ETSI TS 102 176-1 V1.2.1 (2005-07)

Technical Specification

**Electronic Signatures and Infrastructures (ESI);
Algorithms and Parameters for Secure Electronic Signatures;
Part 1: Hash functions and asymmetric algorithms**

Reference

RTS/ESI-000039-1

Keywords

e-commerce, electronic signature, security

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

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

The present document is part 1 of a multi-part deliverable covering the Algorithms and Parameters for Secure Electronic Signatures, as identified below:

Part 1: "Hash functions and asymmetric algorithms";

Part 2: "Secure channel protocols and algorithms for signature creation devices".

Introduction

The present document provides for security and interoperability for the application of the underlying mathematical algorithms and related parameters for electronic signatures in accordance with the Directive 1999/93/EC [1] of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

On the other side the present document is not a legal document answering the question which key lengths or use dates are sufficient to ensure a certain level of liability. In particular the reader is warned that some national signature laws or regulations may demand a higher level of security for qualified electronic signatures than recommended here by the key lengths and use dates in the present document.

The present document defines a list of hash functions, as well as a list of signature schemes together with the requirements on their parameters, as well as the recommended combinations of these schemes with hash functions and padding method in the form of "signature suites" to be used with the data structures defined in the documents developed under the EESSI (European Electronic Signature Standardization Initiative). The present document contains several informative annexes which provide useful information on a number of subjects mentioned in the text.

The present document is not a general purpose document dealing with hash functions and asymmetrical algorithms in general. The goal of the present document is not to list all "good" signature algorithms but those that are most important to be used in the context of advanced electronic signatures. In addition, the intent of the present document is not to have a catalog of all algorithms suitable for advanced electronic signatures, but to limit the list to a reasonable set so that interoperability can be achieved. Interoperability with security is the main issue.

The primary criteria for inclusion of an algorithm in the document is "Secure, widely used and deployed in practice". Whereas all listed algorithms have been checked for security by cryptographic experts, it cannot be concluded from the document, that an algorithm not listed would be insecure.

The second part of this technical standard (protocols and algorithms for SCDev secure channels) defines protocols and symmetric algorithms that may optionally be used to construct a secure channel providing either only integrity or both integrity and confidentiality between an application and a signature creation device (SCDev). Such a secure channel may be used during the operational phase of a signature creation device:

- when the key pair is not generated by the SCDev, to remotely download in the SCDev both a private key and the associated public key certificate;
- when the key pair is generated by the SCDev, to remotely download in the SCDev a public key certificate and associate it with the previously generated private key.

1 Scope

The present document is targeted to support advanced electronic signatures and the related infrastructure.

The present document defines a list of hash functions and a list of signature schemes, as well as the recommended combinations of hash functions and signatures schemes in the form of "signature suites".

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).

This does not mean that other hash functions and signature suites cannot be used, but either they do not correspond to the above criteria or their security have not been assessed.

The 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 electronic signatures. For each data structure, the set of algorithms to be used are specified. Each set is identified by an identifier which is either an OID (Object IDentifier) or a URI /URN.

The use of such identifiers is necessary so that interoperability can be achieved. In order to allow for data interchange, the document references algorithms in terms of OIDs and URIs / URNs together with algorithm parameters.

Different requirements apply to the *issuers* and to the *users* of the data structures in order to allow for interoperability.

RFCs documents use the terms SHALL, SHOULD, MAY, RECOMMENDED in order to allow for interoperability. The same terminology is used in the present document (see RFC 2119 [25]).

Issuers of the data structures (e.g. CSPs, CRL Issuers, OCSP responders, TSUs) need to know the algorithms and key sizes they SHOULD or MAY support. There SHOULD be at least one algorithm recommended to support, but may be more than one.

Users of the data structures (i.e. signers or verifiers of electronic signatures) need to know the algorithms and key sizes they SHALL, SHOULD or MAY support. For *users* and for each data structure, there must be at least one algorithm to support, but may be more than one.

These requirements are listed in annex A.

Annex B provides historical information on the recommended hash functions, algorithms and key sizes for the generation and verification of electronic signatures. This annex will be periodically updated.

Annex C provides more information on the generation of RSA modulus.

Annex D provides more information on the generation of elliptic curve domain parameters.

Annex E addresses the generation of random data.

Annex F lists the algorithm identifiers defined in various documents.

Annex G provides a short abstract of ISO/IEC 10118-3 [3] and ISO/IEC 9796-2 [17].

Annex H provides some guidance on signature maintenance.

Annex I lists the major changes from the previous versions.

The present document defines a set of algorithms (i.e. hash functions, signature schemes and signature suites) and the corresponding parameters that are recommended to be used. If such algorithms are used according to the context where they are expected to be used, then a reasonable security level can be assumed.

The algorithms defined in the present document are usable in particular with the following documents:

- TS 101 733 [18]: "Electronic Signatures and Infrastructures (ESI); Electronic Signature Formats";
- TS 101 903 [19]: "XML Advanced Electronic Signatures (XAdES)";

NOTE: XML language is defined in RFC 3275 [10].

- TS 101 861 [20]: "Time stamping profile";
- TS 101 456 [33]: "Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing qualified certificates";
- TS 102 042 [34]: "Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing public key certificates";
- CWA 14169 [35]: "Secure Signature-Creation Devices "EAL 4+"";
- CWA 14170 [36]: "Security requirements for signature creation applications";
- CWA 14171 [37]: "Procedures for electronic signature verification";
- CWA 14167-1 [38]: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 1: System Security Requirements";
- CWA 14167-2 [39]: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 2: Cryptographic module for CSP Signing Operations with Backup - Protection Profile";
- CWA 14167-3 [40]: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 3: Cryptographic module for CSP key generation services - Protection profile (CMCKG-PP)";
- CWA 14167-4 [41]: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 4: Cryptographic module for CSP signing operations - Protection profile - CMCSO PP";
- RFC 3280 [2]: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile";
- RFC 3281 [21]: "An Internet Attribute Certificate profile for authorization";
- RFC 3161 [9]: (2001): "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)";
- RFC 2560 [22]: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

Patent related issues are out of the scope of the present document.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

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

[1] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

[2] IETF RFC 3280 (2002): "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

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

NOTE: See annex G for main content description.

[4] FIPS Publication 180-2 (2002): "Secures Hash Standard".

NOTE: Change Notice to include SHA-224.

[5] IEEE P1363 (2000): "Standard Specifications for Public-Key Cryptography".

[6] FIPS Publication 186-2 (2000): "Digital Signature Standard (DSS)".

NOTE: With change notice from October 5, 2001.

[7] ANSI X9.62 (1998): "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)".

[8] ISO/IEC 15946-2 (2002): "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures".

NOTE: This IS is confirmed until the update of ISO/IEC 14888-3 is ready.

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

[10] IETF RFC 3275 (2002): "(Extensible Markup Language) XML-Signature Syntax and Processing".

[11] IETF RFC 3278 (2002): "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)".

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

[13] IETF RFC 3370 (2002): "Cryptographic Message Syntax (CMS) Algorithms".

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

[15] 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".

[16] DIN V 66291-1: "Chip cards with digital signature application/function according to SigG and SigV".

NOTE: See CWA 14890-1 section 13.3.1 for an English translation: "Application Interface for smart cards used as Secure Signature Creation Devices - Part 1: Basic requirements".

[17] ISO/IEC 9796-2: "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 2: Integer factorization based mechanisms".

NOTE: See annex G for main content description.

[18] ETSI TS 101 733: "Electronic Signatures and Infrastructures (ESI); Electronic Signature Formats".

[19] ETSI TS 101 903: "XML Advanced Electronic Signatures (XAdES)".

[20] ETSI TS 101 861: "Time stamping profile".

[21] IETF RFC 3281 (2002): "An Internet Attribute Certificate profile for Authorization".

[22] IETF RFC 2560 (1999): "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

[23] IETF RFC 3852 (2004): "Cryptographic Message Syntax (CMS)".

[24] IETF RFC 3270 (2002): "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services".

[25] IETF RFC 2119 (1997): "Key words for use in RFCs to Indicate Requirement Levels".

[26] ISIS-MTT Interoperability Specification (2004), TeleTrusT e.V. Deutschland, www.isis-mtt.de.

[27] IETF RFC 3874 (2005): "A 224-bit One-way Hash Function: SHA-224".

[28] IETF RFC 4050 (2005): "Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures".

[29] IETF RFC 4051 (2005): "Additional XML Security Uniform Resource Identifiers (URIs)".

[30] W3C Recommendation - 12 February 2002: "XML-Signature Syntax and Processing".

NOTE: http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

[31] W3C Recommendation - 10 December 2002: "XML Encryption Syntax and Processing".

NOTE: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[32] ISO/IEC 14888-3 (1999): "Information technology - Security techniques - Digital signatures with appendix - Part 3: Certificate-based mechanisms".

[33] ETSI TS 101 456: "Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing qualified certificates".

[34] ETSI TS 102 042: "Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing public key certificates".

[35] CWA 14169: "Secure Signature-Creation Devices "EAL 4+"".

[36] CWA 14170: "Security requirements for signature creation applications".

[37] CWA 14171: "Procedures for electronic signature verification".

[38] CWA 14167-1: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 1: System Security Requirements".

[39] CWA 14167-2: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 2: Cryptographic module for CSP Signing Operations with Backup - Protection Profile".

[40] CWA 14167-3: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 3: Cryptographic module for CSP key generation services - Protection profile (CMCKG-PP)".

[41] CWA 14167-4: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 4: Cryptographic module for CSP signing operations - Protection profile - CMCSO PP".

3 Definitions and abbreviations

3.1 Definitions

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

bit length: bit length of an integer *p* is *r* if $2^{r-1} \leq p < 2^r$

signature policy: set of rules for the creation and validation of an electronic signature, that defines the technical and procedural requirements for electronic 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

NOTE: The key generation algorithm generates the keys for the two others algorithms.

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

3.2 Abbreviations

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

AA	Attribute Authority
CRL	Certificate Revocation List
CRT	Chinese Remainder Theorem
CSP	Certification-Service-Provider
CWA	CEN Workshop Agreement
DRNG	Deterministic Random Number Generator
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
ECGDSA	Elliptic Curve German Digital Signature Algorithm
EESSI	European Electronic Signature Standardization Initiative
MGF	Mask Generation Function
NRNG	Non-deterministic Random Number Generator
OCSP	Online Certificate Status Protocol
OID	Object IDentifier
RfC	Request for Comments
RNG	Random Number Generator
RSA	Rivest, Shamir and Adleman algorithm
SAGE	Security Algorithms Group of Experts
SCDev	Secure Signature Creation Device
TST	Time-Stamp Token
TSU	Time-Stamping Unit
URI	Uniform Resource Identifier
URN	Uniform Resource Number

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 shall enable dynamic updating of the lists of recommended algorithms and signature suites. An initial list of recommended cryptographic hash functions and signature algorithms is given in the present document.

The maintenance activities will introduce new cryptographic hash functions and signature algorithms and will lead to remove cryptographic hash functions and signature algorithms from the list and need to respond to the following situations:

1) The need to introduce new algorithms and relevant parameters will call 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 an EESSI is based upon.

2) Advances in cryptography will 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 activity will be carried by ETSI.

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

5 Hash functions

5.1 General

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

NOTE: In the present document, "hash function" means a hash function with the three properties defined in this clause (i.e. clause 5.1).

Hash functions may be used in a variety of cases, such as:

- Advanced Electronic Signatures include the identifier of the hash function used to compute the digital signature.

- Time-Stamp tokens include the identifier of the hash algorithm used to compute the hash value for the time-stamped data.

- Public key certificates include the identifier of a signature suite which defines the hash function used to compute the digital signature.

For the purpose of generating signatures the following (informally defined) three properties are required from the hash function h.

1) **Pre-image resistance:** Given $y = h(m)$ (but not m) it is practically infeasible to find m. Without this property, a signature scheme may otherwise be vulnerable to an attack based on generating the signature "backwards", applying the verification function to a randomly chosen signature value.

2) **2nd pre-image resistance:** Given $h(m)$ and m, it is practically infeasible to find another $m' \neq m$ such that $h(m) = h(m')$. For signatures, this property protects from re-using an already existing signature for another message.

3) **Collision resistance:** It is practically infeasible to find any pair of distinct values m, m' such that $h(m) = h(m')$. This property is obviously needed to protect signature against chosen message attacks.

