless it is the only alternative for interoperability.

6.1.2.6 EC-GDSA based on a group $E(F_{2^m})$

NOTE 1: The security of this signature algorithm, referred to as ecgdsa-F2m, is based on the difficulty of computing the elliptic curve discrete logarithm.

The algorithm shall be implemented as specified in ISO/IEC 14888-3 [4].

NOTE 2: The ecgdsa-F2m algorithm is a variant of the ecdsa-F2m algorithm with a modified signature creation equation and verification method.

The parameters are the same as for ecdsa-F2m and therefore shall satisfy all the constraints given in clause 6.1.2.4.

NOTE 3: For the difference between ECDSA and ECGDSA see note 3 in clause 6.1.2.5.

6.1.2.7 Other EC-DSA variants for future applications

NOTE 1: Among the EC-XDSA variants mentioned in ISO/IEC 14888-3 [4] and its amendments [i.16] only EC-KCDSA, EC-SDSA, EC-FSDSA and their optimized variants have provable security guarantees. The reason for this is that for EC-DSA and EC-GDSA the hash function is only applied to the message and not to a combination of the message and the ephemeral public key. All EC-XDSA variants also suffer from lattice attacks against poor ephemeral secret generation. Besides a recommended random number generation method (see clause 8) the ephemeral secret key can be derived by applying a pseudorandom function (with a default key) to a message containing the static secret key and the message to be signed.

NOTE 2: Schnorr signatures according to in ISO/IEC 14888-3/Amd.1 [i.16] differ from other EC-DSA variants by a modified signature creation equation and verification method, but use the same elliptic curves. They have the following advantages: firstly the signing equation is simpler (allowing for some optimizations) and secondly the hash function is applied to the concatenation of the message and the ephemeral key. With this property Schnorr signatures can be proved secure in the random oracle model. There is also a proof in the generic group model.

6.2 Key generation algorithms

6.2.1 General

NOTE 1: Key generation algorithms are not part of the definition of a signature suite and can evolve without the need to change the identifier of the signature suite.

The key generation algorithms listed in table 3 should be used for all signature algorithms considered in the present document. The amount of entropy (defined in ISO/IEC 18031 [16]) is a parameter of the generated random bits.

Table 3: The list of recommended key generation algorithms

Key generator entry index	Short key generator entry name	Signature algorithm	Random number generation method	Random generator parameters	Adoption date	References
3.01	rsagen1	rsa	trueran or pseuran	EntropyBits or SeedEntropy, resp.	01.01.2001	RFC 3447 [3] FIPS Publication 186-4 [2]
3.02	dsagen1	dsa	trueran or pseuran	EntropyBits or SeedEntropy, resp.	01.01.2001	FIPS Publication 186-4 [2]
3.03	ecgen1	ecdsa-Fp, ecgdsa-Fp	trueran or pseuran	EntropyBits or SeedEntropy, resp.	01.01.2001	FIPS Publication 186-4 [2]
3.04	ecgen2	ecdsa-F2m, ecgdsa-F2m	trueran or pseuran	EntropyBits or SeedEntropy, resp.	01.01.2001	FIPS Publication 186-4 [2]

NOTE 2: Each key pair generation method has a unique entry index represented by a string beginning with "3." followed by a two-digit entry number.

6.2.2 Recommended key generation algorithms

6.2.2.1 Key and parameter generation algorithm rsagen1

The components and the properties of the key are described in RFC 3447 [3], section 3.2 and FIPS Publication 186-4 [2], section 5.1. Examples of key generation algorithms are given in FIPS Publication 186-4 [2], annex B.3.

p and *q* shall be generated as indicated in clause 6.1.2.1 by applying a random number generation method satisfying the requirements trueran (see clause 8.2.2) or using a method satisfying pseuran (see clause 8.2.3) with an appropriate size seed.

Each prime shall depend on at least EntropyBits bits of true randomness or a seed of entropy SeedEntropy bits (see table 7).

Random numbers shall be tested for primality until one of them is found to be prime with a probability of error (i.e. of actually being composite) of at most ErrProb (see table 7).

NOTE 1: Details on generating random primes can be found in ISO/IEC 18032 [i.5], in particular clause 8.2.

The public exponent e shall be an odd positive integer such that $2^{16} < e < 2^{256}$.

The private exponent d and the public exponent e shall satisfy $ed \equiv 1 \pmod{\mathrm{lcm}(p-1, q-1)}$.

NOTE 2: It is automatically the case if $ed \equiv 1 \pmod{(p-1)(q-1)}$.

NOTE 3: Annex A of ISO/IEC 18032 [i.5] contains a table of error probabilities for different probabilistic primality tests.

EXAMPLE: For a random number of 1 536 bits tested with three successful iterations of the Miller-Rabin test the probability that this number is not a prime is about 2^{-101}.

NOTE 4: A new modulus is produced for each user of the signature scheme even if different public exponents are used. In practice if the modulus and public exponents are produced as described above (i.e. random modulus and choosing the public exponent) the probability of producing the same modulus or secret exponent is negligible.

6.2.2.2 Key and parameter generation algorithm dsagen1

The primes *p* and *q* shall be generated as described in Appendix B of FIPS Publication 186-4 [2] with primality of an integer regarded as satisfied if the probability that it is composite is at most ErrProb (see table 9).

A secret key *x* shall be generated by applying a random number generation method satisfying the requirements trueran (see clause 8.2.2) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed.

The per-message secret number *k* (ephemeral key) shall be generated using one of these generation methods.

NOTE: Possible methods for this can be found in FIPS Publication 186-4 [2], annex B.

Each value of *x* and *k* shall depend on at least EntropyBits bits of true randomness or a seed of entropy SeedEntropy bits (see table 9).

6.2.2.3 Key and parameter generation algorithm ecgen1 for ecdsa-Fp

The prime numbers *p* and *n*, and the point *G* on $E(F_p)$ shall be selected such that the conditions in clause 6.1.2.3 are satisfied with primality of an integer regarded as satisfied if the probability that it is composite is at most ErrProb (see table 10).

NOTE 1: Possible methods to generate *p*, *n*, *E* and *P* are specified in FIPS Publication 186-4 [2] and specifically in clause D.1.

NOTE 2: Where an intentional choice of weak public parameters (subject to an unknown "insider" attack) seems to be possible, these parameters can be generated verifiably at random at least for the generation of the curve *E*. In RFC 5639 [5] and specifically in clause D.1, possible methods for this are described.

