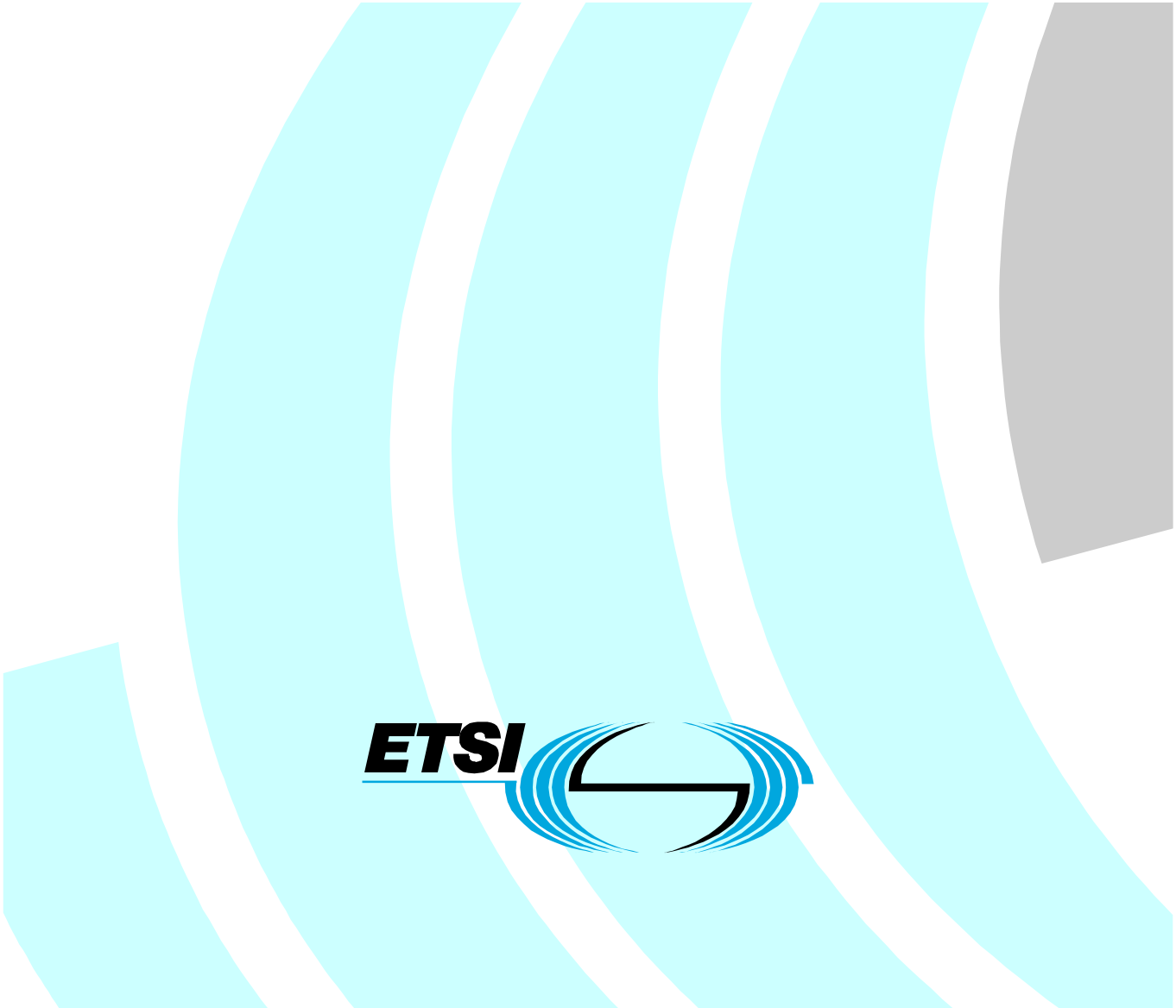
ETSI SR 002 176 V1.1.1 (2003-03)

Special Report

Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures



Reference
DSR/ESI-000016

Keywords
e-commerce, electronic signature, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.org

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2003.
All rights reserved.

DECT™, **PLUGTESTS**™ and **UMTS**™ are Trade Marks of ETSI registered for the benefit of its Members.
TIPHON™ and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.
3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

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).

All published ETSI deliverables shall include information which directs the reader to the above source of information.

Foreword

This Special Report (SR) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI).

Introduction

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

The present document will be handed to the EC via the EESSI-SG through ICTSB as a substantial contribution to be discussed further at the A9C (Article 9 Committee) level. The final decision on how to respect and how to handle this contribution and its future handling process will be the responsibility of the EC and A9C.

The present document defines a list of approved cryptographic algorithms together with the requirements on their parameters, as well as the approved combinations of algorithms in the form of "signature suites". The approved algorithms and parameters shall be referenced in the corresponding Protection Profiles (e.g. for SSCDs or trusted CSP components).

The present document contains several informative annexes which provide useful information on a number of subjects mentioned in the text.

1 Scope

The present document defines an initial set of algorithms and the corresponding parameters to be included in a list of approved methods for producing or verifying Electronic Signatures in Secure Signature-Creating Devices (SSCD) (EESSI-work area F: CWA 14168 / 14169 Secure Signature-Creation Devices), to be referenced in the Certificate Policy documents (EESSI-work area A: TS 101 456: Policy requirements for certification authorities issuing qualified certificates), during the signature creation and validation process and environment (EESSI-work area G1/2: CWA 14170: Security Requirements for Signature Creation Systems; CWA 14171 Procedures for Electronic Signature Verification), in trusted CSP components (Certification Service Provider) (EESSI-work-area D: CWA 14167-1: Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures) and other technical components and related areas.