While one can construct examples of functions that are collision resistant, but not pre-image resistant, one would for practical purposes nevertheless expect that the above list of properties is ordered by difficulty for an attacker, i.e. breaking pre-image resistance is the most difficult. Recently some new attacks against hash function MD5 succeeded, it was shown that MD5 is not collision resistant by constructing classes of messages-pairs with the same hash value. Whereas the loss of collision resistance does not imply that a pre-image or second pre-image can easier be constructed, it is recommended to migrate to other hash functions, if the collision resistance becomes weaker.

In addition to this, more subtle properties are often required as a consequence of mathematical properties of the signature scheme itself. For instance, h should not preserve algebraic structure. The perhaps best known example is the multiplicativity of the (naive) RSA scheme, which would otherwise give a valid signature for a*b from two valid signatures of a and b.

The above properties have led to some signature schemes being defined and proven secure in the so-called Random Oracle Model, where one assume h "behaves" like a completely random function. Intuitively, a completely random function should have all of the above properties so long as the range of the function is large enough.

The list of currently recommended hash functions is given in table 1. Each hash function has a unique entry index represented by a string beginning with "1." followed by a two-digit entry number.

Table 1: The list of recommended hash functions

Hash function entry index	Short hash function entry name	Adoption date	Normative references
1.01	sha1	01.01.2001	ISO/IEC 10118-3 [3] and FIPS Publication 180-2 [4]
1.02	ripemd160	01.01.2001	ISO/IEC 10118-3 [3]
1.03	sha-224	2004	FIPS Publication 180-2 [4]
1.04	sha-256	2004	ISO/IEC 10118-3 [3] and FIPS Publication 180-2 [4]
1.05	Whirlpool	2004	ISO/IEC 10118-3 [3]
NOTE 1: Additional secure hash algorithms, beside the SHA-2 family, was needed. For that reason the Whirlpool algorithm has been added. This algorithm has been reviewed by NESSIE experts. NOTE 2: SHA-384 and SHA-512, defined in ISO/IEC 10118-3 [3], may also be used, the security level they are supposed to provide is above SHA-224 and SHA-256, therefore they are not mentioned here.			

5.2 Recommended one way hash functions

5.2.1 SHA1

1) SHA-1 MAY be used to hash a message, *M*, having a length of up to 2^{64}-1 bits.

The final result of SHA-1 is a 160-bits message digest. The SHA-1 algorithm is described in ISO/IEC 10118-3 [3] and FIPS Publication 180-2 [4].

NOTE: Recently an attack against SHA-1 has been announced that allegedly can produce collisions with an effort of 2^{69} which would be a (theoretical) break of this hash function. Analogy with former attacks against hash functions suggests that probably the attack is well suited for parallelization and that the effort for finding meaningful collisions is not notable higher. A necessary effort of 2^{69} operations can still be regarded as not feasible at this moment but more effective methods may perhaps be found soon. Because of this, at least SHA-224 and SHA-256 SHOULD be implemented for any new product for electronic signatures. One should also develop plans how to switch quickly to other hash functions for electronic signatures in the case that SHA-1 and/or RIPEMD160 in fact turn out to be too weak.

5.2.2 RIPEMD-160

RIPEMD-160 MAY be used to hash a message. RIPEMD-160 is a 160-bit cryptographic hash function, designed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. It is described in ISO/IEC 10118-3 [3]. It is replacing the 128-bit hash function RIPEMD. The maximal message size is to 2^{64}-1.

NOTE: RIPEMD-320 is constructed from RIPEMD-160 by initializing the two parallel lines with different initial values, omitting the combination of the two lines at the end of every application of the compression function, and exchanging a chaining variable between the 2 parallel lines after each round. The security level of the 320-bit extension of RIPEMD-160 is the same as that of RIPEMD-160 itself. Similarly the 256-bit extension of RIPEMD-128, i.e. RIPEMD-256 is the same as that of RIPEMD-128.

5.2.3 SHA-224

SHA-224 MAY be used to hash a message, *M*, having a length of up to 2^{64}-1 bits and the output size is 224 bit. The function is defined in the exact same manner as SHA-256 (clause 6.2), except for two operations in the formation of the initial and final hash values. The SHA-224 algorithm is described in FIPS Publication 180-2 [4].

5.2.4 SHA-256

1) SHA-256 MAY be used to hash a message, *M*, having a length of up to 2^{64}-1bits.

The final result of SHA-256 is a 256-bit message digest. The SHA-256 algorithm is described in FIPS Publication 180-2 [4].

5.2.5 WHIRLPOOL

WHIRLPOOL is a hash function designed by Vincent Rijmen and Paulo S. L. M. Barreto that operates on messages less than 2^{256}-1 bits in length, and produces a message digest of 512 bits.

Whirlpool MAY be used to compute the imprint of a message placed in a time-stamp token.

Whirlpool MAY only be used with a secure signature scheme supporting key sizes that match the Whirlpool output, i.e. 512 bits. DSA and ECDSA cannot be used with Whirlpool. However, it MAY be used with the RSA algorithm. The WHIRLPOOL algorithm is described in ISO/IEC 10118-3 [3].

NOTE: The Whirlpool output, i.e. 512 bits, is much more than what is needed, but there is currently no definition of a Whirlpool algorithm variant with an output less than 512 bits. Whirlpool has been included as an alternative to the SHA-2 family and can be used either to compute a hash value (for a time-stamp token) or with the RSA algorithm.

6 Signature schemes

A signature scheme consists of three algorithms: a key generation algorithm and a signature creation algorithm and a signature verification algorithm. The later 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 currently recommended signature algorithms is given in table 2. Each signature algorithm has a unique entry index represented by a string beginning with "2." followed by a two-digit entry number.

Table 2: The list of recommended signature algorithms

Signature algorithm entry index	Short signature algorithm entry name	Key and Parameter generation algorithms	Normative references
2.01	RSA	Rsagen1	RFC 3447 [14]
2.02	DSA	Dsagen1	FIPS Publication 186-2 [6]
2.03	ecdsa-Fp	Ecgen1	ANSI X9.62 [7]
2.04	ecdsa-F2m	Ecgen2	ANSI X9.62 [7]
2.05	ecgdsa-Fp	Ecgen1	[8]
2.06	ecgdsa-F2m	Ecgen2	[8]

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

The RSA algorithm's security is based on the difficulty of factoring large integers. The RSA computations SHALL be performed as described in RFC 3447 [14]. To generate the key pair two prime numbers, *p* and *q*, are generated randomly and independently, satisfying the following requirements:

- the bit length of the modulus $n = p\ q$ must be at least MinModLen; its length is also referred to as ModLen;
- *p* and *q* should have roughly the same length, e.g. set a range such as $0{,}1 < |\log_2 p - \log_2 q| < 30$;
- the set of primes from which p and q are (randomly and independently) selected SHALL be sufficiently large and reasonably uniformly distributed.

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*.

CRT (Chinese Remainder Theorem) implementations are also allowed, in which case the private key will contain more values derived from the factorization of the modulus n.

For RSA signatures also a padding method has to be specified.

6.1.2.2 DSA

The DSA algorithm's security is based on the difficulty of computing the discrete logarithm in the multiplicative group of a prime field F_p. The DSA computations SHALL be performed as described in FIPS Publication 186-2 [6] with the change notice. The public parameters *p*, *q* and *g* MAY be common to a group of users. The prime modulus *p* SHALL be at least pMinLen bits long. *q*, which is a prime divisor of (*p*-1), SHALL be at least qMinLen bits long. *g* SHALL be computed as indicated in FIPS Publication 186-2 [6] with the change notice.

The private key consists of:

- the public parameters *p*, *q* and *g*;
- a statistically unique and unpredictable integer *x*, $0 < x < q$, which is signatory-specific; and
- a statistically unique and unpredictable integer *k*, $0 < k < q$, which must be regenerated for each signature.

If the distribution of *k* is significantly different from uniform within the interval then there may be weaknesses. Bleichenbacher has presented an attack which can be sub-exhaustive depending on the size of the bias and the number of signatures produced using a single secret key. The value of *k* must be kept secret as well as the private key, even if *k* is only partially known there exists an attack (Nguyen/Shparlinski).

The public key consists of *p*, *q*, *g* and an integer *y* computed as $y = g^x \bmod p$.

When computing a signature of a message M, no padding of the hashcode is necessary. However, the hashcode must be converted to an integer by applying the method described in appendix 2.2 of FIPS Publication 186-2 [6] with the change notice.

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

This signature algorithm is referred to as ecdsa-Fp. The algorithm SHALL be applied as specified in ANSI X9.62 [7]. The same algorithm is also specified in ISO/IEC 14888-3 [32], IEEE P1363 [5] and ISO/IEC 15946-2 [8] which can be used for information. The security of the ecdsa-Fp algorithm is based on the difficulty of computing the elliptic curve discrete logarithm.

The public parameters are as follows:

- *p* prime;
- *q* large prime at least qMinLen bits long, $p \neq q$;
- *E* elliptic curve over a finite field F_p whose order *n* is divisible by *q*; and
- *P* point on $E(F_p)$ of order *q*.

The public parameters MAY be common to a group of users. The quotient *h* of the group order *n* divided by *q* may be considered as a public parameter too.

The class number of the maximal order of the endomorphism ring of *E* SHALL be at least MinClass=200.

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

h = n/q must be less or equal 4. (see http://www.secg.org/collateral/sec1.pdf).

In FIPS Publication 186-2 [6] five curves over a prime field are defined. All these curves fulfil the above requirements.

The private key consists of:

- the public parameters *E*, *m*, *q* and *P*;
- a statistically unique and unpredictable integer *x*, $0 < x < q$, which is signatory-specific; and
- a statistically unique and unpredictable integer *k*, $0 < k < q$, which must be regenerated for each signature.

The public key consists of *E*, *q*, *P* and *Q*, a point of *E*, which is computed as $Q = xP$.

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

This signature algorithm is referred to as ecdsa-F2m. The algorithm SHALL be applied as specified in ANSI X9.62 [7]. The same algorithm is also specified in ISO/IEC 14888-3 [32], IEEE P1363 [5], and ISO/IEC 15946-2 [8] which can be used for information. The security of the ecdsa-F2m algorithm is based on the difficulty of computing the elliptic curve discrete logarithm.

The public parameters are as follows:

- *m* prime number;
- *q* large prime at least qMinLen bits long;
- *E* elliptic curve over a finite field F_{2m} whose *n* order is divisible by *q*;
- it must not be possible to define *E* over $F_{2;}$ and
- *P* point on $E(F_2{}^m)$ of order *q*.

h = n/q must be less or equal 4 (see http://www.secg.org/collateral/sec1.pdf).

The class number of the maximal order of the endomorphism ring of E SHALL be at least MinClass=200. The value r_0:=min(r: q divides 2^{mr}-1) SHALL be greater than r0Min=10^4.

In FIPS Publication 186-2 [6] five pseudorandomly generated curves over F_{2m} are defined. All these curves satisfy the above requirements. Note that the Koblitz curves given in FIPS Publication 186-2 [6] are defined over F_2 and hence do not fulfil the fourth requirement.

A field representation is required, common to both the signatory and the verifier, so that signatures can be interpreted correctly. The representations given in IEEE P1363 [5] and FIPS Publication 186-2 [6] are recommended. Thus if a polynomial basis is required then an irreducible trinomial of the form $x^m + x^a + 1$ with minimal a should be used. If such a polynomial does not exist then an irreducible pentanomial of the form $x^m + x^a + x^b + x^c + 1$ should be used; a should be minimal, b should be minimal given a and c should be minimal given a and b.

The private key consists of:

- the public parameters E, m, q and m;
- a statistically unique and unpredictable integer x, $0 < x < q$, which is signatory-specific; and
- a statistically unique and unpredictable integer k, $0 < k < q$, which must be regenerated for each signature.

The public key consists of E, q, m and Q, a point of E which is computed as $Q=xP$.

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

This signature algorithm is referred to as ecgdsa-Fp. The algorithm SHALL be applied as specified in ISO/IEC 15946-2 [8]. The security of the ecgdsa-Fp algorithm is based on the difficulty of computing the elliptic curve discrete logarithm.

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 should satisfy all the constraints given in clause 6.2.3.

NOTE: The basic difference between ECDSA and EC-GDSA is that during signature creation k does not need to be inverted for ECGDSA. Under certain circumstances this can be advantageous for the design and performance of the SCDev.

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

This signature algorithm is referred to as ecgdsa-F2m. The algorithm SHALL be applied as specified in ISO/IEC 15946-2 [8]. The security of the ecgdsa-F2m algorithm is based on the difficulty of computing the elliptic curve discrete logarithm.