A secret key *x* shall be generated by applying a random number generation method satisfying the requirements trueran (see clause 8.2.2) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed.

The per-message secret number *k* (ephemeral key) shall be generated using a method satisfying the requirements trueran (see clause 8.2.2) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed.

Each value of x and k shall depend on at least EntropyBits bits of true randomness or a seed of entropy SeedEntropy bits (see table 10).

6.2.2.4 Key and parameter generation algorithm ecgen2 for ecdsa-F2m

The prime numbers m and n, the elliptic curve E over F_{2^m} and the point G on $E(F_{2^m})$ shall be selected so that the conditions in clause 6.1.2.4 are satisfied with primality of an integer regarded as satisfied if the probability that it is composite is at most Err Prob (see table 11).

NOTE 1: Possible methods to generate m, n, E and G are specified in ANSI X9.62 [6], ANSI X9.82 [i.8] (and specifically in clause D.2).

Where an intentional choice of public parameters (subject to an unknown "insider" attack) seems to be possible, these parameters should be generated verifiably at random at least for the generation of the curve E.

NOTE 2: In ANSI X9.82 [i.8] a possible method for this is described (compare clause D.2).

A secret key x shall be generated by applying a random number generation method satisfying the requirements trueran (see clause 8.2.2) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed.

The per-message secret number k (ephemeral key) shall be generated using a method satisfying the requirements trueran (see clause 8.2.2) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed, too.

Each value of x and k shall depend on at least EntropyBits bits of true randomness or a seed of entropy SeedEntropy bits (see table 11).

6.2.2.5 Key and parameter generation algorithm ecgen1 for ecgdsa-Fp

The parameter and key generation methods shall be the same as the ecdsa-Fp methods described in clause 6.2.2.3.

6.2.2.6 Key and parameter generation algorithm ecgen2 for ecgdsa-F2m

The parameter and key generation methods shall be the same as the ecdsa-F2m methods described in clause 6.2.2.4.

7 Signature suites

NOTE: The primary criteria for inclusion of an algorithm in the present document are:

- the algorithm is considered as secure;
- the algorithm is commonly used; and
- the algorithm can easily be referenced (for example by means of an OID).

7.1 General

NOTE 1: A cryptographic signature suite is a combination of message encoding functions including a hash function and a defined signature scheme using a standardized signature algorithm. A signature suite consists therefore of the following components:

- a message encoding method including the hash function; and
- a signature algorithm and its associated parameters.

NOTE 2: To allow signing of more or less arbitrary long messages, a signature suite uses a hash function, so that the signing/verification algorithms operate on a fixed-size hash of the message. An important issue is to tie the hash function to the signature scheme. Without this, the weakest available hash function can define the overall security level.

Due to possible interactions which can influence security of signatures, algorithms and parameters for secure signatures shall be used only in predefined combinations referred to as the signature suites.

If any of the components of a suite is modified, then the suite shall be modified accordingly.

The hash functions used in a signature suite shall be selected as specified in clause 5.1.

The message padding methods used in a signature suite shall be selected as specified in clause 7.2.

The signature algorithms used in a signature suite shall be selected as specified in clause 6.1.

The signature suites given in clause 7.3 should be used.

NOTE 3: Key generation is not part of the way to identify a signature suite and can change over time. Key generation methods are addressed in clause 6.2.

NOTE 4: Some key generation methods and some signature suites require generation of a (pseudo-) random number. The (pseudo-) random number generation method is not part of the way to identify a signature suite and can change over time. (Pseudo-) random number methods are addressed in clause 8.

7.2 Padding methods

NOTE 1: Padding is algorithm dependent and some algorithms need non-trivial (non-zero) padding. This is the case for the RSA algorithm. Signature algorithms with appendix require methods that encode a message into an integer message representative that will be the input for the signature primitive. This encoding method can be deterministic, for example a padding of a fixed string to the hash value computed from the message, but can be also probabilistic, incorporating a (randomly generated) salt value, which are converted to and from message representatives. Although these latter encodings are not true padding schemes, they are listed here.

NOTE 2: If the padding is deterministic then the signature algorithm output will be unique, whereas with a probabilistic scheme the outputs will differ from each other. A signature scheme with randomized padding can be proven secure in the random oracle model which gives more confidence in the scheme and the cryptographic resistance for a longer time frame.

The list of padding methods given in table 4 should be used.

Table 4: The list of recommended padding methods

Padding method entry index	Short padding function entry name	Random number generation method	Random generator parameters	References
4.01	emsa-pkcs1-v1.5	-	-	RFC 3447 [3], section 9.2
4.03	emsa-pss	trueran/pseuran	MinSaltEntropy	RFC 3447 [3], section 9.1

NOTE 3: Each padding method has a unique entry index represented by a string beginning with "4." followed by a two-digit entry number.

The emsa-pkcs1-v1.5 padding method may be used and, when used, shall be implemented as defined in RFC 3447 [3].

NOTE 4: Up to December 2013, no real attack on emsa-pkcs1-v1.5 has been published. Some attacks on this padding scheme were caused by implementation errors or side channels. These attacks are impossible in case the emsa-pss scheme is used.

The emsa-pss padding method should be used and, when used, shall be implemented as defined in RFC 3447 [3] section 9.1.

Each salt value should depend on at least MinSaltEntropy bits of true randomness or a seed of entropy at least MinSaltEntropy bits (see table 8).

NOTE 5: This rule implies that the salt length is at least MinSaltEntropy bits.

NOTE 6: One choice for the salt length is the bit length of the hash function used in the signature scheme. The default value is 20 Bytes.

NOTE 7: The emsa-pss padding method has been stable for a long time and is a good improvement to the deterministic and static emsa-pkcs1-v1.5 scheme. Therefore it is better suited for long term use.

NOTE 8: The emsa-pss padding method is parameterized by the choice of hash function and a mask generation function MGF, defined in PKCS#1 (RFC 3447 [3]). MGF is based always on the corresponding hash function used, e.g. SHA-256.

7.3 Recommended signature suites

NOTE 1: A signature suite is defined using the following three parameters:

1) a hash function;

2) a padding method; and

3) a signature algorithm and its associated parameters.

The signature suites listed in table 4.a should be used.