The present document defines a list of approved cryptographic algorithms combined with the requirements on their parameters, as well as the approved combinations of algorithms in the form of "signature suites". The approved algorithms and parameters shall be referenced in the corresponding Protection Profiles (e.g. for SSCDs or trusted CSP components). To support the management activities, a numbering scheme for cryptographic algorithms and their parameters is defined.

Specifically, the present document gives guidance on

- management practices for cryptographic algorithms (see clause 4.1),
- signature suites (see clause 4.2),
- cryptographic hash functions (see clause 4.3),
- padding methods (see clause 4.4),
- signature algorithms and the corresponding key generation algorithms (see clause 4.5), and
- random-number generation (see clause 4.6).

The present document also gives guidance on the management practices that shall be applied to cope with developments such as the examples given in the annex A. It describes a process for updating the approved algorithms lists and parameters. Annex B contains OIDs assigned to the approved algorithms, annex C gives more information on the generation of RSA keys for signatures and annex D addresses the generation of random data.

Currently no algorithms are specified for symmetric encryption of critical data outside a secure device or for proving correspondence between the Signature Creation Data (SCD) and Signature Verification Data (SVD). Nonetheless such algorithms are within the scope of a subsequent version of the present document.

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] ISO/IEC 9979 (1999): "Information technology - Security techniques - Procedures for the registration of cryptographic algorithms".

[3] IETF RFC 2459 (1999): "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", Housley,R., et al.

[4] ISO/IEC 10118-3 (1998): "Information technology - Security techniques - Hash functions - Part 3: Dedicated hash functions".

[5] FIPS Publication 180-1 (1995): "Secure Hash Standard (SHS)".

[6] PKCS #1 v2.0 (1998): "RSA Cryptography Standard".

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

[8] FIPS Publication 140-1 (1994): "Security requirements for cryptographic modules".

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

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

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

[12] ISO/IEC 9796-3 (2000): "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 3: Discrete logarithm based mechanisms".

[13] ISO/IEC FCD 15946-2 (1999): "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures".

[14] ISO/IEC CD 15946-4 (2001): "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 4: Digital signatures giving message recovery".

[15] IETF RFC 1750 (1994): "Randomness Recommendations for Security", Eastlake, D., et al.

[16] ANSIX9.17-1985 (1985): "Financial Institution Key Management (wholesale)".

[17] PKCS #1 v2.1 draft 2 (2001): "RSA Cryptography Standard".

[18] Change Recommendation for ANSI X9.30-1995, (Part 1), Draft, April 2001.

[19] IETF RFC 3161 (2001): "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", Adams, C., et al.

3 Definitions and abbreviations

3.1 Definitions

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

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

Management Activity (MA): action that shall be taken by the electronic signature committee under the circumstances specified in clause 4 of SR 002 176

signature suite: combination of a signature algorithm with its parameters, a key generation algorithm, a padding method, and a cryptographic hash function

NOTE: The currently approved signature suites are specified in the present document.

3.2 Abbreviations

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

A9C	Article 9 Committee
CRL	Certification-Service-Provider
CRT	Chinese Remainder Theorem
CSP	Certification-Service-Provider
CWA	CEN Workshop Agreement
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
ECGDSA	Elliptic Curve German Digital Signature Algorithm
MA	Management Activity
OID	Object Identifier
PRNG	Pseudo Random Number Generator
RSA	Rivest, Shamir and Adleman Algorithm
SCD	Signature-Creation Data
SSCD	Secure-Signature-Creation Device
SVD	Signature-Verification Data
TRNG	True Random Number Generator

4 Algorithms and Parameters for Secure Electronic Signatures

4.1 Management activities

As a response to relevant developments in the area of cryptography and technology, activities for the management of the algorithms and parameters for secure electronic signatures shall enable dynamic updating of the lists of approved algorithms and parameters. The initial lists of approved algorithms and their parameters are given in the present document.

The management activities to introduce new algorithms and their parameters and to delete algorithms and their parameters from the list 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. Normal update cycles of the present document are not deemed to be fast enough to reflect the market need.

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

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

Through A9C, mechanisms shall be put in place that can react within 6 months for cases 1 and 2 and at most 1 month for case 3.

Further information on updating algorithms and parameters is contained in annex A.

4.2 Signature suites for secure electronic signatures

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 signature algorithm with parameters,
- a key generation algorithm,
- a padding method, and

- a cryptographic hash function.

If any of the components of a suite has been cancelled, the suite must be cancelled as well. If any of the components of a suite has been updated, the suite must be updated as well.

The list of currently approved signature suites is given in Table 1.

Each entry in the list of signature suites shall contain a date. This date represents the last day on which the algorithm suite is to be used for producing signatures. The dates will require revision well before they are due for the mentioned suite and should be reviewed at regular intervals, e.g. annually.

For all components only short names are used. Each component is defined in a separate table in the following clauses. Object Identifiers (OIDs) are specified in annex B.

Table 1: The list of approved signature suites

Signature suite entry index	Signature algorithm	Signature algorithm parameters	Key generation algorithm	Padding method	Cryptographic hash function	Valid until (signing)
001	rsa	MinModLen=1020	rsagen1	emsa-pkcs1-v1_5	sha1	31.12.2005
002	rsa	MinModLen=1020	rsagen1	emsa-pss	sha1	31.12.2005
003	rsa	MinModLen=1020	rsagen1	emsa-pkcs1-v1_5	ripemd160	31.12.2005
004	rsa	MinModLen=1020	rsagen1	emsa-pss	ripemd160	31.12.2005
005	dsa	pMinLen=1024 qMinLen=160	dsagen1	-	sha1	31.12.2005
006	ecdsa-Fp	qMinLen=160 r0Min=10^4 MinClass=200	ecgen1	-	sha1	31.12.2005
007	ecdsa-F2m	qMinLen=160 r0Min=10^4 MinClass=200	ecgen2	-	sha1	31.12.2005
008	ecgdsa-Fp	qMinLen=160 r0Min=10^4 MinClass=200	ecgen1	-	sha1	31.12.2005
009	ecgdsa-Fp	qMinLen=160 r0Min=10^4 MinClass=200	ecgen1	-	ripemd160	31.12.2005
010	ecgdsa-F2m	qMinLen=160 r0Min=10^4 MinClass=200	ecgen2	-	sha1	31.12.2005
011	ecgdsa-F2m	qMinLen=160 r0Min=10^4 MinClass=200	ecgen2	-	ripemd160	31.12.2005