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 should satisfy all the constraints given in clause 6.2.4.

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

6.2 Recommended key pair generation methods

6.2.1 General

Key pair generation methods are not part of the definition of a signature suite and may evolve without the need to change the identifier of the signature suite.

Table 3 summarizes the recommended key pair generation methods for all signature algorithms considered in the present document. Each key pair generation method has a unique entry index represented by a string beginning with "3" followed by a two-digit entry number.

Table 3: The list of recommended key pair generation methods

Key generator entry index	Short key generator entry name	Signature algorithm	Random number generation method	Random generator parameters	Adoption date	Normative references
3.01	rsagen1	rsa	trueran or pseuran	Up to 2010: EntropyBits ≥80 or SeedEntropy ≥ 80 Beyond 2010: EntropyBits ≥100 or SeedEntropy ≥ 100	01.01.2001	
3.02	dsagen1	dsa	trueran or pseuran	Up to 2010: EntropyBits ≥ 80 or SeedEntropy ≥80 Beyond 2010: EntropyBits ≥100 or SeedEntropy ≥ 100	01.01.2001	FIPS Publication 186-2 [6]
3.03	ecgen1	ecdsa-Fp, ecgdsa-Fp	trueran or pseuran	Up to 2010: EntropyBits ≥ 80 or SeedEntropy ≥ 80 Beyond 2010: EntropyBits ≥100 or SeedEntropy ≥ 100	01.01.2001	
3.04	ecgen2	ecdsa-F2m, ecgdsa-F2m	trueran or pseuran	Up to 2010: EntropyBits ≥80 or SeedEntropy ≥ 80 Beyond 2010: EntropyBits ≥100 or SeedEntropy ≥ 100	01.01.2001	

6.2.2 Recommended key pair generation methods

6.2.2.1 Key and parameter generation algorithm rsagen1

Generate p and q as indicated in clause 6.1.2.1 by applying a random number generation method satisfying the requirements trueran (see clause 8.2.1) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed. Each prime SHALL effectively be influenced by EntropyBits bits of true randomness or a seed of entropy SeedEntropy bit. 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 2^{-80}. Details on generating random primes can be found in ISO/IEC 18032 (see bibliography), in particular section 8.2. Examples of algorithms to produce RSA moduli, i.e. pairs of primes satisfying the condition $0{,}1 < | \log_2 p - \log_2 q | < 30$ are given in annex C.

NOTE 1: Annex A of ISO/IEC 18032 (see bibliography) contains a table of error probabilities for different probabilistic primality tests.

EXAMPLE: For a random number of 1 024 bit tested with three successful iterations of the Miller-Rabin test the probability that this number is not a prime is about 2^{-93}.

The private exponent d and the public exponent e must satisfy $ed \equiv 1$ (mod lcm (p-1, q-1)) which is automatically the case if $ed \equiv 1$ (mod (p-1)(q-1)). The private exponent d must not be too small (Wiener 1990, Boneh and Durfee 1999, Durfee and Nguyen 1999, see bibliography); it is sufficient to choose d in a range at least $\sqrt{n}$ from its minimum and maximum values.

In practice by randomly choosing the public exponent e (subject to the condition gcd(e,(p-1)(q-1))=1) the corresponding private exponent d will satisfy that condition with very high probability. If e is chosen small (e.g. less than $n^{0,125}$) the condition on d will automatically be satisfied.

NOTE 2: It may also be recommendable to choose e not too small ($e > 2^{16}+1$) as for example results of Boneh and Venkatesan suggest that for very small e the RSA problem could be easier than factoring. Nevertheless in contrast to RSA encryption small public exponents generally are not a direct threat for RSA signatures.

A small public exponent (e.g. e = 3) *MAY* be used if performance is critical, otherwise $e \geq 2^{16}+1$ is RECOMMENDED.

A new modulus has to be produced for each user of the signature scheme even if different public exponents are used. In practice if the moduli 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.

NOTE 3: It is not recommended to use a prime selection algorithm which prefers a special class of primes. For example according to Rivest and Silverman (see bibliography) the use of "strong" primes would not improve security in practice.

6.2.2.2 Key and parameter generation algorithm dsagen1

p and *q* SHALL be generated as described in appendix 2.2 of FIPS Publication 186-2 [6].

Generate *x* by applying a random number generation method satisfying the requirements trueran (see clause 8.2.1) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed. Each value of x SHALL effectively be influenced by EntropyBits bits of true randomness or a seed of entropy SeedEntropy bit. Generate *k* using one of these methods; *k* does not have to be generated using exactly the same method as *x*. Possible methods for this can be found in FIPS Publication 186-2 [6] which contains a Change Notice (due to Bleichenbacher's attack).

6.2.2.3 Key and parameter generation algorithm ecgen1 for ecdsa-F_p

The prime numbers *p* and *q*, and the point *P* on $E(F_p)$ SHALL be selected so 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 2^{-100}. Clause D.1 specifies a possible method to generate *p, q, E* and *P*.

In situations where an intentional choice of weak public parameters (subject to an unknown "insider" attack) seems to be possible a countermeasure is to request that these parameters are generated verifiably at random. In such situations it is recommended to do so at least for the generation of the curve *E*. In clause D.1 a possible method for this is described.

Generate *x* by applying a random number generation method satisfying the requirements trueran (see clause 8.2.1) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed. Each value of *x* SHALL effectively be influenced by EntropyBits bits of true randomness or a seed of entropy SeedEntropy bit. Generate *k* using one of these methods; *k* does not have to be generated using exactly the same method as *x*.

6.2.2.4 Key and parameter generation algorithm ecgen2 for ecdsa-$F_2{}^m$

The prime numbers *m* and *q*, the elliptic curve *E* over $F_2{}^m$ and the point *P* on $E(F_2)$ SHALL be selected so that the conditions in 6.1.2.4 are satisfied with primality of an integer regarded as satisfied if the probability that it is composite is at most 2^{-100}. Clause D.2 specifies a possible method to generate *m, q, E* and *P*.

In situations where an intentional choice of weak public parameters (subject to an unknown "insider" attack) seems to be possible a countermeasure is to demand that these parameters are generated verifiably at random. In such situations it is recommended to do so at least for the generation of the curve *E*. In clause D.2 a possible method for this is described.

Generate *x* by applying a random number generation method satisfying the requirements trueran (see clause 8.2.1) or using a method satisfying pseuran (see clause 8.2.2) with an appropriate size seed. Each value of *x* SHALL effectively be influenced by EntropyBits bits of true randomness or a seed of entropy SeedEntropy bit. Generate *k* using one of these methods; *k* does not have to be generated using exactly the same method as *x*.

6.2.2.5 Key and parameter generation algorithm ecgen1 for ecgdsa-F_p

The parameter and key generation methods should be the same as the ecdsa-F2m methods described in clause 6.2.2.3.

6.2.2.6 Key and parameter generation algorithm ecgen2 for ecgdsa-$F_2{}^m$

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

7 Signature suites

7.1 General

To meet this security requirement and to allow signing of more or less arbitrary long messages, a signature suite requires 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 could define the overall security level.

Due to possible interactions which may influence security of electronic signatures, algorithms and parameters for secure electronic signatures SHALL be used only in predefined combinations referred to as the signature suites. A signature suite consists of the following components:

- a hash function;
- a padding method;
- a signature algorithm and its associated parameters.

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

The list of recommended hash functions is defined in clause 5.2.

The list of recommended padding methods is defined in clause 7.2.

The list of recommended signature algorithms is defined in clause 6.2.

The list of currently recommended signature suites is given in clause 7.3.

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

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

7.2 Padding methods

Padding is algorithm dependent and some algorithms need non-trivial 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 may be also randomized, 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.

The list of currently recommended padding methods is given in table 4. Each padding method has a unique entry index represented by a string beginning with "4" followed by a two-digit entry number.

Table 4: The list of recommended padding methods

Padding method entry index	Short padding function entry name	Random number generation method	Random generator parameters	Normative references
4.01	emsa-pkcs1-v1.5	-	-	RFC 3447 [14]
4.02	emsa-pkcs1-v2.1	-	-	RFC 3447 [14], section 9.2
4.03	emsa-pss	trueran/pseuran	EntropyBits ≥ 64 or SeedEntropy ≥ 64	RFC 3447 [14], section 9.1

Padding method entry index	Short padding function entry name	Random number generation method	Random generator parameters	Normative references
4.04	iso9796ds2	trueran/pseuran	EntropyBits ≥ 64 or SeedEntropy ≥ 64	RFC 3447 [14]
4.05	iso9796-din-rn	trueran/pseuran	EntropyBits ≥ 64 or SeedEntropy ≥ 64	DIN 66291-1 [16]
4.06	iso9796ds3	-	-	RFC 3447 [14]

Each salt value SHALL effectively be influenced by at least 64 bits of true randomness or a seed of entropy at least 64 bit. This rule implies that the salt length is at least 64 bit.

NOTE 1: The above rule of 64 bit salt entropy is not meant in the strict and exclusive manner as the demand for 80 bits of entropy for key generation in clause 6.2, it is a recommendation. For example Coron (see bibliography) showed that for emsa-pss already a significantly shorter salt length than 64 bit allows a reduction of the security of the signature scheme to the RSA problem under realistic assumptions. Nevertheless such a reduction analysis does not take into account every kind of possible weaknesses e.g. side channels. So the salt length should not be too short in particular when high security shall be achieved.

The emsa-pkcs1-v1.5 padding method is included, but it is NOT RECOMMENDED for new implementations, since it will be phased out.

NOTE 2: Up to December 2004, no real attack on emsa-pkcs1-v1.5 has been publicized.

The emsa-pss method is included as, despite not being widely used, it has been stable for a long time and is a good improvement to the two emsa-pkcs1 schemes (i.e. -v1.5 and -v2.1 which only differ by the encoding method) and it is better suited for long term use. The padding method emsa-pss is parameterized by the choice of hash function and a mask generation function MGF, defined in PKCS#1 (RFC 3447 [14]). In this specification, MGF is based on the corresponding hash function used, i.e. SHA-1 or RIPEMD-160.

iso9796ds2 is "digital signature scheme 2" in ISO/IEC 9796-2 [17].

iso9796-din-rn is the variant of a scheme from ISO/IEC 9796-2 [17] called "DSI according to ISO/IEC 9796-2 [17] with random numbers" in DIN V 66291-1 [16]. It is described in annex A of [16].

NOTE 3: This is a variant on Digital Signature Scheme 1 of ISO/IEC 9796-2 [17]. The Digital Signature Scheme 1 has wide deployments and is secure but maybe in the future not be recommended for new systems.

iso9796ds3 is "digital signature scheme 3" in ISO/IEC 9796-2 [17].

NOTE 4: iso9796ds1 does no longer represents state-of-the-art and in a paper presented by Coron, Naccache, Stern at Crypto 99 is shown that the effort to break this padding scheme is about 2^{61} instead of 2^{80}. It could even be easier to forge signatures if the hash values are produced outside the SSCD, that means if an attacker would be given the ability to let "chosen prime numbers be signed". In that case it would no longer be required to solve a linear system of equations as in clause 3.1 of the paper of Coron et al.

7.3 Recommended signature suites

A signature suite is defined using three parameters:

- a hash function;
- a padding method;
- a signature algorithm and its associated parameters.

Table 4.a

entry name of the signature suite	entry name for the hash function	entry name for the padding method	entry name for the signature algorithm
sha-1-with-rsa	sha1	(see note)	rsa
sha-1-with-dsa	sha1	no padding required	dsa
ripemd160-with-rsa	ripemd160	(see note)	rsa
ripemd160-with-dsa	ripemd160	no padding required	dsa
sha224-with-rsa	sha224	(see note)	rsa
sha256-with-rsa	sha256	(see note)	rsa
rsa-pss with mgf1SHA1Identifier	mgf1SHA1		rsa
rsa-pss with mgf1SHA224Identifier	mgf1SHA224		rsa
rsa-pss with mgf1SHA256 Identifier	mgf1SHA256		rsa
sha-1-with-ecdsa	sha1	no padding required	ecdsa-Fp or ecdsa-F2m
sha-1-with-ecgdsa	sha1	no padding required	ecdsa-Fp or ecgdsa-F2m
NOTE: The padding scheme for the RSA signature algorithm SHOULD be selected from the list above.			

8 Random number generation methods

8.1 General

The key generation methods and some signature suites require to generate a random number.