Table 4.a: List of recommended signature suites

Entry name of the signature suite	Entry name for the hash function	Entry name for the padding method	Entry name for the signature algorithm
sha512-with-ecdsa	sha512	no padding required	ecdsa-Fp or ecdsa-F2m
sha256-with-rsa	sha256	see note 2 in clause 7.2	rsa
rsa-pss with mgf1SHA-256Identifier	sha256	emsa-pss with mask generation mgf1SHA-256Identifier	rsa
sha256-with-ecdsa	sha256	no padding required	ecdsa-Fp or ecdsa-F2m
sha224-with-rsa	sha224	see note 2 in clause 7.2	rsa
sha224-with-ecdsa	sha224	no padding required	ecdsa-Fp or ecdsa-F2m
sha384-with-ecdsa	sha384	no padding required	ecdsa-Fp or ecdsa-F2m

sha512-with-ecdsa should be used for very long term signatures, otherwise sha256-with-rsa, rsa-pss with mgf1SHA-256Identifier or sha256-with-ecdsa should be used.

The signature suites listed in table 4.b, as they are based on DSA, should not be used unless it is the only alternative (see clause 6.1.2.2) in which case they may be used.

Table 4.b: List of alternative signature suites

Entry name of the signature suite	Entry name for the hash function	Entry name for the padding method	Entry name for the signature algorithm
sha224-with-dsa	sha224	no padding required	dsa
sha256-with-dsa	sha256	no padding required	dsa
sha384-with-dsa	sha384	no padding required	dsa
sha512-with-dsa	sha512	no padding required	dsa

8 Random number generation methods

8.1 General

NOTE 1: The key generation methods and some signature suites require the generation of a random number.

NOTE 2: The random number generation methods combined with the key generation methods ensure that the expected effort of guessing a cryptographic key is at least equivalent to guessing a random value that is EntropyBits bits resp. SeedEntropy bits long. This can be satisfied with respect to different demands like information theoretic versus just complexity theoretic security, backward secrecy and/or forward secrecy and so on. Clause 8.2 in particular specifies by which RNGs these demands can be satisfied.

8.2 Recommended random number generation methods

8.2.1 General

The random number generation methods listed in table 5 should be used.

Methods satisfying the requirements trueran (see clause 8.2.2) should be used for generating keys that are used more than once.

In the case of the one-time keys *k* for DSA, ECDSA and EC-GDSA methods satisfying pseuran (see clause 8.2.3) may be used.

Table 5: The list of recommended random number generation methods

Random generator entry index	Short random generator entry name	Random generator parameters	Adoption date
5.01	trueran	EntropyBits	01.01.2001
5.02	pseuran	SeedEntropy	01.01.2001

NOTE: Each random number generation method has a unique entry index represented by a string beginning with "5." followed by a two-digit entry number. The terms "trueran" and "pseuran" denote the requirements for NRNGs and DRNGs respectively (i.e. non-deterministic and deterministic random number generators).

8.2.2 Random generator requirements trueran

A random number generator satisfying trueran shall have a *physical* source of randomness.

NOTE 1: Non-physical NRNGs are excluded as the designer has no real control of the amount of the produced entropy.

NOTE 2: Thus a random number generator satisfying trueran is based on a physical primary entropy source and a cryptographic or mathematical post-processing of the output of the primary entropy source.

The recommendations for these components are as follows:

(TR1): There should be a stochastic model for the primary entropy source which is found consistent with thorough adapted tests of prototypes of the source.

(TR2): The primary entropy source should be subjected to an *adapted* statistical *online* test. The original output of the primary entropy source should be tested not the output of the post-treatment instead of that (there may be justified exceptions to this general rule).

NOTE 3: "Online" means that the test will detect loss of quality of the primary entropy source during operation sufficiently soon after such an event occurs and that there will then at once be suitable countermeasures (e.g. stop of the generator). "Adapted" means adapted to the statistical model of the primary entropy source.

The stochastic model and the tests should deliver an estimate for the amount of the produced entropy.

NOTE 4: The primary entropy source is regarded to be *good* if it produces at least 0,997 bit entropy per output bit. For a good primary entropy source no post-treatment is necessary.

(TR3): If the primary entropy source does not produces at least 0,997 bit entropy per output bit, a post-treatment should be employed which by some (necessarily compressing) techniques deliver an output of higher entropy per output bit. There shall be an appropriate stochastic model of the post-treatment as well which together with the stochastic model of the primary entropy source and the tests ensures that the output produces at least 0,997 bit entropy per output bit.

The following alternative set of recommendations may be used although it should not be applied:

(TR1"): There should be mathematical models for the primary entropy source and the post-treatment.

(TR2"): The primary entropy source should be subjected to an online test which will detect most defects of the noise source except for special unlikely events.

(TR3"): There is a post-treatment that under the assumption of the models (assuming that the primary entropy source works as expected) delivers an output of at least 0,997 bit entropy per output bit and that even in the case of a complete breakdown of the primary entropy source (after there has been accumulated enough entropy at the beginning) satisfies the requirements pseuran including condition (**PR3)** of clause 8.2.3.

NOTE 5: This alternative set of requirements is closer to ANSI X9.82 [i.8] while the first set is more similar to AIS 31. In both cases the major target is to achieve forward and backward secrecy. In the latter case this secrecy can be completely complexity theoretic under certain circumstances and security relies rather on the post-treatment than on the primary entropy source in contrast to the first case which delivers information theoretical forward and backward secrecy. With the second set of requirements in the situation of a readout or manipulation of the internal state also forward secrecy is not ensured.

NOTE 6: An example of a possible random number generator design based on a noisy diode is given in clause E.2 of ISO/IEC 18031 [16] although without the necessary details.

8.2.3 Random generator requirements pseuran

A random number generator satisfying pseuran is a *deterministic* generator satisfying the following conditions:

(PR1): The generator shall be initialized by a seed with an entropy of at least SeedEntropy bits; and

(PR2): Even with the knowledge of a partial output bits sequence of the generator and having all information about its initialization (and in the case of a *hybrid* generator also about the output of the additional entropy source) except for the seed, it shall be computationally infeasible to determine any other m bits of the output with a probability larger than Max (2^{-m},$2^{-\text{SeedEntropy}}$).

NOTE 1: The second condition in particular implies that there is no information ascertainable a priori as to the output bits and that neither the seed nor any internal state of the Deterministic Random Number Generator (DRNG) can be recovered from a subset of the output.

NOTE 2: (PR1) means (or even implies) that the seed is produced using a non-deterministic random number generator. This generator does not need to be a physical one. Nevertheless the recommendation below is made to achieve high security.

Trueran (in particular physical, see clause 8.2.1) generator should be used for seeding.

