INTERNATIONAL STANDARD

ISO/IEC 14888-3

Second edition
2006-11-15

AMENDMENT 1
2010-06-15

Information technology — Security techniques — Digital signatures with appendix —

Part 3: Discrete logarithm based mechanisms

AMENDMENT 1: Elliptic Curve Russian Digital Signature Algorithm, Schnorr Digital Signature Algorithm, Elliptic Curve Schnorr Digital Signature Algorithm, and Elliptic Curve Full Schnorr Digital Signature Algorithm

Technologies de l'information — Techniques de sécurité — Signatures numériques avec appendice —

Partie 3: Mécanismes basés sur un logarithme discret

AMENDEMENT 1: Algorithme de signature numérique russe de courbe elliptique, algorithme de signature numérique schnorr, algorithme de signature numérique schnorr de courbe elliptique, et algorithme de signature numérique schnorr totale de courbe elliptique

Reference number
ISO/IEC 14888-3:2006/Amd.1:2010(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC 14888-3:2006 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

Amendment 1 to ISO/IEC 14888-3:2006 introduces four digital signature algorithms: Elliptic Curve Russian Digital Signature Algorithm (EC-RDSA), Schnorr Digital Signature Algorithm (SDSA), Elliptic Curve Schnorr Digital Signature Algorithm (EC-SDSA), and Elliptic Curve Full Schnorr Digital Signature Algorithm (EC-FSDSA). Object identifiers, test vectors, a comparison of certificate-based mechanisms, and claimed features for choosing a mechanism are given.

Information technology — Security techniques — Digital signatures with appendix —

Part 3: Discrete logarithm based mechanisms

AMENDMENT 1: Elliptic Curve Russian Digital Signature Algorithm, Schnorr Digital Signature Algorithm, Elliptic Curve Schnorr Digital Signature Algorithm, and Elliptic Curve Full Schnorr Digital Signature Algorithm

Page 1, Clause 2

In the second normative reference, replace “1998” with “2008”.

Page 3, Clause 4

Add the following to the end of the list of symbols:

Π_Y	y-value of Π
$\mathbf{F}$	a finite field
F_P	a finite field of prime order p
Z_P^*	a multiplicative group over F_P

Page 27, Clause 6

Add the following subclauses after 6.6.4.6:

6.7 EC-RDSA

6.7.1 Introduction to EC-RDSA

EC-RDSA (Elliptic Curve Russian Digital Signature Algorithm) is a signature mechanism with verification key $Y = [X]G$; that is, the parameter D is equal to 1. The message is prepared such that M_1 is empty and M_2 is the message to be signed, i.e., $M_2 = M$. The coefficients (A, B, C) of the EC-RDSA signature equation are set as follows:

$$(A, B, C) = (T_1, T_2, -S),$$

where $(T_1, T_2) = (H, R)$ and $H = h(M)$ is the hash-code of message M, converted to an integer as described in 6.7.4.5.

The witness function is defined by the formula

$$R = FE2I(\Pi_X) \bmod q.$$

Thus the signature equation becomes

$$HK + RX - S \equiv 0 \pmod{q}.$$

NOTE EC-RDSA stands for Elliptic Curve Russian Digital Signature Algorithm. The mechanism is taken from a Russian State Standard [36]. The notation here has been changed from GOST R 34.10-2001 to conform with the notation used in ISO/IEC 14888.

6.7.2 Parameters

p	a prime
$\boldsymbol{E}$	an elliptic curve group over the field $GF(p)$
$\#\boldsymbol{E}$	the cardinality of $\boldsymbol{E}$
q	a prime divisor of $\#\boldsymbol{E}$
G	a point on the elliptic curve of order q

Hash-function identifier or OID with specified hash-function.

All these parameters can be public and can be common to a group of users.

6.7.3 Generation of signature key and verification key

The signature key of a signing entity is a secretly generated random or pseudo-random integer X such that $0 < X < q$. The parameter D is 1. The corresponding public verification key Y is

$$Y = [X]G.$$

A user's secret signature key X and public verification key Y are normally fixed for a period of time. The signature key X shall be kept secret.

NOTE The Russian standard for digital signature (GOST R 34.10-2001) does not completely specify the process of generation of a user's secret signature key X.