NOTE: For detailed information about random number generation and terminology see ISO/IEC FCD 18031 (see bibliography). Some basic information is also given in annex E.

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

8.2 Recommended random number generation methods

Table 5 lists the recommended random number generation methods. 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).

Table 5: The list of recommended random number generation methods

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

It is strongly recommended to use trueran methods for generating keys that are used more than once. In the case of the one-time keys *k* for DSA, ECDSA and ECGDSA there is less urgency for that.

8.2.1 Random generator requirements trueran

A random number generator satisfying trueran has to be a pure or hybrid *physical* NRNG.

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

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

The **recommended** requirements for these components are:

- **(TR1):** There is a stochastical model for the primary entropy source which is found consistent with thorough adapted tests of prototypes of the source.

- **(TR2):** The primary entropy source is subjected to an *adapted* statistical *online* test. "Online" means that the test will detect any non-tolerable 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 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).

See clause E.2 for some more information about tests for the primary entropy sources.

The stochastical model and the tests should deliver an estimate for the amount of the produced entropy. The primary entropy source is regarded to be *good* if it produces nearly one bit entropy per output bit. For a good primary entropy source no post-treatment is necessary.

- **(TR3):** If the primary entropy source is not good a post-treatment is employed which by some (necessarily compressing) techniques delivers an output of nearly one bit entropy per output bit. There must be a reasonable stochastical model of the post-treatment as well which together with the stochastic model of the primary entropy source and the tests ensures this property of the output.

Instead of this set of requirements **(TR1) - (TR3)** the following modified set of requirements is also **sufficient** although not recommended:

- **(TR1"):** There are mathematical models for the primary entropy source and the post-treatment that are plausible.

- **(TR2"):** The primary entropy source is 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 (obligatory in this case) that under the assumption of the models (assuming that the primary entropy source works as expected) delivers an output of nearly one 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 8.2.2.

NOTE 2: This alternative set of requirements is closer to the spirit of ANSI X9.82 (see bibliography) 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 3: An example of a possible random number generator design based on a noisy diode is given in clause E.2 of ISO/IEC FCD 18031 (see bibliography) although without the necessary details.

8.2.2 Random generator requirements pseuran

A random number generator satisfying pseuran is a pure or hybrid DRNG satisfying the following conditions:

- **(PR1):** The DRNG must be initialized by a seed with an entropy of at least SeedEntropy bits.

- **(PR2):** Even with the knowledge of a partial output bit sequence of the DRNG and having all information about its initialization (and in the case of a hybrid DRNG also about the output of the additional entropy source) except for the seed there is no usable method to determine any other *m* bits of the output with a probability significantly larger then 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 DRNG can be recovered from a subset of the output.

(PR1) is meant in the sense (or even implies) that the seed is produced using a NRNG. This NRNG does not need to be a physical one. Nevertheless to achieve high security it is recommended to use trueran (in particular physical, see clause 8.2.1) NRNGs for seeding. **(PR1)** does not exclude constructions in which the DRNG is seeded by a chain of DRNGs as described in clause 9.3.2 of ISO/IEC FCD 18031 (see bibliography). However the first DRNG in this chain must be seeded with the output of a NRNG and in the output of the last DRNG in the chain enough entropy (i.e. at least EntropyBits bits) has to be left over. Moreover of course the whole system (chain + DRNG to be seeded) regarded as a DRNG (including operational freedom like numbers of cycles before the next seeding of links regarded as non-physical additional entropy source) has to satisfy the second condition. The security of a DRNG is only complexity theoretic. With a known seed or a known internal state any future output can be calculated. So the seed has to be kept secret and seeding SHALL follow procedures similar to those for the generation of root keys. No backups of the seed or internal states of a pseuran generator are permitted. The internal state of the DRNG must be secured against any readout and any adversarial manipulation.

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 has to 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. 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.

It is recommended to use DRNGs which in addition to the two above mentioned conditions 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 there is no usable method to determine any previous *m* output bits with a probability significantly larger then Max$(2^{-m}, 2^{-\text{SeedEntropy}})$.

NOTE 2: AIS 20 (see bibliography) defines the classes K3 and K4 for DRNGs. Roughly said K3 DRNGs satisfy conditions **(PR1)** and **(PR2),** K4 DRNGs also satisfy **(PR3)**.

Depending on the environment it may further be recommendable to use hybrid DRNGs rather than pure ones. In the case of an hybrid DRNG according to **(PR2)** even with complete knowledge about the output of the additional entropy source or with a certain influence on this output it must not be feasible to determine any bits of the output with higher than the a priori probability.

The following are examples of pseuran generators:

- ANSI X9.17 (see in bibliography) 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 could be used as building block of the generator.

- Example E.4 in AIS 20 is another DNRG based on a variable strong block cipher which as well does not satisfy condition **(PR3).**

- FIPS 186 generator (see FIPS Publication 186-2 [6]).

- RSA DRNG and Blum-Blum-Shub DRNG (see Menezes et al. 1997 in bibliography). Those DRNGs 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 DRNGs described above, the security of which is only heuristic.

9 Recommended hash functions and key sizes versus time

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.

This clause is structured as follows:

- in the first two clauses, two different ways of looking at key length recommendations, that are called the "liberal view" and the "conservative view", are introduced;
- in clause 3, hash functions versus time are recommended;
- in clause 4, key sizes versus time are recommended.

These recommendations were based upon current predictions in the literature [LenstraVerheul] as well as consultations with bodies such as ETSI SAGE.

Two different views on the necessary key lengths are considered: the liberal and the conservative view. It can generally be stated that the liberal view reflects the smaller security margin and the conservative view the upper limit of several possible security margins. Based on the use and/or the context in which the signature algorithm, signature suite, or hash function is being used a particular user or implementer can choose the liberal and the conservative view (see clause 9 about the use of hash functions, signature algorithms and signature suites).

NOTE: The liberal as well as the conservative view recommendations may not be sufficient to satisfy certain legal demands imposed by some national signature laws.

9.1 Liberal view

The liberal view of algorithm and hash function strength is characterized by:

- An assumption that there will no unpredicted acceleration in the pace of development of techniques to break the algorithm or hash function.
- An assumption that breaking the algorithm or hash function will be based on either a current model or extrapolation of such.
- Taking a small margin above minimum key length based on both extrapolation of current trends as well as estimations based on the necessary computing power needed to break a given algorithm.

NOTE: In practice, the key length estimates given for the liberal view are appropriate when the electronic signature formats that are being used cover the case of a key compromising or of a hash function exhibiting collisions.

9.2 Conservative view

The conservative view of algorithm and hash function strength is characterized by:

- An attempt to "predict" unforeseen advances in state of the art in analyzing the hash function or algorithm.
- An assumption that new models may be developed to break the hash function or algorithm.
- Taking a comfortable margin above minimum key length based on both extrapolation of current trends as well as estimations based on the necessary computing power needed to break a given algorithm.

NOTE: In practice, the conservative view should be applied when measures to recover from an incorrect estimate are either not available (e.g. successful attack on a root key) or deemed to be complicated (e.g. collision for the hash function used by the signer).

9.3 Recommended hash functions versus time

Tables 6 and 7 provides indication about recommended hash functions during X years. With respect to the above distinction between conservative and liberal views, the conservative view is first provided and then the liberal.

Definitions:

Usable: The algorithm with the given security parameters can be considered secure at the given time.

Unknown: The security of the algorithm is unknown; use in this case is environment dependent.

Table 6: Conservative view of recommended hash functions for a resistance during X years

entry name of the hash function	3 years (2008)	5 years (2010)	10 years (2015)
sha1	usable	unknown	unusable
ripemd160	usable	unknown	unknown
sha224	usable	usable	usable
sha256	usable	usable	usable
whirlpool	usable	usable	unknown

Table 7: Liberal view for recommended hash functions for a resistance during X years

entry name of the hash function	3 years (2008)	5 years (2010)	10 years (2015)
sha1	usable	unknown	unknown
ripemd160	usable	unknown	unknown
sha224	usable	usable	usable
sha256	usable	usable	usable
whirlpool	usable	usable	unknown

Table 8 predicts hash function resistance over 20 years. Due to the inherent uncertainty of such predictions, these predictions are largely speculative, and no distinction is made be between conservative and liberal.

Additional definition:

Unusable: The algorithm cannot be considered secure for any kind of use in the context of electronic signatures.

Table 8: Speculated hash function resistance over 20 years, based on current trends and estimated computational power

entry name of the hash function	Speculated Usability in 20 years (2025)
sha1	unusable
ripemd160	unusable
sha224	usable
sha256	usable
whirlpool	usable

9.4 Recommended key sizes versus time

Tables 9 and 10 provides indication about recommended key lengths for a resistance of the algorithm or the signature suite during X years. A conservative view and a liberal view are provided. For explanations of these terms please refer to clauses 2 and 3 in this clause. These tables are provided for 3 years, 5 years and 10 years.

Definitions:

Unusable: The algorithm cannot be considered secure for any kind of use in the context of electronic signatures.

Unknown: The security of the algorithm is unknown at this time.

Table 9: Conservative view of recommended key lengths for a resistance during X years

entry name of the signature suite	3 years (2008)	5 years (2010)	10 years (2015)
sha-1-with-rsa	≥ 768	unknown	unusable
sha224-with-rsa	≥ 768	1024	2048
sha256-with-rsa	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA1Identifier	≥ 768	1024	unusable
RSASSA-PSS with mgf1SHA224Identifier	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA256Identifier	≥ 768	1024	2048
sha1-with-dsa	≥ 768	unknown	unusable
sha1-with-ecdsa	163	unknown	unusable
sha224-with-ecdsa	224	224	224
sha256-with-ecdsa	256	256	256

Table 10: Liberal view of recommended key lengths for a resistance during X years

entry name of the signature suite	3 years (2008)	5 years (2010)	10 years (2015)
sha-1-with-rsa	≥ 768	unknown	unknown
sha224-with-rsa	≥ 768	1024	2048
sha256-with-rsa	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA1Identifier	≥ 768	1024	unknown
RSASSA-PSS with mgf1SHA224Identifier	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA256Identifier	≥ 768	1024	2048
sha1-with-dsa	≥ 768	unknown	unknown
sha1-with-ecdsa	163	unknown	unknown
sha224-with-ecdsa	224	224	224
sha256-with-ecdsa	256	256	256

An additional table with speculations regarding security over 20 years is provided. Due to the unpredictability in such long term statements, there is no distinction made between liberal and conservative for the 20 years prediction, and such information should be considered as largely speculative.

Table 11: Speculated resistance of recommended algorithm parameters for the next 20 years, based on current trends and estimated computational power

entry name of the signature suite	20 years (2025)
sha-1-with-rsa	Unusable
sha224-with-rsa	2048
sha256-with-rsa	2048
RSASSA-PSS with mgf1SHA1Identifier	unusable
RSASSA-PSS with mgf1SHA224Identifier	2048
RSASSA-PSS with mgf1SHA256Identifier	2048
sha1-with-dsa	unusable
ecPublicKey	unknown
sha1-with-ecdsa	unusable

10 Time period resistance of hash functions and keys

The hash functions and signature algorithms defined in the present document are suitable to be used in the context of advanced electronic signatures as defined by the EESSI documents (both ETSI TSs and CWAs).

As a general rule, a private key SHALL resist during the validity period of certificates, (defined by the "notBefore" and "notAfter" elements of the validity period field) which contain the corresponding public key.

NOTE: The validity period is defined by the "notBefore" and "notAfter" elements of the validity period field from the certificate.

Since key sizes are directly dependent upon the usage of the certificate, no single key size value may be given.

The time period during which a given key SHALL or SHOULD resist depends on the usage of the key. To this respect different use cases will be explored. Once the time period is known, then the figures provided in clause 9 can be used to know the appropriate key size.

10.1 Time period resistance for hash functions

As a general rule, 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 H for more information).

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

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. It may be advisable, in order to reduce the signature maintenance process, to use a hash function that is presumed to be resistant over a very long time period. If the hash function that has been used the signature suite by the signer is also presumed to be resistant over a very long time period, then the signature maintenance process can be minimized.

10.2 Time period resistance for signer's key

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

Signer's keys SHOULD resist during the validity period (from `notBefore` to `notAfter`) of the associated certificate. If they do not, revocation will be necessary, and there would be a large burden to re-issue new keys and certificates. However, there is no security breach.

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.

For signer's keys, the liberal view for the resistance SHOULD be chosen.

10.3 Time period resistance for trust anchors

For trust anchors, the conservative view for the resistance SHOULD be chosen.