(PR1) does not exclude constructions in which the DRNG is seeded by a chain of RNGs as described in clause 9.3.2 of ISO/IEC 18031 [16]. However:

- the first deterministic RNG in this chain shall be seeded with the output of a non-deterministic RNG;
- in the output of the last generator in the chain at least EntropyBits bits shall be left over; and
- the whole system regarded as a DRNG (including operational freedom like numbers of cycles before the next seeding of links regarded as non-physical additional entropy source) shall satisfy the second condition (PR2).

With a known seed or a known internal state any future output of a DRNG can be calculated. Therefore:

- the seed shall be kept secret;
- seeding shall follow procedures similar to those for the generation of root keys;

- there shall be no backups of the seed or internal states of a pseuran generator;
- the internal state of the RNG shall be secured against any readout and any adversarial manipulation; and
- in situations in which such readout or manipulation of an internal state of the DRNG does not seem to be completely excluded a re-seeding or a seed-update shall be executed from time to time. If re-seeding is employed the security of the re-seeding process shall be as strong as that of the original seeding.

NOTE 3: The frequency of this procedure (i.e. the amount of entropy that is fed in per output bit) depends on the actual risk of such readouts or manipulations.

In addition to the two above mentioned conditions DRNGs should satisfy the following additional condition ensuring backward secrecy even in the case of a known internal state:

(PR3): Even with complete knowledge of an internal state it is computationally infeasible to determine any previous *m* output bits with a probability greater than Max(2^{-m},$2^{-\text{SeedEntropy}}$).

NOTE 4: AIS 20/31 [i.9] defines the classes DRG.3 and DRG.4 for DRNGs. Roughly said RNGs of class DRG.3 satisfy conditions **(PR1)** and **(PR2),** RNGs of class DRG.4 also satisfy **(PR3)**.

The following are examples of pseuran generators:

EXAMPLE 1: ANSI X9.17 [i.10] generator. This DRNG was designed to pseudo randomly generate keys and initialization vectors for use of DES. It uses the triple-DES algorithm with a fixed key to mix a 64-bit seed with the current date. Iterated encryption enables to generate as many output bits as needed. Condition **(PR3)** is not satisfied at least without any further assumptions about the clock input. Instead of triple-DES also other strong block ciphers can be used as building block of the generator.

EXAMPLE 2: Example E.4 in AIS 20/31 [i.9] is another deterministic RNG based on a variable strong block cipher which as well does not satisfy condition **(PR3).**

EXAMPLE 3: Hash_DRGB, HMAC_DRBG, CTR_DRBG (see NIST Special Publication SP 800-90A [i.11]).

EXAMPLE 4: RSA deterministic RNG and Blum-Blum-Shub Generator [i.11]. These RNGs are based on iterated exponentiation modulo a composite modulus. The advantage is to base the security on the intractability of number theoretic problem (respectively RSA and the factorization problem) but the main drawback is the poor efficiency in comparison with the other deterministic RNGs described above, the security of which is only heuristic.

The Dual Elliptic Curve Deterministic Random Bit Generator (RBG) (Dual_EC_DRBG) defined in NIST Special Publication SP 800-90A [i.11] is removed in the Revision 1 Draft due to a security flaw and shall not be used.

9 Recommended hash functions and key sizes versus time

NOTE 1: In this clause recommendations are provided regarding the use of hash functions given in clause 5 and the key sizes to be used with the algorithms mentioned in clause 6.

NOTE 2: This clause is structured as follows:

- Clause 9.1 explains the considerations on which the recommendations are based.
- In clause 9.2, hash functions versus time are recommended.
- In clause 9.3, key sizes versus time are recommended.

9.1 Basis for the recommendations

NOTE 1: The recommendations for algorithm and parameter strengths are characterized by taking a reasonable margin above minimum key lengths based on both extrapolations of current trends as well as estimations based on the necessary computing power needed to break a given algorithm. Such extrapolations can be found in the literature, e.g. in the ENISA 2013 Recommendation [i.1].

NOTE 2: There are no rigorous security proofs for the components of signature schemes (hash function, signature algorithm, RNG), basically all security statements rely on results about the most effective attacks known at the time of writing of the present document. The possibility of a complete break of such a component (like, e.g. a fast universal factorization algorithm against RSA) that renders it useless can theoretically not completely be excluded but "breakthroughs" of that kind are regarded as improbable. In contrast to that certain unforeseen advances of moderate degree in analyzing cryptographic algorithms are regarded as a realistic threat: An example is given by the collision attacks on SHA-1 which demonstrated that this hash function is actually much weaker against collision attacks than predicted. The security margin for the recommendations below is chosen so that advances of this level are expected to be compensated without changing the parameters.

NOTE 3: Stability of the requirements in the present document is highly desirable for reasons of planning reliability. This means that if in e.g. 2013 a key length y is declared as sufficient for resistance during 6 years, i.e. at least until the end of 2019, an updated version in e.g. 2015 normally still declares this key length y as sufficient at least until the end of 2019. The following tables contain recommendations for the lifetime of keys and were chosen accordingly. The recommendations for a whole decade ("during 10 years") are explicitly declared "speculative" because of the uncertainty of predictions over such long periods. That means that the described principle of stability may not apply to these recommendations.

NOTE 4: The concept of a "liberal view" appearing in the previous versions of the document is removed for the near time frames, because significant differences did not appear therein.

NOTE 5: An attempt was made to achieve roughly similar security for all the components. For example the security level demanded by the tables is very roughly equivalent to 80 bits symmetric keys for an intended use in the short term and 100 bits for an intended use in the medium and longer term.

9.2 Recommended hash functions versus time

Hash functions in table 6 should be used for a resistance during X years.

Table 6: Recommended hash functions for a resistance during X years

Entry name of the hash function	1 year	3 years	6 years	10 years (speculative)
sha224	usable	usable	usable	unusable
sha256	usable	usable	usable	unknown
sha384	usable	usable	usable	usable
sha512	usable	usable	usable	Usable

LEGEND:
"usable": The algorithm with the given security parameters can be considered secure at the given time.
"unknown": The security of the algorithm is unknown; it may become secure if additional measures are applied.
"unusable": The algorithm cannot be considered secure for any kind of use in the context of secure signatures.

NOTE 1: "3 years" in table 7 means "until the end of 2017" and so on. To keep the tables for further revisions unchanged, the time interval notion is used instead of fixed dates.
NOTE 2: The listed hash functions are expected to be 2nd pre-image resistant and pre-image resistant for a longer period of time.

9.3 Recommended key sizes versus time

The parameters and schemes defined in tables 7 to 11 should be used.