4.3 Cryptographic hash functions

A cryptographic hash function is a preimage resistant and 2^{nd} preimage resistant function with a constant-length output used to compute the hashcode of a document to be signed. For secure electronic signatures a hash function must be collision-resistant, which means that it is computationally infeasible to find two different documents yielding the same hashcode (which implies that it is also infeasible to find a different document yielding the same hashcode as a given document).

If due to advances in cryptography or computing technology any of these conditions is no longer met, the hash function is considered insecure and must be removed from the list of approved hash functions (see clause 4.1).

The list of currently approved hash functions is given in Table 2. Each hash function has a unique entry index represented by a string beginning with "2." followed by a two-digit entry number. The relevant OIDs are given in annex B.

Table 2: The list of approved cryptographic hash functions

Hash function entry index	Short hash function entry name	Adoption date	Normative references
2.01	sha1	01.01.2001	[4] and [5]
2.02	ripemd160	01.01.2001	[4]

4.4 Padding methods

Some signature algorithms require a hashcode to be padded to up to a certain block length used with a specific algorithm setting (e.g. RSA modulus length). The present document does not specify mandatory padding methods, but requires a padding method, if required by an algorithm (as indicated in clause 4.5), to meet certain requirements defined in the given normative references.

The list of currently approved padding methods is given in Table 3. Each padding 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 approved padding methods

Padding method entry index	Short padding function entry name	Random number generation method	Random generator parameters	Adoption date	Normative references
3.01	emsa-pkcs1-v1_5	-	-	01.01.2001	[6], clause 9.2.1
3.02	emsa-pss	TBD	TBD	TBD	[17], clause 9.2.2

It is intended that further padding schemes will be added as and when they have been fully standardized. These include several methods based on ISO/IEC 9796-2, currently under review. The emsa-pss method is included as, despite not being finalized, it has been stable for a long time and is a good improvement to the emsa-pkcs1-v1_5 scheme.

4.5 Signature algorithms

4.5.1 General comments

A signature algorithm is applied to the hashcode of the document to be signed to generate a signature with the SCD. The algorithm to be applied for verification with the SVD must be given. Before a signature algorithm is applicable a complete set of approved parameters must be defined. It must be practically impossible to compute the SCD from the SVD.

The list of currently approved signature algorithms is given in Table 4. Each signature algorithm has a unique entry index represented by a string beginning with "1." followed by a two-digit entry number.

Note that the minimum modulus length for RSA is given as 1 020 bits rather than the more natural 1 024 bits. This is to allow for implementations that cannot use the topmost bit(s).

Table 4: The list of approved signature algorithms

Signature algorithm entry index	Short signature algorithm entry name	Signature algorithm parameters	Key and Parameter generation algorithms	Adoption date	Normative references
1.01	rsa	MinModLen=1020	rsagen1	01.01.2001	[7]
1.02	dsa	pMinLen=1024 qMinLen=160	dsagen1	01.01.2001	[9]
1.03	ecdsa-Fp	qMinLen=160 r0Min=10^4 MinClass=200	ecgen1	01.01.2001	[9], [11]
1.04	ecdsa-F2m	qMinLen=160 r0Min=10^4 MinClass=200	ecgen2	01.01.2001	[9], [11]
1.05	ecgdsa-Fp	qMinLen=160 r0Min=10^4 MinClass=200	ecgen1	25.06.2001	[13]
1.06	ecgdsa-F2m	qMinLen=160 r0Min=10^4 MinClass=200	ecgen2	25.06.2001	[13]

The following clauses describe the parameters and key generation algorithms for the signature algorithms listed in Table 4. Table 5 summarizes the approved key generation algorithms for all signature algorithms considered in the present document.

Table 5: The list of approved key generation algorithms

Key generator entry index	Short key generator entry name	Signature algorithm	Random number generation method	Random generator parameters	Adoption date	Normative references
4.01	rsagen1	rsa	trueran or pseuran	EntropyBits ≥ 128 or SeedLen ≥ 128	01.01.2001	
4.02	dsagen1	dsa	trueran or pseuran	EntropyBits ≥ 128 or SeedLen ≥ 128	01.01.2001	[18]
4.03	ecgen1	ecdsa-Fp, ecgdsa-Fp	trueran or pseuran	EntropyBits ≥ 128 or SeedLen ≥ 128	01.01.2001	
4.04	ecgen2	ecdsa-F2m, ecgdsa-F2m	trueran or pseuran	EntropyBits ≥ 128 or SeedLen ≥ 128	01.01.2001	

4.5.2 RSA

4.5.2.1 Parameters

The RSA algorithm's security is based on the difficulty of factoring large integers. To generate SCD and SVD two prime numbers, *p* and *q*, are generated randomly and independently, satisfying the following properties:

- the bit length of the modulus $n = pq$ 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 $0,5 < | \log_2 p - \log_2 q | < 30$;
- there should be sufficiently many primes to choose from and they should be reasonably uniformly distributed.

The SCD consists of the private exponent *d* and the modulus *n*.

The SVD consists of the public exponent *e* and the modulus *n*.