A trust anchor SHOULD remain secure during the whole time period during which advanced electronic signature needs to be verified. If it does not, 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.

10.4 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.

If they do not, a maintenance process SHOULD be applied before the algorithm is broken.

For these keys, the conservative view for the resistance SHOULD be chosen if no signature maintenance process is being envisaged, while the liberal view for resistance MAY be chosen if a signature maintenance process is applied.

11 Practical ways to identify hash functions and signature algorithms

In order to be able to use a function or an algorithm with the EESSI documents, it is mandatory to be able to reference it, and when the algorithm has parameters to be able to define these parameters. An "object" needs to be defined to support these parameters. That object MUST be referenced using an OID and/or a URN. Only the owner of the OID or the URN is allowed to define its meaning and thus the meaning of the algorithm, usual referencing another document. It may be observed that ISO standards are not referenced in RFCs documents. The primary reason is that these documents are sold and the IETF always gives its preference to documents that can be obtained for free.

As a general rule the "OID/URN criterion" may be applied: An algorithm to be included must be defined unambiguously by an OID/URN. If such an OID/URN is not available it may be useful to define it.

11.1 Hash functions and signature algorithms objects identified using OIDs

11.1.1 Hash functions

The hash functions are defined using the following OIDs.

Table 12

Short object name	OID	Normative references
id-sha1	{ iso(1) identifiedOrganization(3) oIW(14) oIWSecSig(3) oIWSecAlgorithm(2) 26 }	RFC 3279 [12]
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 [15]
id-sha256	{ joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3) nistalgorithm(4) hashalgs(2) 1 }	RFC 4055 [15]
ripemd160	{ iso(1) identifiedOrganization(3) teletrust(36) algorithm(3) hashAlgorithm(2) ripemd160(1) }	ISIS-MTT Part 6 [26]
whirlpool	{ iso(1) standard(0) encryption-algorithms(10118) part3(3) algorithm(0) whirlpool(55) }	ISO/IEC 10118-3 [3]

11.1.2 Signature algorithms

Table 13

Short object name	OID	Normative references
rsaEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }	RFC 3279 [12]
id-dsa	{ iso(1) member-body(2) us(840) x9-57(10040) x9cm(4) 1 }	RFC 3279 [12]
id-ecPublicKey	{ iso(1) member-body(2) us(840) 10045 2 1 }	RFC 3278 [11]

11.1.3 Signature suites

Table 14

Short object name	OID	Normative references
sha-1withRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }	RFC 3279 [12]
sha224WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 14 }	RFC 4055 [15]
sha256WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 11 }	RFC 4055 [15]
id-RSASSA-PSS with mgf1SHA1Identifier	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }	RFC 4055 [15]
id-RSASSA-PSS with mgf1SHA224Identifier	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }	RFC 4055 [15]
id-RSASSA-PSS with mgf1SHA256Identifier	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }	RFC 4055 [15]
rsaSignatureWithripemd160	{iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) rsaSignature(1) rsaSignatureWithripemd160(2)}	ISIS-MTT [26]
id-dsa-with-sha1	{ iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 3 }	RFC 3279 [12]
id-ecdsa-with-sha1	{ iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) 1 }	RFC 3279 [12]

11.2 Hash functions and signature algorithms identified objects using URNs

11.2.1 Hash functions

The hash functions are defined using the following URNs.

Table 15

Short object name	URN	Normative references
sha1	http://www.w3c.org/2000/09/xmldsig#sha1	W3C Recommendation XML-Signature Syntax and Processing [30].
ripemd160	http://www.w3.org/2001/04/xmlenc#ripemd160	W3C Recommendation XML Encryption Syntax and Processing. 10 December 2002 [31]
sha224	http://www.w3.org/2001/04/xmldsig-more#sha224	RFC 4050 [28]
sha256	http://www.w3.org/2001/04/xmlenc#sha256	W3C Recommendation XML Encryption Syntax and Processing. 10 December 2002 [31]

11.2.2 Signature algorithms

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

11.2.3 Signature suites

The signature suites are defined using the following URNs.

Table 16

Short object name	URN	Normative references
dsa-sha1	http://www.w3.org/2000/09/xmldsig#dsa-sha1	XML-Signature Syntax and Processing. W3C Recommendation [30]
rsa-sha1	http://www.w3.org/2000/09/xmldsig#rsa-sha1	XML-Signature Syntax and Processing. W3C Recommendation [30]
ecdsa-sha1	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1	RFC 4050 [28]
rsa-ripemd160	http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160	RFC 4050 [28]
rsa-sha256	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	RFC 4050 [28]

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

WHIRLPOOL

Signature suites based on a combination of a hash algorithm based on Whirlpool and a signature scheme do not yet have an OID. It would be desirable to have variants of Whirlpool with an output less than 512 bits in order to match the requirements of Elliptic Curves algorithms.

ECGDSA

The ecgdsa signature scheme and signature suites based on ecgdsa have an OID but no normative reference for the data structures defined by these OIDs.

ecgdsa

"Elliptic Curve German DSA"

{1(iso) 3(identified organization) 36(teletrust) 3(algorithm) 3(signature algorithm) 2(ecSign) 5(ecgdsa)},

ecgdsaWithsha1

"Elliptic Curve German DSA with SHA1"

{1(iso) 3(identified organization) 36(teletrust) 3(algorithm) 3(signature algorithm) 2(ecSign) 6(ecgdsaWithsha1)},

ecgdsaWithRipemd160

"Elliptic Curve German DSA with RIPEMD160"

{1(iso) 3(identified organization) 36(teletrust) 3(algorithm) 3(signature algorithm) 2(ecSign) 7(ecgdsaWithRipemd160)}

NOTE: TeleTrust OIDs are available at http://www.teletrust.de/.

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

Whirlpool

Whirlpool has currently no URN. In addition, it would be desirable to have variants of Whirlpool with an output less than 512 bits in order to match the requirements of Elliptic Curves algorithms.

Signature suites based on a combination of a hash algorithm based on Whirlpool and a signature scheme do not yet have a URN.

ECGDSA

The ecgdsa signature scheme and signature suites based on ecgdsa do not have a URN, and thus no normative reference for the data structures.

Annex A (normative): Algorithms for various data structures

TS 101 733 [18] and TS 101 903 [19] define the formats of advanced electronic signatures. These two documents reference other documents defining various standardized data structures.

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

- Signer Certificates (RFC 3280 [2] and RFC 3279 [12]);
- Certificate Revocation Lists (RFC 3280 [2] and RFC 3279 [12]);
- OCSP responses (RFC 2560 [22]);
- Certification Authority Certificates (RFC 3280 [2] and RFC 3279 [12]);
- Self-signed certificates for CA certificates (RFC 3280 [2] and RFC 3279 [12]);
- Time-Stamping Tokens (TSTs) (RFC 3161 [9] and TS 101 861 [20]);
- Time-Stamping Unit certificates (RFC 3161 [9] and TS 101 861 [20]);
- Self-signed certificates for TSU Certificates (RFC 3280 [2] and RFC 3279 [12]);
- Attribute Certificates (ACs) (RFC 3280 [2] and RFC 3279 [12]);
- Attribute Authority Certificates (RFC 3281 [21]).

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.

Each set is identified by an identifier which is either an OID (Object IDentifier) or a URI /URN. The use of such identifiers is necessary so that interoperability can be achieved. In order to allow for data interchange, the document references algorithms in terms of OIDs and URIs / URNs together with algorithm parameters.

The algorithms which MAY be supported by issuers or users are NOT indicated.

A.1 Advanced Electronic Signatures based on TS 101 733

An advanced electronic signature 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 must be consistent with the identifier of the signature algorithm contained in the signer's certificate.

Requirements apply both to the hash function and the signature algorithm.

Since TS 101 733 [18] is built upon RFC 3852 [23], the algorithm requirements defined in RFC 3270 [24] apply. At that time the MD5 hash functions was recommended. Since it has been broken in August 2004, it is no more mentioned.

Table A.1

AdES based on TS 101 733 [18]	*Issuers* of AdES	*Users* of AdES
Hash functions	SHOULD support sha1	SHALL support sha1
Signature algorithms	SHOULD support RSA or SHOULD support DSA or SHOULD support ECDSA	SHALL support RSA SHOULD support DSA SHOULD support ECDSA

A.2 Advanced Electronic Signatures based on TS 101 903

TS 101 903 [19] uses a URN to reference the hash function in the ds:DigestMethod element. Since TS 101 903 [19] is built upon XML DigSig, the algorithm requirements from XML DigSig apply.

Table A.2

AdES based on TS 101 903 [19]	*Issuers* of AdES	*Users* of AdES
Hash functions	SHOULD support sha1	SHALL support sha1
Signature algorithms	SHOULD support DSAwithSHA1 MAY support RSAwithSHA1, or ECDSA	SHALL support DSAwithSHA1 SHOULD support RSAwithSHA1, or ECDSA

NOTE: For canonicalization:

1. Required Canonical XML (omits comments)
   http://www.w3.org/TR/2001/REC-xml-c14n-20010315
2. Recommended Canonical XML with Comments
   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. The algorithm requirements from RFC 3279 [12] apply. These requirements apply to signer public keys and CA issuing keys.

Table A.3

Signer certificates	*Issuers* of signer certificates	*Users* of signer certificates
Signer public keys	SHOULD support RSA SHOULD support DSA SHOULD support ECDSA	SHALL support RSA SHOULD support DSA SHOULD support ECDSA
CA issuing keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1

A.4 CRLs

A CRL is signed by a CRL Issuer. The algorithm requirements from RFC 3279 [12] apply. These requirements apply to CRL Issuer public keys.

Table A.4

CRLs	*Issuers* of CRLs	*Users* of CRLs
CRL issuer keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1

A.5 OCSP responses

A CRL is signed by an OCSP responder. The algorithm requirements from RFC 2560 [22] apply, i.e. "Clients that request OCSP services SHALL be capable of processing responses signed used DSA keys identified by the DSA sig-alg-oid specified in clause 7.2.2 of RFC 3280 [2]. Clients SHOULD also be capable of processing RSA signatures as specified in clause 7.2.1 of RFC 3280 [2]. OCSP responders SHALL support the SHA1 hashing algorithm." These requirements apply to OCSP the hash algorithm and the signature algorithm used by OCSP responders.

NOTE: RFC 2459 is mentioned in RFC 2560 [22], but has been obsoleted by RFC 3280 [2].

Table A.5

OCSP response	*Issuers* of OCSP responses	*Users* of OCSP response
OCSP responder keys	SHOULD support sha1 with dsa SHOULD support sha1 with rsa	SHALL support sha1 with dsa SHOULD support sha1 with rsa

A.6 CA certificates

A CA certificate contains a CA public key and is signed by a CA private key. The algorithm requirements from RFC 3279 [12] apply. These requirements apply to CA public keys (as subject) and CA public keys (as issuer).

Table A.6

CA certificates	*Issuers* of CA certificates	*Users* of CA certificates
Subject CA public key	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1
Issuer CA public keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1

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

A self-signed certificate contains a single root CA public key. The algorithm requirements from RFC 3279 [12] apply. Self-signed certificates need to resist quite long (e.g. more than 10 years). For that reason, the SHA-224, used in combination with RSA is also recommended. These requirements apply to root CA public keys.

Table A.7

Self-signed certificates	*Issuers* of self-signed certificates	*Users* of self-signed certificates
Root CA public keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1 SHOULD support RSA with SHA-224	SHALL support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support ECDSA with SHA1 SHOULD support RSA with SHA-224

A.8 TSTs based on RFC 3161 and TS 101 861

The following requirements apply to hash functions and TST signature algorithms. The algorithm requirements from TS 101 861 [20] apply. However, MD5 which was in the list has been dropped, since it has been broken in August 2004:

- for the requests: The following hash algorithms MAY be used to hash the information to be time-stamped: SHA-1, RIPEMD-160;

- for the responses, the following signature algorithm must be supported: SHA-1 with RSA.

However, for time-stamp tokens that need to resist quite long (e.g. more than 10 years), the SHA-2 family (SHA-256, SHA-384 and SHA-512) is recommended.

Table A.8

Time-Stamping Tokens	TST requesters	TST issuers	TST verifiers
Hash function	SHOULD support Sha1 SHOULD support ripemd160	SHALL support Sha1 SHOULD support ripemd160	SHALL support Sha1 SHOULD support ripemd160
TST signature algorithms	SHALL support sha1 with rsa	SHALL support sha1 with rsa	SHALL support sha1 with rsa

A.9 TSU certificates

A TSU certificate contains a TSU public key and is signed by a CA private key. The algorithm requirements from RFC 3279 [12] apply. These requirements apply to TSU public keys (as subject) and CA public keys (as issuer).