Table 7: Recommended parameters for RSA and rsagen1 for a resistance during X years

Parameter	1 year	3 years	6 years	10 years (speculative)
MinModLen	1 536	2 048	3 072	4 096
ErrProb	2^{-80}	2^{-100}	2^{-100}	2^{-120}
SeedEntropy/EntropyBits	80	100	100	?

NOTE 1: A recommendation for RSA of the form "MinModLen=*y* for a resistance during 3 years" means "MinModLen should be *y* for RSA keys with an intended life time of 3 years (i.e. until end of 2017)". To keep the tables for further revisions unchanged, the time interval notion is used instead of fixed dates.

NOTE 2: A recommendation for EC-(G)DSA of the form "qMinLen=*y* for a resistance during 6 years" means "qMinLen should be *y* for keys (and curve parameters) with an intended life time of six years (i.e. until the end of 2020)".

NOTE 3: The meaning of all the listed parameters is explained in the respective sub-clauses of clause 6.

NOTE 4: There exist implementations in hardware for which a few of the 2 048 base number bits can be reserved for some other information such that the maximum modulus length is in fact 1 976 bits for these implementations. Since the loss of security seems to be negligible a MinModLen of 1976 provides the same security level.

NOTE 5: Up to June 2014, no real attacks on RSA-1024 are reported. The factorization of RSA-768 dated on December 2010 was done using several hundred PCs over two years of calendar time. Therefore a freshly generated 1 024 bits RSA key certified for an end user and valid for extremely short term, e.g. one month, can be considered as secure too. Nevertheless it can be necessary to apply additional cryptographic measures urgently during the validity time of such end user certificates, including their revocation, if the algorithm RSA-1024 becomes really weak, e.g. if the next RSA challenges RSA-896 (see http://www.rsa.com/rsalabs) is factored.

NOTE 6: In the gap left between the recommendations for one and three years it is on the discretion of the issuing CA to use either the first or the second recommendation. Depending on the measures foreseen by the CA in case of cryptographic weakness it can even be still appropriate to use RSA-1024 keys for end user certificates for a short term, if the CA is prepared to revoke all these certificates.

Table 8: Recommended padding schemes and values for MinSaltEntropy for a resistance during X years

Entry name of the padding scheme	1 year	3 years	6 years	10 years (speculative)
PKCS#1-v1.5	usable	usable	usable	unusable
PKCS#1-PSS	usable/ MinSaltEntropy=64	usable/ MinSaltEntropy=64	usable/ MinSaltEntropy=64	usable/ MinSaltEntropy=64

Table 9: Recommended parameters for DSA and dsagen1 for a resistance during X years

Parameter	1 year	3 years	6 years	10 years (speculative)
pMinLen	2 048	3 072	3 072	4 096
qMinLen	224	256	256	256
ErrProb	2^{-80}	2^{-80}	2^{-100}	2^{-100}
SeedEntropy/EntropyBits	80	80	100	100

If it is not required by a signature length restriction qMinLen=256 should be used, such that DSA can be used with SHA-256 and without truncation.

Table 10: Recommended parameters for ecdsa-Fp and ecgen1 for a resistance during X years

Parameter	1 year	3 years	6 years	10 years (speculative)
pMinLen	-	-	-	-
nMinLen	224	256	256	256
r0Min	104	104	104	?
MinClass	200	200	200	?
ErrProb	2^{-100}	2^{-100}	2^{-100}	2^{-100}
SeedEntropy/EntropyBits	100	120	120	256

Table 11: Recommended parameters for ecdsa-F2m and ecgen2 for a resistance during X years

Parameter	1 year	3 years	6 years	10 years (speculative)
pMinLen	-	-	-	-
nMinLen	224	256	256	256
r0Min	10^4	10^4	10^4	10^6
MinClass	200	200	200	200
ErrProb	2^{-80}	2^{-100}	2^{-120}	2^{-100}
SeedEntropy/EntropyBits	80	100	120	256

NOTE 7: Table 12 summarizes the recommendations from tables 7 to 11.

Table 12: Recommended signature suites for a resistance during X years

Entry name of the signature suite	1 year	3 years	6 years	10 years
sha256-with-rsa	1 536	2 048	2 048	not recommended
RSASSA-PSS with mgf1SHA-1Identifier	1 536	not recommended		
RSASSA-PSS with mgf1SHA-224Identifier	1 536	2 048	2 048	not recommended
RSASSA-PSS with mgf1SHA-256Identifier	1 536	2 048	2 048	3 072
sha224-with-ecdsa	224	224	not recommended	
sha256-with-ecdsa	256	256	256	256
sha256-with-dsa	256	256	256	256

NOTE 8: Because sha224-with-rsa has no security or performance advantages or disadvantages compared with sha256-with-rsa it is not listed here for interoperability reasons only.

NOTE 9: The generation of pseudo random masks used in PSS formatting requires the mixing properties of the MGF (mask generation function), which is not affected by the collision attacks on SHA-1.

NOTE 10: Table 13 provides the absolute dates for the recommendations from table 12.

Table 13: Recommended signature suites for a resistance up to X

Entry name of the signature suite	2015	2017	2020	2030
sha256-with-rsa	1 536	2 048	2 048	not recommended
rsa-pss with mgf1SHA-1Identifier	1 536	not recommended		
rsa-pss with mgf1SHA-224Identifier	1 536	2 048	2 048	not recommended
rsa-pss with mgf1SHA-256Identifier	1 536	2 048	2 048	3 072
sha224-with-ecdsa	224	224	not recommended	
sha256-with-ecdsa	256	256	256	256
sha256-with-dsa	256	256	256	256

10 Time period resistance of hash functions and keys

10.1 General notes

NOTE 1: The hash functions and signature algorithms defined in the present document are suitable to be used in the context of advanced electronic signatures TS 101 733 [i.12], TS 101 903 [i.13] and TS 102 778 [i.14].

NOTE 2: The time period for the resistance of a given key depends on the usage of the key. To this respect different use cases are addressed. Once the time period is known, then the figures provided in clause 9 can be used to know the appropriate key size.

10.2 Time period resistance for hash functions

Hash functions should resist as long as a signature verification still needs to be done.

If not, a specific signature maintenance process shall be performed (see annex C for more information).

A hash function used to compute the hash of a certificate, which is not a self-signed certificate, should resist during the validity period of that certificate.

A hash function used to compute the hash of a self-signed certificate shall resist during the validity period of that self-signed certificate.