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

4.5.2.2 Key and parameter generation algorithm rsagen1

Generate *p* and *q* as indicated above by applying a random number generation method satisfying the requirements trueran (see clause 4.6.2) or using a method satisfying pseuran (see clause 4.6.3) with an appropriate size seed. Each prime shall effectively be influenced by EntropyBits bits of true randomness or a seed of length SeedLen. 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^{-60}. See annex C for further information on primality testing.

Select the public exponent *e* as an integer between 3 and *n*-1 so that gcd(*e*, lcm(*p*-1, *q*-1))=1 holds. Compute the private exponent *d* such that $ed \equiv 1 \pmod{\text{lcm}(p-1, q-1)}$. Note that any solution *d* to the equation $ed \equiv 1 \pmod{(p-1)(q-1)}$ will automatically satisfy this.

4.5.3 DSA

4.5.3.1 Parameters

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 [9]. 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 [9].

The SCD consists of:

- the public parameters *p*, *q* and *g*;
- a randomly or pseudorandomly generated integer *x*, $0 < x < q$, which is signatory-specific; and
- a randomly or pseudorandomly generated 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 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 SVD 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 [9].

4.5.3.2 Key and parameter generation algorithm dsagen1

p and *q* shall be generated as described in appendix 2.2 of [9].

Generate *x* by applying a random number generation method satisfying the requirements trueran (see clause 4.6.2) or using a method satisfying pseuran (see clause 4.6.3) with an appropriate size seed. Each value of x shall effectively be influenced by EntropyBits bits of true randomness or a seed of length SeedLen. Generate *k* using one of these methods; *k* does not have to be generated using exactly the same method as *x*. The pseudo-random number methods of FIPS 186-2 [9] are no longer recommended for pseuran (due to Bleichenbacher's attack); the method described in [18] should be used instead.

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

4.5.4.1 Parameters

This signature algorithm is referred to as ecdsa-Fp. The algorithm shall be applied as specified in the normative references n in Table 4. The same algorithm is also specified in references [10], [7], [13] 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 large prime;
- q large prime at least qMinLen bits long, $p \neq q$;
- E elliptic curve over a finite field F_p whose order is divisible by q; and
- P fixed point on E of order q.

The class number of the maximal order of the endomorphism ring of E shall be at least MinClass. The value r_0:=min(r: q divides p^r-1) shall be greater than r0Min.

In FIPS 186-2 [9] five curves over a prime field are defined. All these curves fulfil the above requirements.

The SCD consists of:

- the public parameters E, 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 SVD consists of E, q, P and Q, a point of E, which is computed as $Q=xP$.

4.5.4.2 Key and parameter generation algorithm ecgen1 for ecdsa-Fp

The prime number p, which determines the field size, is recommended to be generated as specified in [11]. Alternatively one of the five generalized Mersenne primes given in [9] can be used. The elliptic curve over F_p shall be selected to have an order divisible by a prime q of length ≥ qMinLen ≥ 160 as specified in [11]. Generate x by applying a random number generation method satisfying the requirements trueran or using a method satisfying pseuran with an appropriate size seed. Each value of x shall effectively be influenced by EntropyBits bits of true randomness or a seed of length SeedLen. Generate k using one of these methods; k does not have to be generated using exactly the same method as x.

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

4.5.5.1 Parameters

This signature algorithm is referred to as ecdsa-F2m. The algorithm shall be applied as specified in the normative references [9], [11] given in Table 4. The same algorithm is also specified in references [10], [7], [13] 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 order is divisible by q;
- it must not be possible to define E over F_2, and
- P fixed point on E of order q.

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

In FIPS 186-2 [9] five pseudorandomly generated curves over F_{2m} are defined. All these curves satisfy the above requirements. Note that the Koblitz curves given in [9] 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 [9] and [10] 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 SCD consists of:

- the public parameters E, 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 SVD consists of E, q, m and Q, a point of E which is computed as $Q=xP$.

4.5.5.2 Key and parameter generation algorithm ecgen2 for ecdsa-F2m

The prime number m, which determines the field size, shall be fixed as specified in [11]. The elliptic curve over F_{2m} shall be selected to have an order divisible by a prime q of length ≥ qMinLen ≥ 160 as specified in [11]. Generate x by applying a random number generation method satisfying the requirements trueran or using a method satisfying pseuran with an appropriate size seed. Each value of x shall effectively be influenced by EntropyBits bits of true randomness or a seed of length SeedLen. Generate k using one of these methods; k does not have to be generated using exactly the same method as x.

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

4.5.6.1 Parameters

This signature algorithm is referred to as ecgdsa-Fp. The algorithm shall be applied as specified in [13]. 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 4.5.4.1.

4.5.6.2 Key and parameter generation algorithm ecgen1 for ecgdsa-Fp

The parameter and key generation methods should be the same as the ecdsa-Fp methods described in clause 4.5.4.2.

4.5.7 EC-GDSA based on a group $E(F_2^m)$

4.5.7.1 Parameters

This signature algorithm is referred to as ecgdsa-F2m. The algorithm shall be applied as specified in [13]. 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 4.5.5.1.

4.5.7.2 Key and parameter generation algorithm ecgen2 for ecgdsa-F2m

The parameter and key generation methods should be the same as the ecdsa-F2m methods described in clause 4.5.5.2.

4.6 Random number generation

4.6.1 General comments

Random number generation is relevant to generating the SCD or to generating random parameters in some cryptographic algorithms (e.g. DSA). In some cases it may also be relevant to hash function padding. Therefore the possible choices of random number generation requirements are always given in connection with padding methods (Table 3) and key generation algorithms (Table 5).