Table A.9

TSU certificates	*Issuers* of TSU certificates	*Users* of TSU certificates
TSU public key	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1
Issuer CA public keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1

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

A self-signed certificate contains a single root CA public key. The algorithm requirements from RFC 3279 [12] apply. Self-signed certificates need to resist quite long (e.g. more than 10 years). For that reason, the SHA-224, used in combination with RSA is also recommended.

These requirements apply to root CA public keys.

Table A.10

Self-signed certificates	*Issuers* of self-signed certificates	*Users* of self-signed certificates
root CA public keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support RSA with SHA-224	SHALL support RSA with SHA1 SHOULD support DSA with SHA1 SHOULD support RSA with SHA-224

A.11 Attribute certificates

An Attribute Certificate is signed by an Attribute Authority. The algorithm requirements from RFC 3279 [12] apply. These requirements apply to Attribute Authority public keys.

Table A.11

Attribute Certificates	***Issuers* of OCSP Attribute Certificates**	***Users* of OCSP Attribute Certificates**
Attribute Authority public keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1

A.12 AA certificates

An AA certificate contains an Attribute Authority public key and is signed by a CA private key. The algorithm requirements from RFC 3279 [12] apply. These requirements apply to Attribute Authority public keys (as subject) and CA public keys (as issuer).

Table A.12

AA certificates	***Issuers* of AA certificates**	***Users* of AA certificates**
Attribute Authority public key	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1
Issuer CA public keys	SHOULD support RSA with SHA1 SHOULD support DSA with SHA1	SHALL support RSA with SHA1 SHOULD support DSA with SHA1

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

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

2005-04: Taking in consideration the recently published attacks on hash functions, the clause 9.3 in the main body is updated. The former text is provided here and the values that have been changed are printed in bold and underlined.

10.3 (2005-04) Recommended hash functions versus time

Tables 6 and 7 provides indication about recommended hash functions during X years. With respect to the above distinction between conservative and liberal views, the conservative view is first provided and then the liberal.

Definitions:

Usable: The algorithm with the given security parameters can be considered secure at the given time.

Unknown: The security of the algorithm is unknown; use in this case is environment dependent.

Table 6/clause 9.3: Conservative view of recommended hash functions for a resistance during X years

entry name of the hash function	3 years (2008)	5 years (2010)	10 years (2015)
sha1	usable	unknown	**unknown**
ripemd160	usable	**usable**	unknown
sha224	usable	usable	usable
sha256	usable	usable	usable
Whirlpool	usable	usable	unknown

Table 7/clause 9.3: Liberal view for recommended hash functions for a resistance during X years

entry name of the hash function	3 years (2008)	5 years (2010)	10 years (2015)
sha1	usable	usable	unknown
ripemd160	usable	**usable**	unknown
sha224	usable	usable	usable
sha256	usable	usable	usable
whirlpool	usable	usable	unknown

Table B.3 predicts hash function resistance over 20 years. Due to the inherent unpredictability of such predictions, these predictions are largely speculative, and no distinction is made be between conservative and liberal.

Additional definition:

Unusable: The algorithm cannot be considered secure for any kind of use in the context of electronic signatures.

Table 8/clause 9.3: Speculated hash function resistance over 20 years, based on current trends and estimated computational power

entry name of the hash function	Speculated Usability in 20 years (2025)
sha1	unusable
ripemd160	unusable
sha224	usable
sha256	usable
whirlpool	usable

2005-04: Taking in consideration the recently published attacks on hash functions, the clause 9.4 in the main body is updated. The former text is provided here and the values that have been changed are printed in bold and underlined.

10.4 (2005-04) Recommended key sizes versus time

Tables B.4 and B.5 provides indication about recommended key lengths for a resistance of the algorithm or the signature suite during X years. A conservative view and a liberal view are provided. For explanations of these terms please refer to clauses 2 and 3 in this clause. These tables are provided for 3 years, 5 years and 10 years.

Definitions:

Unusable: The algorithm cannot be considered secure for any kind of use in the context of electronic signatures.

Unknown: The security of the algorithm is unknown at this time.

Table 9/clause 9.4: Conservative view of recommended key lengths for a resistance during X years

entry name of the signature suite	3 years (2008)	5 years (2010)	10 years (2015)
sha-1-with-rsa	≥ 768	unknown	**unknown**
sha224-with-rsa	≥ 768	1024	2048
sha256-with-rsa	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA1Identifier	≥ 768	1024	**unknown**
RSASSA-PSS with mgf1SHA224Identifier	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA256Identifier	≥ 768	1024	2048
sha1-with-dsa	≥ 768	unknown	**unknown**
sha1-with-ecdsa	163	unknown	**unknown**
sha224-with-ecdsa	224	224	224
sha256-with-ecdsa	256	256	256

Table 10/clause 9.4: Liberal view of recommended key lengths for a resistance during X years

entry name of the signature suite	3 years (2008)	5 years (2010)	10 years (2015)
sha-1-with-rsa	≥ 768	**1024**	unknown
sha224-with-rsa	≥ 768	1024	2048
sha256-with-rsa	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA1Identifier	≥ 768	1024	unknown
RSASSA-PSS with mgf1SHA224Identifier	≥ 768	1024	2048
RSASSA-PSS with mgf1SHA256Identifier	≥ 768	1024	2048
sha1-with-dsa	≥ 768	**1024**	unknown
sha1-with-ecdsa	163	**190**	unknown
sha224-with-ecdsa	224	224	224
sha256-with-ecdsa	256	256	256

An additional table with speculations regarding security over 20 years is provided. Due to the unpredictability in such long term statements, there is no distinction made between liberal and conservative for the 20 year prediction, and such information should be considered as largely speculative.

Table 11/clause 9.4: Speculated resistance of recommended algorithm parameters for the next 20 years, based on current trends and estimated computational power

entry name of the signature suite	**20 years (2025)**
sha-1-with-rsa	unusable
sha224-with-rsa	2048
sha256-with-rsa	2048
RSASSA-PSS with mgf1SHA1Identifier	unusable
RSASSA-PSS with mgf1SHA224Identifier	2048
RSASSA-PSS with mgf1SHA256Identifier	2048
sha1-with-dsa	**2048**
ecPublicKey	unknown
sha1-with-ecdsa	**unknown**

Annex C (informative): Generation of RSA modulus

An RSA modulus is obtained by multiplying two prime numbers of roughly the same size. Furthermore, the two factors must not be too close in order to be far enough from the square root of the modulus.

If we let p and q be the two prime factors of the modulus n, we can require that, for example:

$$0{,}1 < |\log_2(p) - \log_2(q)| < 30$$

which means that none of the factors is small or close to the square root of the modulus. This condition implies that:

$$\log_2(n)/2 - 15 < \log_2(p), \log_2(q) < \log_2(n) / 2 + 15$$

The generation of an RSA modulus of exactly k bits could be done with the following algorithm:

- Choose a random prime number p in the range $]2^{k/2-15}, 2^{k/2+15}[$;
- Choose a random prime number q in the range $[2^{k-1}/p, 2^k/p[$;
- If the condition $0.1 < |\log_2(p)-\log_2(q)| < 30$ is not satisfied, go back to the first step;
- Let n be the product of p and q.

A more complicated method that avoids the third step altogether but produces differently distributed primes is:

- Choose a random prime number p in the range $[2^{k/2-9/20}, 2^{k/2+15}[$;
- Choose a random prime number q in the range$]a,b[$ where a=max(ceil($2^{k-1}/p$)-1, $p.2^{-30}$) and b=min($2^k/p$, $p.2^{-1/10}$);
- Let n be the product of p and q.

Annex D (informative): Generation of elliptic curve domain parameters

This annex describes possible ways to generate elliptic curve domain parameters for ECDSA and ECGDSA satisfying the conditions given in clauses 6.1.2.3 and 6.1.2.4 and also ways to select curves verifiably at random.

For this, basically the algorithms described in ANSI X9.62 [7] annex A.3 can be used. The only necessary modifications are due to the fact that two of the conditions imposed on elliptic curves in this TS where not included in ANSI X9.62 [7]: In step 4. of algorithm A.3.2 only the condition about r_0 (which is equivalent to the "MOV condition" of ANSI X9.62 [7]) and the condition $p \neq q$ of clause 6.1.2.3 (which follows from the "anomalous condition" of ANSI X9.62 [7]) are ensured while the condition on the class number of the maximal order of the endomorphism ring of the curve and (in the F_{2^m} case) the condition that the curve must not be definable over F_2 are not respected by the algorithm. These latter two conditions should also be checked during the generation of the curve. Clauses D.1 and D.2 describe in more detail how the algorithm A.3.2 of ANSI X9.62 [7] can be modified accordingly. The algorithms A.3.3.1 and A.3.3.2 of ANSI X9.62 [7] for selecting curves verifiably at random which basically produce random *j*-invariants from a seed by means of a hash function only need to be modified in the case that the security of SHA-1 is no longer regarded to be sufficient.

Clause D.3 gives more information about the class number condition and in particular describes how it can be checked. For checking the class number condition an integer M has to be factorized and a certain triple (α,β,γ) of integers has to be found. To ease the validation of the domain parameters the prime factorization of M and the triple (α,β,γ) should always be made public together with the domain parameters. Otherwise in particular the factorization of M would consume much time.

NOTE: Additional standard curves and further information about the class number condition can be found in the ECC Brainpool publication "ECC Brainpool Standard Curves and Curve Generation"; OID:

- {1(iso) 3(identified organization) 36(teletrust) 3(algorithm) 3(signature algorithm) 2(ecSign) 8(ecStdCurvesAndGeneration)}.

D.1 ECDSA and ECGDSA based on a group $E(F_p)$

The prime *p* can be generated by one of the algorithms described in ISO/IEC 18032 (see bibliography) in a way that the probability of being composite is at most 2^{-100}.

The generation of an appropriate curve *E*, the point *P* and the prime *q c*an be done with the algorithm in annex A.3.2 of ANSI V9.62 [7] with lower bound $r_{min} > 2^{qMinLen}$, with MOV threshold *B*= r0Min and with Step 4. of algorithm A.3.2 substituted by: "4.

- Check the MOV condition (see annex A.1.1) with inputs *B*, *q* and *n*. If the result is "false" go to Step 1.
- Check the Anomalous condition (see annex A.1.2). If the result is "false" go to Step 1.
- Find an element in the ideal class group of the number field K:=Q($\sqrt{-d}$) of order at least MinClass where d is the squarefree factor of $M := 4p - (p+1-\#E(F_p))^2$. If such an element of the class group cannot be found go to Step 1."

The number M is always a positive integer and by the prime factorization of M one determines uniquely defined positive integers d,l with $M = dl^2$. Then d is called the squarefree factor of M .

If an element of the ideal class group of order at least MinClass can be found then the class number condition of clause 6.1.2.3 is satisfied.

Elements of the ideal class group of K can effectively be represented by certain triples (α,β,γ) of integers. The details in particular about the group operation and the neutral element in this set of triples are given in clause D.3.

Here it is neither specified how to choose the elements in the ideal class group which are checked for sufficiently high order nor after how many unsuccessful selections of elements to decide that the demanded element "cannot be found". This is another reason to attach the triple (α,β,γ) (which has an order at least MinClass) to the domain parameters for validation.

The algorithm to generate *E*, *P* and *q* can be successful only if *p* is chosen large enough, i.e. at least about as large as *q* which itself is greater than $2^{\text{qMinLen-1}}$.

To select a curve *verifiably at random* one can use the algorithm given in annex A.3.3.1 of ANSI X9.62 [7]. The hash function SHA-1 has to be substituted by a more secure hash function after the recommended use date for SHA-1.

D.2 ECDSA and ECGDSA based on a group $E(F_{2^m})$

The selection of an appropriate curve *E*, the point *P* and the prime *q* can be done with the algorithm in annex A.3.2 of ANSI X9.62 [7] with lower bound $r_{min} > 2^{\text{qMinLen}}$, with MOV threshold *B*=r0Min and with Step 4. of algorithm A.3.2 substituted by: "4.

- Verify that $b \neq 1$. If this is not the case go to Step 1.
- Check the MOV condition (see annex A.1.1) with inputs *B*, *q* and *n*. If the result is "false" go to Step 1.
- Find an element in the ideal class group of the number field K:=Q($\sqrt{-d}$) of order at least MinClass where d is the squarefree factor of $M := 2^{m+2} - (2^m + 1 - \#E(F_{2^m}))^2$. If such an element in the ideal class group cannot be found go to Step 1".