NOTE 1: A hash function used to compute the imprint of a message placed in a time-stamp token is not used in combination of a signature scheme. The length of its output is not dependent upon the size of the parameters of the signature scheme.

A hash function used to compute the imprint of a message placed in a time-stamp token should resist at least 10 years.

NOTE 2: If the signature suite that has been used by the signer is also presumed to be resistant over this time period, then the signature maintenance process can be minimized.

10.3 Time period resistance for signer's key

NOTE 1: The focus is very often placed on the resistance of signer's keys.

Signer's keys shall resist during the validity period (from `notBefore` to `notAfter`) of the associated certificate.

NOTE 2: If they become weak due to progress in cryptographic research, revocation will be necessary, and there would be a large burden to re-issue new keys and certificates. However, there is no security breach after revocation.

NOTE 3: If a signer's key does not resist during the validity period of its associated certificate, then the protection provided through the use of time-stamping is sufficient to provide an adequate protection.

10.4 Time period resistance for trust anchors

A trust anchor should remain secure during the whole time period during which advanced electronic signature TS 101 733 [i.12], TS 101 903 [i.13] and TS 102 778 [i.14] needs to be verified.

NOTE 1: This can be longer than the life time of the associated certificate. If it becomes weak, it cannot be used anymore for immediate verifications. It can be used for subsequent verifications, if a specific maintenance process is performed before the trust anchor becomes insecure.

NOTE 2: This is an important difference to the estimation of the time period resistance for signer's key.

10.5 Time period resistance for other keys

All other keys (TSU keys, CA keys, CRL issuer keys, OCSP responder keys) should resist during the validity period of the associated certificate and the certificates that rely on its validity.

Their security parameters should then be chosen stronger than the corresponding parameters of the certified keys.

If they do not resist the foreseen time period, a maintenance process should be applied before the algorithm is broken.

For these keys the same rule as for trust anchors in clause 10.4 applies.

11 Practical ways to identify hash functions and signature algorithms

11.1 General

Hash functions and signatures algorithms shall be referenced using an OID and/or a URN.

NOTE 1: Only the owner of the OID or the URN is allowed to define its meaning and thus the meaning of the algorithm, usually referencing another document.

NOTE 2: If such an OID/URN is not available the algorithm is unusable.

11.2 Hash functions and signature algorithms objects identified using OIDs

11.2.1 Hash functions

The hash functions shall be identified using the OIDs in table 14.

Table 14

Short object name	OID	References
id-sha1	{ iso(1) identifiedOrganization(3) oIW(14) oIWSecSig(3) oIWSecAlgorithm(2) 26 }	RFC 3279 [7]
id-sha224	{ joint-iso-itu-t(2)country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) sha224(4) }	RFC 4055 [8]
id-sha256	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) 1 }	RFC 4055 [8]
id-sha384	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) 2 }	RFC 4055 [8]
id-sha512	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) 3 }	RFC 4055 [8]

11.2.2 Signature algorithms

The signature algorithms shall be identified using the OIDs in table 15.

Table 15

Short object name	OID	References
rsaEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }	RFC 3279 [7]
id-dsa	{ iso(1) member-body(2) us(840) x9-57(10040) x9cm(4) 1 }	RFC 3279 [7]
id-ecPublicKey	{ iso(1) member-body(2) us(840) 10045 2 1 }	RFC 5753 [9]

11.2.3 Signature suites

The signature suites shall be identified using the OIDs in table 16.

Table 16

Short object name	OID	References
sha256WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 11 }	RFC 4055 [8]
sha512WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 13 }	RFC 4055 [8]
id-RSASSA-PSS with mgf1SHA-224Identifier	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }	RFC 4055 [8]
id-RSASSA-PSS with mgf1SHA-256Identifier	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }	RFC 4055 [8]
id-dsa-with-sha224	{ joint-iso-ccitt(2) country(16) us(840) organization(1) gov(101) csor(3) algorithms(4) id-dsa-with-sha2(3) 1 }	not recommended
id-dsa-with-sha256	{ joint-iso-ccitt(2) country(16) us(840) organization(1) gov(101) csor(3) algorithms(4) id-dsa-with-sha2(3) 2 }	not recommended
ecdsa-with-Recommended	{ iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) recommended(2) }	ANSI X9.62 [6]
ecdsa-with-Sha224	{ iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) specified(3) 1 }	ANSI X9.62 [6]
ecdsa-with-Sha256	{ iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) specified(3) 2 }	ANSI X9.62 [6]
ecdsa-with-Sha384	{ iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) specified(3) 3 }	ANSI X9.62 [6]
ecdsa-with-Sha512	{ iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) specified(3) 4 }	ANSI X9.62 [6]

11.3 Hash functions and signature algorithms identified objects using URNs

11.3.1 Hash functions

The hash functions shall be identified using the URNs in table 17.

Table 17

Short object name	URN	References
sha224	http://www.w3.org/2001/04/xmldsig-more#sha224	RFC 6931 [10]
sha256	http://www.w3.org/2001/04/xmlenc#sha256	W3C Recommendation XML Encryption Syntax and Processing, April 2013 [11]
sha384	http://www.w3.org/2001/04/xmldsig-more#sha384	RFC 6931 [10]
sha512	http://www.w3.org/2001/04/xmlenc#sha512	W3C Recommendation XML Encryption Syntax and Processing, April 2013 [11]

11.3.2 Signature algorithms

NOTE: There is no need to define such URNs since XAdES uses the signature algorithms contained in X.509 certificates which are referenced using OIDs.

11.3.3 Signature suites

The signature suites shall be identified using the URNs in table 18.

Table 18

Short object name	URN	References
rsa-sha256	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	RFC 6931 [10]
rsa-sha384	http://www.w3.org/2001/04/xmldsig-more#rsa-sha384	RFC 6931 [10]
rsa-sha512	http://www.w3.org/2001/04/xmldsig-more#rsa-sha512	RFC 6931 [10]
rsapss-with-parameters	http://www.w3.org/2007/05/xmldsig-more#rsa-pss	RFC 6931 [10]
rsapss-with-defaults-sha224	http://www.w3.org/2007/05/xmldsig-more#sha224-rsa-MGF1	RFC 6931 [10]
rsapss-with-defaults-sha256	http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1	RFC 6931 [10]
rsapss-with-defaults-sha384	http://www.w3.org/2007/05/xmldsig-more#sha384-rsa-MGF1	RFC 6931 [10]
rsapss-with-defaults-sha512	http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1	RFC 6931 [10]
ecdsa-sha224	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha224	RFC 6931 [10]
ecdsa-sha256	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256	RFC 6931 [10]
ecdsa-sha384	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384	RFC 6931 [10]
ecdsa-sha512	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512	RFC 6931 [10]

11.4 Recommended hash functions and signature algorithms objects that do not yet have an OID or a description

NOTE: The following URNs are already assigned, but the SHA-3 standardization is not finished yet.