Table 6: The list of approved 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	SeedLen	01.01.2001	
5.03	cr_to_X9.30_x	SeedLen	01.01.2001	[18]
5.04	cr_to_X9.30_k	SeedLen	01.01.2001	[18]

4.6.2 Random generator requirements trueran

A physical random generator is based on a physical noise source (primary noise) and a cryptographic or mathematical post-treatment of the primary noise. The primary noise must be subjected to an adapted statistical test on a regular basis (see, e.g. [8], clause 4.11.1, Statistical random number generator tests). See annex D for more detailed information on generating random numbers. The expected effort of guessing a cryptographic key shall be at least equivalent to guessing a random value that is EntropyBits long.

4.6.3 Random generator requirements pseuran

A pseudo-random generator must be initialized by a genuine random number. The initialization value is called seed, of length SeedLen. The output of the generator must satisfy the following requirements:

- no information is ascertainable a priori as to the output bits which are generated;
- the knowledge of a partial output sequence permits no inferences with regard to any of the remaining bits with probability non-negligibly different from random;
- there is no usable method of recovering any previously generated, or future, output, internal state or seed from a subset of the output of the generator.

A pseudo-random generator satisfying these requirements is described in, for example, Blum and Micali (see bibliography). The expected effort to recover any information about the internal state of the generator should be essentially as difficult as guessing a random value that is SeedLen bits long.

If the generator is seeded by at least SeedLen bits once, up to n=100 successive sets of signature creation data can be used in the same way as if they were output from a trueran generator. For mass production (by the CSP) of k keys, $k > n$, it is permissible to slowly feed in true randomness (from a trueran generator) at a rate of at least j=8 bits per output on top of the initial entropy requirement, otherwise the generator should be completely re-seeded.

If re-seeding is employed the security of the re-seeding process shall be as strong as that of the original seeding and follow procedures similar to those for the generation of root keys. The use of re-seeding in smartcards is not permitted.

No backups of the seed or internal states of a pseuran generator are permitted.

4.6.4 Random number generator cr_to_X9.30_x

This random number generator is specified in clause B.2.1 of [18].

4.6.5 Random number generator cr_to_X9.30_k

This random number generator is specified in clause B.2.2 of [18].

Annex A (normative): Updating algorithms and parameters

A.1 Introduction

Cryptographic algorithms in general do not offer unlimited perfect security in the information-theoretical sense. Their security depends on:

- the difficulty of solving a hard mathematical problem that they are based on, and
- the computational infeasibility of solving the problem using the current technology.

This effectively means that a previously secure cryptographic algorithm cannot be considered secure any longer if, for example,

- a mathematical method has been found so that the previously hard problem the algorithm was based on is no longer hard, or
- the advances in technology make it possible to solve the problem within a significantly shorter period of time.

In any of these cases a cryptographic algorithm cannot be considered secure any longer. It is therefore of crucial importance to establish suitable management practices to cope with such developments to prevent the use of insecure algorithms.

This annex defines the management practices to enable fast and technologically appropriate reactions to new developments in computing technology and new findings in the area of cryptography.

A.2 Management Process

As a response to relevant developments in the area of cryptography and technology, it is recommended that the lists of approved algorithms and parameter values be dynamically updated. The initial lists of approved algorithms and their parameter values are given in the present document. This annex identifies several cases where an update of the lists is required:

Adopting a new algorithm. As a result of monitoring the relevant developments in cryptography and computing technology, a notified body of a member state may propose adoption of a new algorithm. An algorithm can be adopted if at least one complete set of parameters for this algorithm has been adopted. Definitions of the complete sets of parameters for a specific algorithm are given in the present document.

Cancelling an algorithm. As a result of monitoring the relevant developments in cryptography and computing technology, a notified body of a member state may propose cancellation of a currently approved algorithm. The proposal should contain a clear reasoning about the insufficient security of the algorithm to be cancelled and about the implications for existing products using this algorithm.

Updating parameter values for an approved algorithm. As a result of monitoring the relevant developments in cryptography and computing technology, a notified body of a member state may propose an update to the parameter values of an approved algorithm. The proposal should contain clear reasoning about the insufficient security of the parameter values to be updated and about the implications for existing products using these parameter values. The dates associated with a suite may be extended or reduced in such a proposal.

If any of the components of a **signature suite** has been cancelled/updated, the suite must be cancelled/updated as well.

Monitoring the relevant developments in cryptography and computing technology should be a continuous activity performed by the notified bodies of the member states. The activity should be focused on answering the following questions:

- Is a currently approved algorithm still considered secure?
- Are the currently approved parameter values for an algorithm still considered secure?
- Are there new algorithms that should be considered? Is there a market demand for such algorithms?

The cancellation/update date should be set in such a way as to allow some **transition period** within which the algorithm/parameter values to be cancelled/updated may be used in the existing products (e.g. signature devices, digital certificates). It should, however, be indicated that the cancelled/updated items should not be considered for use in new products (e.g. signature devices under development, certificates to be issued). The transition period should allow the vendors using the cancelled/updated items time to alter their production process. If the security implications of a cancellation/update are considered very serious, it should be recommended to withdraw the products using the cancelled/updated item before their planned expiry date.

For example, if a signature key length is no longer considered secure under a low or medium attack potential (say 2^{60} operations to recover one key), digital certificates containing signature keys of this length should be revoked as soon as possible. In addition, a transition period should be recommended within which the existing signatures generated with keys of insufficient length can be verified as being valid. After the transition period the corresponding documents should be re-signed with a longer key. When issuing a certificate, a CSP should be careful not to define the validity period of the certificate to be longer than the presumed validity period of the applied cryptographic hash functions, signature algorithms, and their parameter values. It is recommended that the length of the CSP's signature keys (for issuing certificates and CRLs) is always chosen to be longer than the currently recommended key length for the signature algorithm used.

Annex B (informative): Algorithm Object Identifiers

The cryptographic algorithm objects listed in the main body of the present document are specified using their short name and the normative reference where the exact specification of the algorithm is to be found.