6.7.4 Signature process

6.7.4.1 Producing the randomizer

The signing entity generates a random or pseudo-random integer K such that $0 < K < q$.

6.7.4.2 Producing the pre-signature

The input to this stage is the randomizer K, and the signing entity computes

$$\Pi = [K]G.$$

6.7.4.3 Preparing message for signing

The message is prepared such that M_1 is empty and M_2 is the message to be signed, i.e., $M_2 = M$.

6.7.4.4 Computing the witness

The signing entity computes $R = FE2I(\Pi_X) \bmod q$.

6.7.4.5 Computing the assignment

The signing entity computes $H = h(M_2)$. H is then converted to an integer according to conversion rule *BS2I* in Annex B. If H is equal to $0 \bmod q$, then H is set to 1. The assignment (T_1, T_2) is ($BS2I(H)$, R), if $BS2I(H) \neq 0 \pmod q$, or (1, R) otherwise.

6.7.4.6 Computing the second part of the signature

The signature is (R, S) where R is computed as given in 6.7.4.4, and

$$S = RX + KH \bmod q.$$

The signer should check whether $R = 0$ or $S = 0$. If either $R = 0$ or $S = 0$, a new value of K should be generated and the signature should be recalculated.

6.7.4.7 Constructing the appendix

The appendix will be the concatenation of (R, S) and an optional text field *text*, i.e. it will equal $((R,S)\,||\,text)$.

6.7.4.8 Constructing the signed message

A signed message is the concatenation of the message M and the appendix

$$M\,||\,((R,S)\,||\,text).$$

6.7.5 Verification process

6.7.5.1 Retrieving the witness

The verifier retrieves the witness R and the second part of the signature S from the appendix. The verifier then checks whether $0 < R < q$ and $0 < S < q$; if either condition does not hold, the signature shall be rejected.

6.7.5.2 Preparing message for verification

The verifier retrieves M from the signed message and divides the message into two parts M_1 and M_2. M_1 will be empty and $M_2 = M$.

6.7.5.3 Retrieving the assignment

This stage is identical to 6.7.4.5. The inputs to the assignment function consist of the witness R from 6.7.5.1 and M_2 from 6.7.5.2. The assignment $T = (T_1, T_2)$ is recomputed as the output from the assignment function given in 6.7.4.5.

6.7.5.4 Recomputing the pre-signature

The inputs to this stage are system parameters, the verification key *Y*, the assignment $T = (T_1, T_2)$ from 6.7.5.3, and the second part of the signature *S* from 6.7.5.1. The verifier obtains a recomputed value $\bar{\Pi}$ of the pre-signature by computing it using the formula

$$\bar{\Pi} = [-T_1^{-1}T_2 \bmod q]Y + [T_1^{-1}S \bmod q]G.$$

6.7.5.5 Recomputing the witness

The computations at this stage are the same as in 6.7.4.4. The verifier executes the witness function. The input is $\bar{\Pi}$ from 6.7.5.4. The output is the recomputed witness $\bar{R}$.

6.7.5.6 Verifying the witness

The verifier compares the recomputed witness, $\bar{R}$ from 6.7.5.5 to the retrieved version of *R* from 6.7.5.1. If $\bar{R} = R$, then the signature is verified.

6.8 SDSA

6.8.1 Introduction to SDSA

SDSA (Schnorr Digital Signature Algorithm) is a signature mechanism with $\boldsymbol{E} = Z_P^*$, *p* a prime, and *q* a prime dividing *p*-1. The parameter *D* is equal to 1. The message is prepared such that M_1 is the message to be signed, i.e., $M_1 = M$, and M_2 is empty. The witness function is defined by setting *R* equal to a hash-code. The assignment function is defined by setting $T_1 = -1$ and T_2 equal to the negative of the integer which is obtained by converting *R* to an integer according to the conversion rule, *BS2I*, given in Annex B and then reducing modulo *q*.