Table 19

Short object name	URN	References
rsapss-with-sha3-224	http://www.w3.org/2007/05/xmldsig-more#sha3-224-rsa-MGF1	RFC 6931 [10]
rsapss-with-sha3-256	http://www.w3.org/2007/05/xmldsig-more#sha3-256-rsa-MGF1	RFC 6931 [10]
rsapss-with-sha3-384	http://www.w3.org/2007/05/xmldsig-more#sha3-384-rsa-MGF1	RFC 6931 [10]
rsapss-with-sha3-512	http://www.w3.org/2007/05/xmldsig-more#sha3-512-rsa-MGF1	RFC 6931 [10]

Annex A (normative): Algorithms for various data structures

ETSI TS 101 733 [i.12], ETSI TS 101 903 [i.13] and ETSI TS 102 778 [i.14] define the formats of advanced electronic signatures. These three documents reference other documents defining various standardized data structures.

These other documents or companion documents define the algorithms which can be supported by the issuers of the data structures and the algorithms which will (for interoperability purposes) and can be supported by the users of the data structures.

- Signer Certificates (RFC 5280 [i.15] and RFC 3279 [7]).
- Certificate Revocation Lists (RFC 5280 [i.i.15] and RFC 3279 [7]).
- OCSP responses (RFC 6960 [13]).
- Certification Authority Certificates (RFC 5280 [i.i.15] and RFC 3279 [7]).
- Self-signed certificates for CA certificates (RFC 5280 [i.15] and RFC 3279 [7]).
- Time-Stamping Tokens (TSTs) (RFC 3161 [12] and ETSI TS 101 861 [15]).
- Time-Stamping Unit certificates (RFC 3161 [12] and ETSI TS 101 861 [15]).
- Self-signed certificates for TSU Certificates (RFC 5280 [i.15] and RFC 3279 [7]).
- Attribute Certificates (ACs) (RFC 5280 [i.15] and RFC 3279 [7]).

For each data structure, the set of algorithms to be used is specified.

Since many of these documents have been published some years ago, they cannot be all up to date with the latest cryptographic advancements. In particular, some of the algorithms specified in the above documents exhibit weaknesses or, worse, are now broken.

For that reason, when it is the case, algorithms that were initially recommended and that shall or should not be used anymore will be indicated.

In the same way, more recent algorithms do not appear in these documents. This does not mean that they should not be used, but that at this time they do not yet fall into the "shall" or "should" categories.

The algorithms which may be supported by issuers or users are not indicated.

A.1 CAdES and PAdES

A CMS based digital signature (ETSI TS 101 733 [i.12] and ETSI TS 102 778 [i.14]) contains an identifier of the hash function that has been used (contained in the `digestAlgorithm` element from the `SignerInfo` data structure) and an identifier of the signature algorithm that has been used (contained in the `signatureAlgorithm` element from the `SignerInfo` data structure) which will be consistent with the identifier of the signature algorithm contained in the signer's certificate.

Requirements in table A.1 apply to CAdES [i.12] and PAdES [i.14]. They apply both to the hash function and the signature algorithm.

Table A.1

CAdES [i.12] and PAdES [i.14]	***Issuers* of AdES**	***Users* of AdES**
Hash functions	shall support sha256 shall not use sha1	shall support sha1 shall support sha256
Signature algorithms	should support RSA or DSA or ECDSA	shall support RSA should support DSA should support ECDSA

NOTE: Because the usage of SHA-224 as hash functions gives no advantage compared with SHA-256 neither in security nor in performance there is no requirement for SHA-224 support as a hash function.

A.2 XAdES

ETSI TS 101 903 [i.13] uses a URN to reference the hash function in the ds:DigestMethod element. Since ETSI TS 101 903 [i.13] is built upon XML DigSig, the algorithm requirements from XML DigSig [14] shall apply with the amendments defined in table A.2.

Table A.2: Hash functions and signature algorithms for XAdES

XAdES [i.13]	***Issuers* of AdES**	***Users* of AdES**
Hash functions	shall support sha256 shall not use sha1	shall support sha1 shall support sha256
Signature algorithms	should support RSA or DSA or ECDSA	shall support RSA should support DSA should support ECDSA

For canonicalization:

1) the following Canonical XML (omits comments) [i.17] should be used: http://www.w3.org/TR/2001/REC-xml-c14n-20010315

2) the following Canonical XML with Comments [i.18] may be used: http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments

A.3 Signer's certificates

A signer certificate contains a subject public key and is signed by a CA issuing key. RFC 5280 [i.15] does not require to use any particular cryptographic algorithms. However, RFC 3279 [7] does. The requirements in RFC 3279 [7] shall apply to signer public keys and CA issuing keys with the amendments defined in table A.3.

Table A.3: Algorithms for signer public keys and CA issuing keys

Signer certificates	***Issuers* of signer certificates**	***Users* of signer certificates**
Signer public keys	should support RSA or ECDSA	shall support RSA should support DSA should support ECDSA
CA issuing keys	shall support RSA with SHA-256 or ECDSA with SHA-256	should support RSA with SHA-1 should support DSA with SHA-1 should support ECDSA with SHA-224 shall support RSA with SHA-256 should support DSA with SHA-256 shall support ECDSA with SHA-256

NOTE: Because the usage of SHA-224 with RSA and DSA gives no advantage compared with SHA-256 neither in security nor in performance there is no requirement on SHA-224 support with these algorithms.

With RSA and DSA the hash functions SHA-256 and SHA-512 should be used instead of SHA-224 or SHA-384.

A.4 CRLs

A CRL is signed by a CRL Issuer. RFC 5280 [i.15] does not require to use any particular cryptographic algorithms. However, RFC 3279 [7] does. The requirements defined in RFC 3279 [7] shall apply to CRL Issuer public keys with the amendments defined in table A.4.

Table A.4: Algorithms for CRL issuer public keys

CRLs	*Issuers* of CRLs	*Users* of CRLs
CRL issuer keys	shall support RSA with SHA-256	should support RSA with SHA-1 should support DSA with SHA-1 should support ECDSA with SHA-1 should support ECDSA with SHA-224 shall support RSA with SHA-256 should support DSA with SHA-256 shall support ECDSA with SHA-256

NOTE: Because the usage of SHA-224 with RSA and DSA gives no advantage compared with SHA-256 neither in security nor in performance there is no requirement on SHA-224 support with these algorithms.