In most cases the cryptographic objects have been assigned a full Object Identifier (OID) by their owner organizations to ensure interoperability of different implementations. In [1] the tree of algorithm object identifiers is depicted, and a list is given of cryptographic algorithms that can be identified using OID.

Table B.1 relates the short name of the algorithms, as they appear in the present document, to their respective object identifiers.

Table B.1: Object Identifiers

Short object name	OID	Names used in the present document
rsa	{ joint-iso-ccitt(2) ds(5) module(1) algorithm(8) encryptionAlgorithm(1) 1 }	rsa
rsaEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }	rsa, emsa-pkcs1-v1_5
sha-1WithRSAEncryption	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }	rsa, sha1, emsa-pkcs, etc.
id-dsa	{ iso(1) member-body(2) us(840) x9-57(10040) x9cm(4) 1 }	dsa
id-dsa-with-sha1	{ iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 3 }	dsa, sha1
sha1	{ iso(1) identifiedOrganization(3) oIW(14) oIWSecSig(3) oIWSecAlgorithm(2) 26 }	sha1
ripemd160	{ iso(1) identifiedOrganization(3) teletrust(36) algorithm(3) hashAlgorithm(2) 1 }	ripemd160
id-ecdsa-with-sha1	{ iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) 1 }	ecdsa, sha1
id-rsassa-pss	{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 10 }	emsa-pss

The object identifiers are for information only. The user of the present document is encouraged to consult the owner organization for updates or modifications.

See OID tree structure http://www.darmstadt.gmd.de/secude/Doc/htm/oidgraph.htm for further details.

Annex C (informative): Generation of RSA keys for signatures

C.1 Generation of random prime numbers

C.1.1 Probabilistic primality test

The generation of large prime numbers for cryptographic applications is usually done using probabilistic primality tests. These algorithms are very efficient but may declare that a composite number is prime with some (small) probability. The Miller-Rabin test is one such algorithm. It is known that the probability of declaring a composite number as prime is at most $(1/4)^k$ after k iterations of the elementary algorithm. Consequently, the probability for a random number to be wrongly declared prime after 30 iterations is less than $1/2^{60}$. More precise analysis of the Miller-Rabin algorithm enables a reduction in the number of iterations required; see for example Damgard, Landrock and Pomerance 1993 (see bibliography). For some empirical results on the reliability of simple primality tests see, for example, Rivest 1991 (see bibliography).

The generation of random prime numbers in the range [a,b] can be done using the following algorithm:

- randomly choose an odd integer x in the range [a,b];
- try to divide x by all the prime numbers smaller than a (small) bound B; if x is divisible, go back to the first step;
- using the Miller-Rabin probabilistic primality algorithm, test if x is probably prime; if this is not the case, go back to the first step.

Note that the second step just makes the generation faster since the small prime numbers are the most probable prime factors. Furthermore, this test can be done in a single operation by just testing if x is relatively prime with the precomputed product of all the prime numbers smaller than B.

The probability for an odd integer randomly chosen in the range [a,b] to be prime is about $2/\ln(b)$ so the number of repetitions of the algorithm will be about $\ln(b)/2$. Furthermore, the entropy of k-bit primes generated with this method is about k-$\ln(k)$.

From a practical point of view, the time needed to generate a prime number can be reduced if, instead of choosing a new random integer x for each test, we look for the smallest prime number larger than a random integer x. This can be done by *sieving*, efficiently testing all the numbers in the range x to $x+d$ (for some suitable d) for divisibility by all small primes less than B, before doing any probabilistic primality tests. From a theoretical point of view this makes the distribution of the primes produced less uniform, but in practice the effect is insignificant.

C.1.2 Strong prime numbers

For some cryptographic applications, it is sometimes advised to use so-called *strong* prime numbers. A prime p is said to be strong if:

- p-1 has a large prime factor r;
- p+1 has a large prime factor;
- r-1 has a large prime factor.

Those additional requirements are made to avoid certain factoring algorithms (Pollard p-1 algorithm and its generalizations). If such verifications can be easily done and if the efficiency of prime number generation is not a critical issue, they can be added to the prime number generation procedure. However, it has been proved that the probability for a randomly chosen prime number to fulfil those requirements is overwhelming for the current parameter sizes. Furthermore, the Pollard method is generalized by the elliptic curve factoring method so it is not possible to make an exhaustive list of weak forms of prime numbers.

In conclusion, the prime numbers must be randomly chosen and must not have any kind of special form; if this is done, additional tests can be added.

C.2 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,

$$0{,}5 < |\log_2(p) - \log_2(q)| < 30$$

This 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 $0.5 < |\log_2(p)-\log_2(q)| < 30$ 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 $[2^{k/2-1/4}, 2^{k/2+15}]$;
- Choose a random prime number q in the range $[a,b]$ where $a=\max(2^{k-1}/p, p.2^{-30})$ and $b=\min(2^k/p, p.2^{-1/2})$;
- Let n be the product of p and q.

C.3 Generation of RSA keys

An RSA public key is made of an RSA modulus $n=p.q$ generated as explained in the previous section and of a public exponent e. The only requirement for e is to be relatively prime to $\text{lcm}(p-1,q-1)$. This public exponent may be chosen as small as $e=3$.

The related RSA secret key d (or signature generation key) is computed using the extended Euclidian algorithm:

$$e.d \equiv 1 \bmod \text{lcm}(p-1,q-1)$$

The optimization that consists of first choosing the secret exponent d and then computing the public exponent e can be used but in this case d must be randomly chosen. The value of d should not be too small otherwise there are attacks which can factor the modulus (Wiener 1990, Boneh and Durfee 1999, Durfee and Nguyen 1999, see bibliography). A conservative method would be to choose d randomly in a range at least $\sqrt{n}$ from its minimum and maximum values. In practice choosing d uniformly in the range $[3,n]$ has negligible probability of producing an exploitable value.