The coefficients (*A*, *B*, *C*) of the SDSA signature equation are set as follows

$$(A, B, C) = (T_1, T_2, S).$$

Thus the signature equation becomes

$$-K + T_2X + S \equiv 0 \pmod q).$$

6.8.2 Parameters

p	a prime, where $2^{\alpha-1} < p < 2^{\alpha}$.
q	a prime divisor of *p* -1, where $2^{\beta-1} < q < 2^{\beta}$.
G	a generator of the subgroup of order *q*, such that 1 < *G* < *q*.

Four choices for the pair (*α, h*) are allowed in SDSA, namely (1024, SHA-1), (2048, SHA-224), (2048, SHA-256), and (3072, SHA-256). Corresponding *β* should be selected according to *α* in 5.1.3.1, Table 1.

The integers *p*, *q*, and *G* can be public and can be common to a group of users.

The parameters *p*, *q* and *G* are generated as specified in Annex D. The parameters *p* and *q* can be generated using the prime generation techniques given in ISO/IEC 18032.

NOTE 1 It is recommended that all users check the proper generation of the SDSA public parameters according to FIPS PUB 186-3.

NOTE 2 SHA-1 has recently been demonstrated to provide less than 80 bits of security for digital signatures. The use of SHA-1 is not recommended for the generation of digital signatures.

6.8.3 Generation of signature key and verification key

The signature key of a signing entity is a secretly generated random or pseudo-random integer X such that $0 < X < q$. The parameter D is 1. The corresponding public verification key Y is

$$Y = G^X \bmod p.$$

A user's secret signature key X and public verification key Y are normally fixed for a period of time. The signature key X shall be kept secret.

6.8.4 Signature process

6.8.4.1 Producing the randomizer

The signing entity computes a random or pseudo-random integer K such that $0 < K < q$.

6.8.4.2 Producing the pre-signature

The input to this stage is the randomizer K, and the signing entity computes

$$\Pi = G^K \bmod p.$$

6.8.4.3 Preparing message for signing

The message is prepared such that M_1 is the message to be signed, i.e., $M_1 = M$, and M_2 is empty.

6.8.4.4 Computing the witness

The signing entity computes the witness R as the hash-code of the pre-signature Π and the first part of the message M_1

$$R = h(\Pi \| M).$$

6.8.4.5 Computing the assignment

The witness R is converted to an integer according to conversion rule, *BS2I,* in Annex B and then reducing modulo q. The assignment (T_1, T_2) is (-1, $-BS2I(R) \bmod q$).

6.8.4.6 Computing the second part of the signature

The signature is (R, S) where $S = (K + BS2I(R)X) \bmod q$.

As an option, one may wish to check if $R = 0$ or $S = 0$. If either $R = 0$ or $S = 0$, a new value of K should be generated and the signature should be recalculated. (It is extremely unlikely that $R = 0$ or $S = 0$ if signatures are generated properly).

6.8.4.7 Constructing the appendix

The appendix will be the concatenation of (R, S) and an optional text field *text*.

6.8.4.8 Constructing the signed message

A signed message is the concatenation of the message, M, and the appendix

$$M \,||\, ((R, S) \,||\, text).$$

6.8.5 Verification process

The verifying entity acquires the necessary data items required for the verification process.

6.8.5.1 Retrieving the witness

The verifier retrieves the witness R and the second part of the signature S from the appendix. The verifier checks to see that R is a non-zero string within the range of the hash function and that $0 < S < q$.

6.8.5.2 Preparing message for verification

The verifier retrieves M from the signed message and divides the message into two parts M_1 and M_2, such that $M_1 = M$ and M_2 is empty.

6.8.5.3 Retrieving the assignment

The input to the assignment function consists of the witness R from 6.8.5.1. The assignment

$$T = (T_1, T_2) = (-1, -BS2I(R) \bmod q).$$

6.8.5.4 Recomputing the pre-signature

The inputs to this stage are domain parameters, verification key Y, assignment $T = (T_1, T_2)$ from 6.8.5.3 and second part of the signature S from 6.8.5.1. The verifier obtains a recomputed value Π' of the pre-signature by computing it using the formula

$$\Pi' = Y^{(T_2 \bmod q)} G^{(-ST_1 \bmod q)} \bmod p.$$

6.8.5.5 Recomputing the witness

The computations at this stage are the same as in 6.8.4.4 and 6.8.4.5. The verifier executes the witness function. The input is Π' from 6.8.5.4 and M_1 from 6.8.5.2. The output is the recomputed witness R' which is the hash-code of the recomputed pre-signature Π' and the message M.