The parameter b is the constant term in the representation $y^2+xy=x^3+ax^2+b$ of the curve *E* used in the algorithm. b is the *j*-invariant of E and as in a former step of the algorithm $b \neq 0$ was already checked $b \neq 1$ means that the *j*-invariant is not contained in F_2 and in particular that *E* cannot be defined over F_2.

The number M is always a positive integer and by the prime factorization of M one determines the uniquely defined positive integers d, l with $M = dl^2$. Then d is called the squarefree factor of M.

If an element of the ideal class group of order at least MinClass can be found then the class number condition of clause 6.1.2.4 is satisfied.

Elements of the ideal class group of K can effectively be represented by certain triples (α,β,γ) of integers. The details in particular about the group operation and the neutral element in this set of triples are given in clause D.3.

Here it is neither specified how to choose elements in the ideal class group which are checked for sufficiently high order nor after how many unsuccessful selections of elements to decide that the demanded element "cannot be found". This is another reason to attach the triple (α,β,γ) (which has an order at least MinClass) to the domain parameters for validation.

The algorithm to generate *E* and *P* can be successful only if 2^m is chosen large enough, i.e. at least about as large as *q* which itself is greater than $2^{\text{qMinLen-1}}$.

To select a curve *verifiably at random* one can use the algorithm given in annex A.3.3.1 of ANSI X9.62 [7]. The hash function SHA-1 has to be substituted by a more secure hash function after the recommended use date for SHA-1.

D.3 The class number condition

The class number condition was introduced because of the following reason: A hypothetical lift of the curve to an elliptic curve over a number field cannot exist if the degree of the number field is less then the class number of the endomorphism ring End(E) of E (regarded as order in the imaginary quadratic number field K:=Q($\sqrt{-d}$) defined in clauses D.1 and D.2 respectively) and if the degree of the number field is large then a solution of the corresponding lift of the elliptic curve discrete logarithm problem is not feasible. The class number of End(E) is always a multiple of the class number of K so what is actually demanded is a sufficiently large class number of K. Also the recent results of Huang and Raskind (see bibliography) can be regarded as arguments for demanding a sufficiently large class number of K.

Because the complexity of the best known algorithms for explicitly determining the class number of K is too high in practice one just tries to find elements of the ideal class group of K with a large order as the class number is not smaller than the order of an element.

Randomly selected curves will violate the class number condition with very low probability. But the best known rigorously proven upper bounds do not exclude the possibility that the actual probability is significantly higher than 2^{-80} and heuristic arguments show that this probability should be at least of about the same magnitude as 2^{-80} (for a key length of approximately 160 bits). So the class number condition should be checked also for randomly selected curves.

We now briefly describe how the elements of the ideal class group of the number field K:=Q($\sqrt{-d}$) for a positive squarefree integer d can be represented, how to determine the product of two elements and what the representation of the neutral element with respect to this product looks like. Details can be found in the text book of Cohen (see bibliography).

First define $D := \begin{cases} -d & if \quad -d \equiv 1(4) \\ -4d & if \quad -d \equiv 2(4) \quad or \quad -d \equiv 3(4) \end{cases}$

A triple (α,β,γ) of integers satisfying =:

- gcd(α,β,γ)=1;
- $\alpha>0$ and $|\beta| \leq \alpha \leq \gamma$ and if $\alpha=\gamma$ or $|\beta|=\alpha$ then also $\beta \geq 0$;
- $\beta^2-4\alpha\gamma= D$;

is called a ***primitive reduced triple of discriminant*** D .

NOTE 1: (α,β,γ) is a primitive reduced triple of discriminant D if and only if the quadratic form $\alpha x^2+ \beta xy+\gamma y^2$ is primitive reduced and has discriminant D (which in particular implies that it is positive definite).

The elements of the ideal class group of the number field K:=Q($\sqrt{-d}$) correspond one-to-one to the primitive reduced triples of discriminant D . The group operation in this set of triples can be calculated as follows.

Given two primitive reduced triples $(\alpha_1,\beta_1,\gamma_1)$ and $(\alpha_2,\beta_2,\gamma_2)$ the so called composition (α',β',γ') of $(\alpha_1,\beta_1,\gamma_1)$ and $(\alpha_2,\beta_2,\gamma_2)$ can be determined by algorithm 5.4.7 of Cohen's book (see bibliography). This triple (α',β',γ') is primitive and has discriminant D but is not necessarily reduced. Applying the reduction algorithm 5.4.2 of the same book to this triple (α',β',γ') delivers a primitive reduced triple (α,β,γ) with determinant D . This triple represents the product of the two elements representing $(\alpha_1,\beta_1,\gamma_1)$ and $(\alpha_2,\beta_2,\gamma_2)$.

$$(\alpha_1,\beta_1,\gamma_1) \circ (\alpha_2,\beta_2,\gamma_2) := (\alpha,\beta,\gamma)$$

The neutral element is represented by the triple $(1,0,-D/4)$ if $D \equiv 0(4)$ and it is represented by the triple $(1,1,(1-D)/4)$ if $D \equiv 1(4)$. In either case the triple corresponding to the neutral element shall be denoted **I.**

The following is an algorithm that determines whether the order of an element of the ideal class group of the number field K:=Q($\sqrt{-d}$) has an order of at least MinClass:

Input: A primitive reduced triple (α,β,γ) of discriminant D .

Output: The message "true" if the order of the corresponding element of the ideal class group is at least MinClass; the message "false" otherwise.

```
1)  Set t=I.
2)  For i from 1 to MinClass-1 do:
      -  Set t:=t ∘ (α,β,γ).
      -  If t=I then output "false" and stop.
3)  Output "true".
```

In 2. the triple t ∘ (α,β,γ) is calculated by the procedure described above.

NOTE 2: Most of the common computer algebra packages contain implementations for the described manipulations in the class group of K or in the set of primitive reduced quadratic forms respectively.

Annex E (informative): On the generation of random data

E.1 Classes of random number generators

Figure E.1 shows a schematic classification of random number generators according to ISO/IEC FCD 18031 (see bibliography) where more detailed information can be found. That document uses the term "random bit generator" while here the term "random number generator" is used.

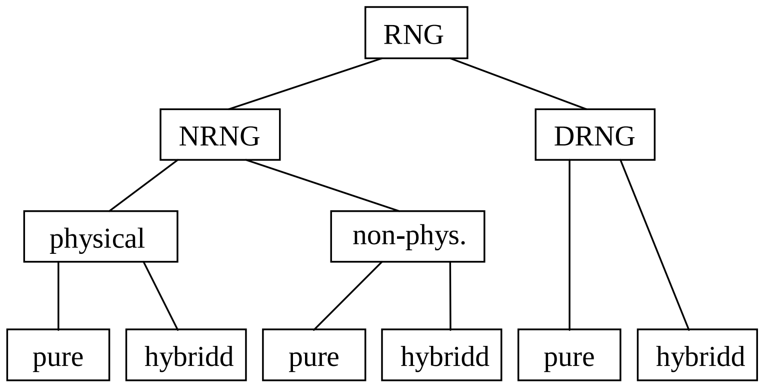

Figure E.1: schematic classification of random number generators according to ISO/IEC FCD 18031

Every random number generator RNG must have a primary entropy source. If this entropy source is non-deterministic which means unrepeatable and unpredictable the RNG is called non-deterministic or a NRNG. If this entropy source consists just of seed values it is called deterministic and also the RNG is called deterministic or a DRNG.

The primary entropy source of a NRNG can either be physical or non-physical. A physical primary entropy source (also called physical primary noise source) uses dedicated hardware to measure the physical characteristics of a sequence of events in the physical world, e.g. radioactive emissions of atoms or the noise of diodes. Typical non-physical primary entropy sources are based for example on RAM contents, system clocks or "random user inputs" via PC-keyboard or PC-mouse.

If the only entropy source for a RNG is the primary entropy source it is called pure RNG. A RNG can also have an additional entropy source. A NRNG with an additional deterministic entropy source (i.e. seed values) is called hybrid NRNG. A DRNG with an additional non-deterministic entropy source is called hybrid DRNG.

A well constructed NRNG is information theoretically secure while (pure) DRNGs can only be complexity theoretically secure that means there is no *feasible* way to break its security. The advantage of the former is obviously that there is no (even theoretical) possibility to calculate future or previous outputs from known ones. The security of DRNGs depends on assumptions about the algorithmic complexity of certain problems which may turn out to be wrong sooner or later. So NRNGs are better suited for long term security.

The following terminology for DRNGs is used in clause 8.2.2:

- A **re-seeding** of a DRNG is a complete new initialization of the DRNG with a newly produced seed.
- A **seed-update** of a DRNG is an external modification of the internal state (not by the regular updating function of the DRNG) in a way that: (i) After the modification the modifier has no more information about the internal state than before. (ii) Anybody else than the modifier having some information about the previous internal state has *less* information about the internal state after the modification.

NOTE: The difference between these two possible ways to add new entropy to a DRNG is that if the new seed is known then the future output is known after re-seeding while this is not the case for a seed-update.

Of course it is desirable that after a seed-update the loss of information about the internal state in condition (ii) should be as large as possible. In an ideal case there remains *no* information. A typical example for that is an XOR of the internal state with a new seed produced with a strong NRNG.

E.2 On tests for NRNGs

Examples of generic test suites for the statistical properties of the primary entropy source can be found in *Ruhkin et al.* (see bibliography). But usually tests specifically adapted to the mathematical model of the source are more suitable.

Online tests should be specific to the primary entropy source. An example for such an online test can be found in Example E7 of AIS 31 (see bibliography).

To avoid a misunderstanding about tests and test suites it should be pointed out that:

There is no test or test suite which can show that the output of a generator has a certain minimum entropy without certain additional statistical assumptions about the source.

EXAMPLE: The term "universal" for Maurer's test (Maurer 1991, see bibliography) could cause some confusion about this fact. Actually in Maurer's article it is assumed that the source is a binary, stationary, ergodic source with finite memory. These assumptions are explicitly mentioned in that article.

It should also be observed that an NRNG has to be evaluated as an entire system i.e. taking into account the interaction of the components. Thus it is not enough to regard the mathematical model, the online tests and the evaluation of the post-treatment separately.

Annex F (informative): Algorithms identifiers defined in various documents

F.1 Algorithms identifiers defined in RFC 3278

The title of the document is: "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)" by S. Blake-Wilson, D. Brown, P. Lambert [11].

Signature suite

ECDSA with SHA1

```
ecdsa-with-SHA1  OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) 10045 signatures(4) 1 }
```

F.2 Algorithms identifiers defined in RFC 3279

The title of the document is: "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" by W. Polk, R. Housley, L. Bassham. [12]

Signature suites for CA issuing keys and CRL issuing keys

RSA with SHA1

```
sha-1WithRSAEncryption OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }
```

DSA with SHA1

```
id-dsa-with-sha1 OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 3 }
```

ECDSA with SHA1

```
ecdsa-with-SHA1  OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) 10045 signatures(4) 1 }
```

Preferred signature algorithms for subject public keys (any is allowed)

RSA

```
rsaEncryption OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 1}
```

DSA

```
id-dsa OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) x9-57(10040) x9cm(4) 1 }
```

ECDSA

```
id-ecPublicKey OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) 10045 2 1 }
```

F.3 Algorithms identifiers defined in RFC 3370

The title of the document is: "Cryptographic Message Syntax (CMS) Algorithms" [13].

Hash-functions

```
sha-1 OBJECT IDENTIFIER ::=
{ iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }
```

Signature suite

DSA is always used with the SHA-1 message digest algorithm. The algorithm identifier for DSA is:

```
id-dsa-with-sha1 OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 3 }
```

F.4 Algorithms identifiers defined in RFC 3447

The title of the document is: "PKCS #1: RSA Cryptography Specifications" [14].

Signature algorithm

The algorithm identifier **for RSA is:**

```
rsaEncryption OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }
```

F.5 Algorithm identifier defined in RFC 3874

The title of the document is: "A 224-bit One-way Hash Function: SHA-224" [27].

id-sha224

```
id-sha224  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
csor(3) nistalgorithm(4) hashalgs(2) sha224(4) }
```

F.6 Algorithms identifiers defined in XML-Signature Syntax and Processing W3C Recommendation

This recommendation from February 12, 2002 is about "XML-Signature Syntax and Processing" [30].