Let us also remember that a new modulus n must be used for each user of the signature scheme. A modulus must not be shared by some users, even if different public exponents are used. Furthermore, notice that if the RSA keys are generated as explained above, the probability of generating two keys with the same modulus or the same secret exponent is totally negligible, even if many keys are computed.

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

D.1 Why cryptography needs random numbers

Random data is commonly used in many cryptographic systems. However, different kinds of randomness have to be considered according to the application.

Randomness to avoid replay and collisions. Firstly, in some cases we just need to generate numbers that are distinct. For example, when some challenges are used for authentication, we only require that the same challenge is not generated twice. A solution is to store already used data or to use a counter but a much simpler solution is to generate enough random data. Consequently, as one of the applications of the birthday paradox, we know that, if the challenges are chosen in a large enough set, they will all be different with overwhelming probability. In this case, we see that the statistical properties of "random" data is unimportant; we can tolerate some bias without compromising the security.

Randomness to generate cryptographic data.

Secret keys. A stronger notion of random data must be used for applications such as the generation of secret keys for asymmetric algorithms such as signature. In this case, we need data with strong statistical properties in order to avoid attacks that may use known bias in secret key generation to increase their efficiency. A consequence is that the probability of generating the same secret key twice is negligible.

Public parameters. In addition to secret keys, public parameters, such as the prime numbers in DSA [9], are also generated using a non-deterministic procedure. Furthermore, in this case the randomness of those parameters may by proved to users, just by giving the initial random data and the algorithm used to derive public parameters from the data. Consequently, people are quite sure that the system wide parameters have not been chosen in a very specific way, with some properties that may lower the security.

Temporary secret data. Random bits are also necessary in many protocols for temporary use. For example, the DSA signature scheme (see FIPS 186-2 [9]) uses the secret key of the user who signs the message and also a one-time secret key that has to be regenerated for each signature (the signature of two messages with the same temporary key reveals both the key and the user's signature secret key!). The security analysis of DSA shows that both secrets have the same importance so it does not have any sense to carefully generate users secret keys if temporary keys are not good enough.

Stream ciphers. Finally, in applications such as the generation of secret keys for symmetric algorithms, the random data must be random in a very strong sense; for example, it should not be possible to derive any knowledge of generated data from previously generated data, even if the previously generated data is known. This situation may also occur in the context of signatures, for example if an authority generates secret keys and an attacker tries to gain information on some of those keys after having obtained some others. Consequently, there must not be any usable link between different keys.

Truly random bits vs. pseudorandom bits. The generation of random bits appears to be a difficult task. Consequently, pseudorandom bit generators often have to be used in many applications. They are not generators of truly random bits since they are deterministic but, starting with a random seed, they are able to generate sequences of bits that are random looking. In the next section we describe some ways of generating truly random bits. Then we will focus of pseudorandom bit generation.

D.2 Generation of truly random bits

Let us first understand what we mean by a true random number generator (TRNG); the aim of a TRNG is to generate individual bits, with uniform probability and without any correlation between those bits. Consequently, the knowledge of some bits does not give any information (in a strong information theoretic sense) about the other generated bits.

Notice that for cryptographic applications, public sources of (almost) random bits such as the digits of π cannot be used even if they are random-looking since they are not secret at all!

In practice, perfect random sources are not available, so the idea is to use a source of bits that may not be perfectly random and then to hash the bits in order to obtain really random bits. In this process, we assume that a hash function, such as SHA-1, is able to "extract" the randomness from a biased bit string. Formally, the amount of randomness of such a string is measured by its entropy. The notion of entropy has to be used very carefully since it applies to probability distributions and not to actual bit strings. From a more practical point of view, it is useful to consider that a random source generates sequences of bits and that only a ratio is random. For example, on evaluation we may find that for eight generated bits we have one bit of randomness; consequently, hashing 1 280 generated bits with SHA-1 will hopefully produce 160 truly random bits.

Randomness sources. The available random sources can vary a lot from an application to another. A computer specialized in the generation of secret keys may, for example, use specific devices based on quantum effects to generate high quality random bits (see IEEE P1363 [10], annex D for a list of such sources).

More generally, a normal computer can measure the variations in mouse movements or keystrokes. It can also measure the timing of some tasks that are not achieved in constant time such as hard disk access. The randomness of such sources and an estimation of their entropy may be tested using statistical tests (see clause D.3). In addition the source should be described by a stochastic model. This model should produce similar biases and dependencies to the source taking into account the physical effect that the source is based on. The use of a stochastic model allows the notion of entropy. With the help of the model a sufficient loss rate can be approximated.

Ideally, random data should come from a few different sources and hashed with a high loss rate (for example hash 1 Kbytes to obtain 160 bits of random bits); see [15] for further details.

Be aware that the functions that generate random numbers that are implemented in programming libraries do not always output cryptographically secure bits! It is, for example, well known that in the old versions of the C *rand* function, the least significant bits were not very random at all. Furthermore, remember that good statistical properties of a sequence, for example the digits of π or linear congruential generators, are not sufficient to make them useful and secure for cryptographic applications.

Random number generation is very difficult in simple environments such as smart cards. Random sources are not directly available since there is no physical device and everything is very deterministic. Usually simple mechanisms such as pairs of asynchronous clocks are implemented but the quality of such randomness generators has to be analysed very carefully using mathematical modelling and statistical tests.