6.8.5.6 Verifying the witness

The verifier compares the recomputed witness, R' from 6.8.5.5 to the retrieved version of R from 6.8.5.1. If $R' = R$, then the signature is verified.

6.9 EC-SDSA

6.9.1 Introduction to EC-SDSA

EC-SDSA (Elliptic Curve Schnorr Digital Signature Algorithm) is a signature mechanism with verification key $Y = [X]G$; that is, the parameter D is equal to 1. The message is prepared such that M_2 is empty and $M_1 = M$ the message to be signed. The witness R is computed as hash-code

$$R = h(FE2BS(\Pi_X) \,||\, FE2BS(\Pi_Y) \,||\, M).$$

NOTE This specification of EC-SDSA generates the witness by hashing the concatenation of the x-coordinate, the y-coordinate and the message. However, the mechanism would remain secure even if the y-coordinate was omitted from the hash computation. As a result, future versions of this standard may permit the omission of the y-coordinate from the hash calculation to improve performance.

The coefficients (A, B, C) of the EC-SDSA signature equation are set as follows:

$$(A, B, C) = (T_1, T_2, S),$$

where $(T_1, T_2) = (-1, -BS2I(R) \bmod q)$.

Thus the signature equation becomes:

$$-K + T_2X + S \equiv 0 \pmod{q}.$$

6.9.2 Parameters

$\boldsymbol{F}$	a finite field
$\boldsymbol{E}$	elliptic curve group over the field $\boldsymbol{F}$
$\#\boldsymbol{E}$	cardinality of $\boldsymbol{E}$
q	prime divisor of $\#\boldsymbol{E}$
G	a point of order q on the elliptic curve

Hash-function identifier or OID with specified hash-function.

All these parameters can be public and can be common to a group of users.

NOTE It is recommended that all users check the proper generation of the public parameters according to X9.62 [5].

6.9.3 Generation of signature key and verification key

The signature key of a signing entity is a secretly generated random or pseudo-random integer X such that $0 < X < q$. The parameter D is 1. The corresponding public verification key Y is $Y = [X]G$.

A user's secret signature key X and public verification key Y are normally fixed for a period of time. The signature key X shall be kept secret.

6.9.4 Signature process

6.9.4.1 Producing the randomizer

The signing entity computes a random or pseudo-random integer K such that $0 < K < q$.

6.9.4.2 Producing the pre-signature

The input to this stage is the randomizer K, and the signing entity computes $\Pi = [K]G$.

6.9.4.3 Preparing message for signing

The message is prepared such that M_1 is the message to be signed, i.e., $M_1 = M$, and M_2 is empty.

6.9.4.4 Computing the witness

The signing entity computes the witness $R = h(FE2BS(\Pi_X)||FE2BS(\Pi_Y)||M)$.

6.9.4.5 Computing the assignment

The witness R is converted to an integer according to conversion rule, *BS2I,* in Annex B and then reducing modulo q.
The assignment (T_1, T_2) is (-1, $-BS2I(R)$ mod q).

6.9.4.6 Computing the second part of the signature

The signature is (R, S) where $S = (K + BS2I(R)X)$ mod q.

As an option, one may wish to check if $R = 0$ or $S = 0$. If either $R = 0$ or $S = 0$, a new value of K should be generated and the signature should be recalculated. (It is extremely unlikely that $R = 0$ or $S = 0$ if signatures are generated properly).

6.9.4.7 Constructing the appendix

The appendix will be the concatenation of (R, S) and an optional text field, *text*.

6.9.4.8 Constructing the signed message

A signed message is the concatenation of the message, M, and the appendix:

$$M \,||\, ((R, S) \,||\, text).$$

6.9.5 Verification process

The verifying entity acquires the necessary data items required for the verification process.

6.9.5.1 Retrieving the witness

The verifier retrieves the witness R and the second part of the signature S from the appendix. The verifier first checks to see that R is a non-zero string within the range of the hash function and $0 < S < q$; if either condition is violated the signature shall be rejected.