Hash-function

SHA-1: http://www.w3.org/2000/09/xmldsig#sha1

Signature suite

DSAwithSHA1 (DSS): http://www.w3.org/2000/09/xmldsig#dsa-sha1 (Required)

RSAwithSHA1 (RSA): http://www.w3.org/2000/09/xmldsig#rsa-sha1 (Recommended)

F.7 Algorithms identifiers defined in XML Encryption Syntax and Processing. W3C Recommendation

This recommendation from December 10, 2002 is about "in XML Encryption Syntax and Processing " [31].

Hash-functions

SHA1: http://www.w3.org/2000/09/xmldsig#sha1

SHA256: http://www.w3.org/2001/04/xmlenc#sha256

RIPEMD-160: http://www.w3.org/2001/04/xmlenc#ripemd160

F.8 Algorithms identifiers defined in RFC 4050

The title of the document is: "Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures" [28].

Signature suite

ECDSA: http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1

NOTE: Like DSA, ECDSA incorporates the use of a hash function. Currently, the only hash function defined for use with ECDSA is the SHA-1 message digest algorithm.

F.9 Algorithms identifiers defined in RFC 4051

The title of the document is: "Additional XML Security Uniform Resource Identifiers (URIs)" [29]

SHA-224

http://www.w3.org/2001/04/xmldsig-more#sha224

RSA-SHA256

http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

RSA-RIPEMD160

http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160

ECDSA-SHA

http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1

http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha224

http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256

F.10 Algorithms identifiers defined in RFC 4055

The title of the document is: "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure. Certificate and Certificate Revocation List (CRL) Profile".

RFC 4055 [15] supplements RFC 3279 [12] to describe how to use some newer cryptographic algorithms.

Hash-functions

```
id-sha224  OBJECT IDENTIFIER  ::=  {{ joint-iso-itu-t(2) country(16) us(840) organization(1)
gov(101) csor(3) nistalgorithm(4) hashalgs(2) 4 }
id-sha256  OBJECT IDENTIFIER  ::= { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
csor(3) nistalgorithm(4) hashalgs(2) 1 }
id-sha384  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
csor(3) nistalgorithm(4) hashalgs(2) 2 }
id-sha512  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
csor(3) nistalgorithm(4) hashalgs(2) 3 }
```

Mask Generation functions

```
mgf1SHA1Identifier  AlgorithmIdentifier  ::= { id-mgf1, sha1Identifier }
mgf1SHA224Identifier  AlgorithmIdentifier  ::= { id-mgf1, sha224Identifier }
mgf1SHA256Identifier  AlgorithmIdentifier  ::= { id-mgf1, sha256Identifier }
mgf1SHA384Identifier  AlgorithmIdentifier  ::= { id-mgf1, sha384Identifier }
mgf1SHA512Identifier  AlgorithmIdentifier  ::= { id-mgf1, sha512Identifier }
```

Signature algorithms

```
id-RSASSA-PSS  OBJECT IDENTIFIER  ::=  { pkcs-1 10 }
```

Signature suites

```
sha224WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 14 }
sha256WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 11 }
sha384WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 12 }
sha512WithRSAEncryption  OBJECT IDENTIFIER  ::=  { pkcs-1 13 }
```

Annex G (informative): Abstracts of ISO/IEC 10118-3 and ISO/IEC 9796-2

Abstract of ISO/IEC 10118-3 [3]

ISO/IEC 10118-3 [3] specifies the following seven dedicated hash-functions, i.e. specially-designed hash-functions:

1) RIPEMD-160 in clause 7 provides hash-codes of lengths up to 160 bits;
2) RIPEMD-128 in clause 8 provides hash-codes of lengths up to 128 bits;
3) SHA-1 in clause 9 provides hash-codes of lengths up to 160 bits;
4) SHA-256 in clause 10 provides hash-codes of lengths up to 256 bits;
5) SHA-512 in clause 11 provides hash-codes of lengths up to 512 bits;
6) SHA-384 in clause 12 provides hash-codes of a fixed length, 384 bits; and
7) WHIRLPOOL in clause 13 provides hash-codes of lengths up to 512 bits.

For each of these dedicated hash-functions, ISO/IEC 10118-3 [3] specifies a round-function that consists of a sequence of sub-functions, a padding method, initializing values, parameters, constants, and an object identifier as normative information, and also specifies several computation examples as informative information.

Abstract of ISO/IEC 9796-2 [17]

ISO/IEC 9796-2 [17]:2002 specifies three digital signature schemes giving message recovery, two of which are deterministic (non-randomized) and one of which is randomized. The security of all three schemes is based on the difficulty of factorizing large numbers. All three schemes can provide either total or partial message recovery.

The method for key production for the three signature schemes is specified in ISO/IEC 9796-2 [17]. However, techniques for key management and for random number generation (as required for the randomized signature scheme), are outside the scope of ISO/IEC 9796-2 [17].

Wherever possible, the second mechanism (Digital signature scheme 2) is RECOMMENDED. However, in environments where generation of random variables by the signer is deemed infeasible, then Digital signature scheme 3 is RECOMMENDED. Digital signature scheme 1 SHALL only be used in environments where compatibility is required with systems implementing the first edition of this International Standard.

Annex H (informative): Signature maintenance

An advanced electronic signatures SHOULD be verified according to a signature policy that meets the business needs.

NOTE: There may exist valid reasons under particular circumstances to use a signature policy different from the one which should normally be used. In such a case, the full implications must be understood and carefully weighted by the verifier.

A signature policy MAY 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 may 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 may not be secure anymore. Additional security measures need to be taken so that this can be done.

It may 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 may 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 may be defined in the signature policy itself or "elsewhere" in a set of rules called a "signature maintenance policy" which will allow to maintain the validity of advanced electronic signatures.

When there is an interest to be able to re-verify advanced electronic signatures under a given signature policy at a time where it is possible or likely that the algorithms and key lengths originally used will not be secure anymore, then a signature maintenance process MUST be applied to these advanced electronic signatures. The sooner the process is applied, the better. This process MAY need to be performed again and again when advanced electronic signatures need to be verified during a very long time period.

Annex I (informative): Major changes from previous versions

This annex is currently empty since it is the first version of the present document. It will later on contain a description of the major changes between the several versions so that an history can be easily be done.

Annex J (informative): Bibliography

- ECRYPT Yearly Report on Algorithms and Key Lengths. Revision 1.1, 17 March 2005 http://www.ecrypt.eu.org/documents/D.SPA.10-1.1.pdf

- ECRYPT Position Paper: Recent Collision Attacks on Hash Functions. Revision 1.1 17 February 2005 http://www.ecrypt.eu.org/documents/STVL-ERICS-2-HASH_STMT-1.1.pdf

- [LenstraVerheul] Selecting Cryptographic Key-sizes, A.K. Lenstra E.R. Verheul, Journal of Cryptology 14 (2001) pp 255-293.

- Dobbertin, H., Bosselaers, A. and Preneel, B., "RIPEMD-160: A strengthened version of RIPEMD", in *Fast software encryption, Proceedings of the Third International Workshop, Cambridge, UK, February 21-23, 1996*, pp. 71-82, D. Gollmann (ed.), LNCS 1039, Springer-Verlag, 1996.

- Rivest, R.L., "Finding four million random primes", in *Advances in Cryptology - Crypto '90*, pp. 625-626, A.J. Menezes (ed.), LNCS 537, Springer-Verlag, 1991.

- Blum, M. and Micali, S., "How to generate cryptographically strong sequences of pseudo-random bits", SIAM Journal on Computing, Vol. 4, No. 13, pp. 850-863, 1984.

- Rivest, R., Shamir, A. and Adleman, L., "A method for obtaining digital signatures and public key cryptosystems", *Communications of the ACM*, Vol. 21, No. 2, pp. 120-126, 1978.

- R.L. Rivest, R.D. Silverman, "Are 'Strong' Primes needed for RSA?", Peprint 1999 December 1, 1998 http://cnscenter.future.co.kr/resource/rsc-center/vendor-wp/RSA/sp2.pdf

- AIS 20: "Application Notes and Interpretation of the Scheme: Functionality classes and evaluation methodology for deterministic random number generators"., Version 1, Available at http://www.bsi.bund.de/zertifiz/zert/interpr/ais20e.pdf

- AIS 31: Functionality Classes and Evaluation Methodology for Physical Random Number Generators, Version 3.1 (25.09.2001); Available at http://www.bsi.bund.de/zertifiz/zert/interpr/trngk31e.pdf

- ANSI X9.17-1985 (1985): "Financial Institution Key Management (wholesale)".

- ANSI X9.82 Random Number Generation Parts 1-3, Drafts June 2004

- "Regulierungsbehörde für Post und Telekommunikation", Geeignete Kryptoalgorithmen, 5.7.2001, at http://www.regtp.de/imperia/md/content/tech_reg_t/digisign/29.pdf.

- The Austrian Federal Government, "Signaturverordnung - SigV", BGBl. II Nr. 30/2000.

- RSA Security, Inc., "RSA Laboratories' Frequently Asked Questions About Today's Cryptography", Version 4.1, July 2000, http://www.rsasecurity.com/rsalabs/faq/index.html.

- GMD, "SECUDE: Algorithms", SECUDE-5.1, November 1997 http://www.darmstadt.gmd.de/secude/Doc/htm/algs.htm.

- Galbraith, S.D., and Smart, N.P., "A Cryptographic Application of Weil Descent", in *Cryptography and Coding*, M. Walker (ed.), LNCS 1746, Springer-Verlag, 1999.

- Johnson, D.B., and Menezes, A.J., "Elliptic Curve DSA (ECDSA): An Enhanced DSA" athttp://www.certicom.com/research/wecdsa.html.

- OID tree structure, http://www.darmstadt.gmd.de/secude/Doc/htm/oidgraph.htm

- Damgard, I., Landrock P.and Pomerance C., "Average case error estimates for the strong probable prime test", Math. Comp., 61:177-194, 1993.

- Ruhkin, A. et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication 800-22 with revisions dated 15.05.2001; available at Menezes, A., van Oorschot, P. and Vanstone, S., "Handbook of Applied Cryptography (chapter 5)", CRC Press, 1997, http://www.cacr.math.uwaterloo.ca/hac.

- Knuth, D., "The art of computer programming - Volume 2 / Seminumerical algorithms (chapter 3)", Addison-Wesley, 1981.

- Maurer, U., "A universal statistical test for random bit generators", *Advances in Cryptology - Crypto '90*, LNCS 537, pp. 409-420, 1991.

- Kelsey, J., Schneier, B., Wagner, D. and Hall, C., "Cryptanalytic Attacks on Pseudorandom Number Generators", *Fast Software Encryption '98*, LNCS 1372, pp. 168-188, 1998.

- Wiener, M., "Cryptanalysis of short RSA secret exponents", *IEEE Transactions on Information Theory*, Vol. 36 1990.

- Boneh, D. and Durfee, G., "Cryptanalysis of RSA with private key less than $N^{0.292}$", *Proc. Eurocrypt '99*, LNCS, J. Stern (ed.), Springer-Verlag, 1999. Final version in IEEE Trans. Information Theory, Vol. 46 2000.

- Durfee, G. and Nguyen, P., "Cryptanalysis of RSA with short private exponent", *Proc. Asiacrypt '99*, LNCS, Springer-Verlag, 1999.

- M.D. Huang, W. Raskind "Global Methods for Discrete Logarithm Problems". See: http://www.cacr.math.uwaterloo.ca/conferences/2004/ecc2004/huang.pdf

- "A Unified Approach to the Discrete Logarithm Problem for the Multiplicative Group and for Elliptic Curves over finite Fields". September 20, 2004. See: http://www.cacr.math.uwaterloo.ca/conferences/2004/ecc2004/raskind.pdf

- ISO/IEC FCD 18031: "Information technology - Security techniques - Random bit generation".

- ISO/IEC FDIS 18032 (draft 2004): "Information technology - Security techniques - Prime number generation".

- J.S. Coron: "Optimal Security Proofs for PSS and Other Signature Suites", EUROCRYPT 2002, LNCS 2332, Springer, Berlin, 2002.

- H. Cohen: "A Course in Computational Algebraic Number Theory", Springer, Berlin, 1993.

- Common ISIS-MTT Specification for interoperable PKI applications. Part 6. Cryptographic Algorithms. Version 1.1. 16 March 2004

History

Document history		
V1.1.1	March 2003	Publication as SR 002 176
V1.2.1	July 2005	Publication