Algorithmic countermeasures to improve security. Some general algorithmic solutions may be used to increase the security if a good source of randomness is not available. For example, consider the DSA signature scheme; the signature algorithm involves a secret key x related to the public verification key $g^x \bmod p$ and a temporary secret key k that has to be refreshed for each signature. FIPS 186-2 [9] says that k may be randomly or pseudo-randomly generated. This means that the values of k do not have to be perfectly independent (note that if the discrete logarithm problem is hard, k is never revealed).

The secret key x is generated once, usually outside of the smart card, so we can assume it has good randomness properties. Consequently, when a bit string is generated from the available source, the following transformations may be used to increase the security:

- encrypt the bit strings with a stream cipher in order to hide possible repetitions while keeping the same number of available bits;
- combine the obtained bits with the secret key x (using a hash function or a block cipher for example);
- other data, such as a counter and/or a smart card unique serial number, can be added to increase security.

D.3 Statistical tests

Some statistical tests can be used to verify the quality of a random source. **It should be clear that passing those tests is not a sufficient guarantee of randomness!** Consider for example the π digits or the bit string obtained by the concatenation of SHA-1(0), SHA-1(1),… They are bad cryptographic bit strings but they pass all the tests we may imagine.

FIPS 140-1 [8] describes four very simple tests. They can be easily implemented and consequently can be used as a way to check obvious failures of a TRNG.

Let us consider a string of 20 000 bits output for a generator. If all the four following tests are passed, the generator passes the test (we insist on the fact that this only means that the generator is not out of order but it does not imply anything on the randomness of generated bits).

- **Test 1:** the number n of "1"s should satisfy $9654<n<10346$;
- **Test 2:** divide the 20 000 bits into 5 000 contiguous 4 bit segments. Let $f(i)$ be the number of occurrences of i interpreted has a segment of 4 bits ($0<=i<=15$). Compute $X = 16 / 5\,000(f(0)^2 + f(1)^2 + \ldots f(15)^2) - 5\,000$. The test is passed if $1{,}03 < X < 57{,}4$;
- **Test 3:** a run is a maximal sequence of consecutive bits of either all "1"s or all "0"s. The number of runs of length 1 of 0"s or 1"s must be between 2 267 and 2 733 ; for length 2 between 1 079 and 1 421, for length 3 between 502 and 748, for length 4 between 223 and 402, for length 5 and for length larger or equal to 6 between 90 and 223;
- **Test 4:** there is no run of length 34 or more.

These tests are described and analysed more precisely in chapter 5 of "Handbook of Applied Cryptography" (see bibliography). They are tuned in such a way that a truly random bit string fails any of the tests with probability less than 10^{-6}.

Some publicly available programs implement much more sophisticated tests such as Maurer's universal test (see bibliography) based on compression techniques. One may for example refer to the NIST test suite (see bibliography).

D.4 Pseudorandom bit generation

D.4.1 General

In applications that require many random bits or when random sources are not available, random-looking bits are generated from an initial random seed using a deterministic algorithm. It should be clear that this does not "create" additional randomness but only derives random looking bits from the randomness initially present in the seed.

The differences between the security levels of PRNGs are determined by the assumptions that are made on the availability of generated bits. Weak PRNGs may generate random-looking bits but the knowledge of a sequence of bits enables the computation of following ones. In strong PRNGs, the knowledge of a sequence of bits does not give information on bits that are subsequently (and even previously) generated. Furthermore, some PRNGs are provably secure based on number theoretical assumptions. A formal classification of PRNGs and their suitability for different cryptographic applications is defined in AIS 20 (see bibliography). Here is a short list of the most well known PRNGs.

D.4.2 ANSI X9.17 generator

NOTE: See ANSI X9.17 [16].

This PRNG is designed to pseudorandomly 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.

D.4.3 FIPS 186 generator

NOTE: See FIPS 186-2 [9].

The digital signature standard proposes two kinds of PRNGs to generate public parameters, secret keys and temporary secret keys. The first one is based on a hash function such as SHA-1 and the second one uses a block cipher such as DES. The PRNGs are no longer considered suitable for secret key generation.

D.4.4 RSA PRNG and Blum-Blum-Shub PRNG

NOTE: See Menezes et al. 1997 in bibliography.

Those PRNGs 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 PRNGs described above, the security of which is only heuristic.

A security analysis of PRNGs appeared in the paper of Kelsey et al. at FSE '98 (see bibliography). Let us remind ourselves of some simple ways they propose to increase the security of existing PRNGs:

- the output may be hashed, using SHA-1 for example, to make direct cryptanalysis of the PRNG much more difficult;
- the initial seed may be hashed with a counter and/or the date to avoid replay attacks;
- the same seed may not be used to generate too many output bits.

D.5 Conclusion

In conclusion, cryptographically secure generation of random bits usually requires the use of either a good true random noise source or a good pseudorandom bit generator initially fed with a good truly generated seed. Both of those primitives (the TRNG and the PRNG) have to be carefully chosen according to the application and the environment in which it is implemented. For TRNGs and PRNGs the requirements listed in clauses 4.6.2 and 4.6.3 need to be observed.

Annex E (informative): Verification

The present document specifies dates beyond which signature suites should not be used for producing signatures. It is difficult to put a single date to the time beyond which verification of a signature loses its status because other measures outside the scope of the present document can be applied to the signature, such as time-stamping [19] or re-signing. If no measures such as these are used then it is recommended that - from a technical perspective with no legal impact - verification is done at most one year after the date at which the algorithm suite expires for signing purposes.

Annex F (informative): Bibliography

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.

AIS 20: "Application Notes and Interpretation of the Scheme: Functionality classes and evaluation methodology for deterministic random number generators". Available at http://www.bsi.bund.de/aufgaben/ii/zert/jil_ais/ais20e.pdf.

"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" at http://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.

"A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications" NIST Special Publication 800-22, 2000, http://csrc.nist.gov/rng.

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.

History

Document history		
V1.1.1	March 2003	Publication