With RSA and DSA the hash functions SHA-256 and SHA-512 should be used instead of SHA-224 or SHA-384.

A.5 OCSP responses

A CRL is signed by an OCSP responder. The algorithm requirements from RFC 6960 [13] clause 4.3 shall apply with the amendments defined in table A.5. These requirements shall apply to the hash algorithm and the signature algorithm used by OCSP responders.

Table A.5: Algorithms for OCSP responders

OCSP response	*Issuers* of OCSP responses	*Users* of OCSP response
OCSP responder keys	should support sha1 with dsa should support sha1 with rsa should support sha256 with rsa	should support sha1 with dsa shall support sha1 with rsa shall support sha256 with rsa should support sha256 with dsa should support ECDSA with SHA-256

A.6 CA certificates

A CA certificate contains a CA public key and is signed by a CA private key. For CA public keys (as subject) and CA public keys (as issuer), the algorithm requirements from RFC 3279 [7] shall apply with the amendments defined in table A.6.

Table A.6: Algorithms for certification authorities

CA certificates	*Issuers* of CA certificates	*Users* of CA certificates
Subject CA public key	should support RSA with SHA-256	should support RSA with SHA-1 shall support RSA with SHA-256 should support DSA with SHA-256 should support ECDSA with SHA-224 should support ECDSA with SHA-256
Issuer CA public keys	should support RSA with SHA-256	should support RSA with SHA-1 should support ECDSA with SHA-224 shall support RSA with SHA-256 should support DSA with SHA-256 should support ECDSA with SHA-256

NOTE: Because the usage of SHA-224 with RSA and DSA gives no advantage compared with SHA-256 neither in security nor in performance there is no requirement on SHA-224 support with these algorithms.

With RSA and DSA SHA-256 and SHA-512 should be used instead of SHA-224 or SHA-384.

A.7 Self-signed certificates for CA issuing CA certificates

A self-signed certificate contains a single root CA public key. For root CA public keys, the algorithm requirements from RFC 3279 [7] shall apply with the amendments defined in table A.7.

NOTE: Self-signed certificates need to resist quite long (e.g. more than 10 years).

Table A.7: Algorithms for self-signed certificates

Self-signed certificates	*Issuers* of self-signed certificates	*Users* of self-signed certificates
Root CA public keys	shall support RSA with SHA-256 should support ECDSA with SHA-256	shall support RSA with SHA-1 shall support RSA with SHA-256 should support DSA with SHA-256 should support ECDSA with SHA-256

A.8 TSTs based on RFC 3161

The following requirements apply to hash functions and TST signature algorithms. The algorithm requirements from RFC 3161 [12] shall apply with the amendments defined in table A.8.

Table A.8: Algorithms for time stamps

Time-Stamping Tokens	TST requesters	TST issuers	TST verifiers
Hash function	should support sha1 should support sha256	should support sha1 shall support sha256	should support sha1 shall support sha256
TST signature algorithms	should support sha1 with rsa shall support sha256 with rsa	should support sha1 with rsa shall support sha256 with rsa	should support sha1 with rsa shall support sha256 with rsa should support ECDSA with SHA-256

A.9 TSU certificates

A TSU certificate contains a TSU public key and is signed by a CA private key. For TSU public keys (as subject) and CA public keys (as issuer), the algorithm requirements from RFC 3279 [7] shall apply with the amendments defined in table A.9.

Table A.9: Algorithms for time stamping units

TSU certificates	*Issuers* of TSU certificates	*Users* of TSU certificates
TSU public key	should support RSA with SHA-256 should support ECDSA with SHA-256	should support RSA with SHA-1 shall support RSA with SHA-256 should support ECDSA with SHA-256
Issuer CA public keys	should support RSA with SHA-256 should support ECDSA with SHA-256	should support RSA with SHA-1 should support DSA with SHA-256 shall support RSA with SHA-256 should support ECDSA with SHA-256

A.10 Self-signed certificates for CAs issuing TSU certificates

A self-signed certificate contains a single root CA public key. For self-signed certificates for CAs issuing TSU certificates, the algorithm requirements from RFC 3279 [7] shall apply with the amendments defined in table A.7 (see clause A.7).

Annex B (informative): Recommended key sizes (historical)

This annex will later on contain the outdated tables provided in clause 9 so that a history about previously recommended hash functions and key sizes can be easily done at a given time and for a given time period.

Annex C (informative): Signature maintenance

An advanced electronic signature ETSI TS 101 733 [i.12], ETSI TS 101 903 [i.13] and ETSI TS 102 778 [i.14] can be verified according to a signature policy that meets the business needs.

A signature policy can include constraints about which algorithms and key lengths are deemed appropriate under that policy and/or define a time beyond which the algorithms/keys related to an advanced electronic signature should not be trusted anymore, unless additional security measures are taken.

It can be needed to re-verify advanced electronic signatures (this is called a subsequent verification) well beyond the time they were initially verified. At the time of re-verification, trust anchors and algorithms that were initially defined in the signature policy can be not secure anymore. Additional security measures need to be taken so that this can be done.

It can also happen that some keys were secure at the time the initial verification of an advanced electronic signature was performed, but due to some "accident" this is no more the case later on (e.g. due to a key compromise).

In both cases, it is possible to maintain the security of an advanced electronic signature which has already been successfully verified. This can be done with security measures such as:

- the secure archival of both the definition of the signature policy (or an unambiguous reference to it) and all the data initially used to verify the advanced electronic signature according to that signature policy; or

- the secure archival of both the definition of the signature policy and the addition to the advanced electronic signature of other data (e.g. time-stamps) that will allow subsequent verifications.

These measures can be defined in the signature policy itself or "elsewhere" in a set of rules called a "signature maintenance policy" which will allow maintenance of the validity of advanced electronic signatures.

At a time where it is possible or likely that the algorithms and key lengths originally used will not be secure anymore, an application of a signature maintenance process allows nevertheless to re-verify advanced electronic signatures under a given signature policy. The sooner the process is applied, the better.

History

Document history		
V1.1.1	March 2003	Publication as SR 002 176
V1.2.1	July 2005	Publication as TS 102 176-1
V2.0.0	November 2007	Publication as TS 102 176-1
V2.1.1	July 2011	Publication as TS 102 176-1
V1.1.1	November 2014	Publication