INTERNATIONAL STANDARD

ISO/IEC 15946-2

First edition
2002-12-01

Information technology — Security techniques — Cryptographic techniques based on elliptic curves —

Part 2: Digital signatures

Technologies de l'information — Techniques de sécurité — Techniques cryptographiques basées sur les courbes elliptiques —

Partie 2: Signatures digitales

Reference number
ISO/IEC 15946-2:2002(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. 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.

International Standard ISO/IEC 15946-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 15946 consists of the following parts, under the general title *Information technology — Security techniques — Cryptographic techniques based on elliptic curves*:

— *Part 1: General*

— *Part 2: Digital signatures*

— *Part 3: Key establishment*

— *Part 4: Digital signatures giving message recovery*

Annexes A and B of this part of ISO/IEC 15946 are for information only.

Introduction

Some of the most interesting and potentially useful of the public-key cryptosystems that are currently available are cryptosystems based on elliptic curves defined over finite fields. The concept of an elliptic curve based public-key cryptosystem is rather simple:

— Every elliptic curve is endowed with a binary operation "+" under which it forms a finite abelian group.

— The group law on elliptic curves extends in a natural way to a "discrete exponentiation" on the point group of the elliptic curve.

— Based on the discrete exponentiation on an elliptic curve one can easily derive elliptic curve analogues of the well known public-key schemes of Diffie-Hellman and ElGamal type.

The security of such a public-key system depends on the difficulty of determining discrete logarithms in the group of points of an elliptic curve. This problem is - with current knowledge - much harder than the factorization of integers or the computation of discrete logarithms in a finite field. Indeed, since Miller and Koblitz in 1985 independently suggested the use of elliptic curves for public-key cryptographic systems, no substantial progress in tackling the elliptic curve discrete logarithm problem has been reported. The only known general algorithms to determine elliptic curve discrete logarithms take fully exponential time. Thus, it is possible for elliptic curve based public-key systems to use much shorter parameters than the RSA system or the classical discrete logarithm based systems that make use of the multiplicative group of some finite field. This yields significantly shorter digital signatures and system parameters and allows for computations using smaller integers.

It is the purpose of this document to meet the increasing interest in elliptic curve based public key technology and describe the components that are necessary to implement a secure digital signature system based on elliptic curves.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this International Standard may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from:

ISO/IEC JTC 1/SC 27 Standing Document 8 (SD 8)

SD 8 is publicly available at:
http://www.din.de/ni/sc27

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 15946 may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 2: Digital signatures

1 Scope

This part of ISO/IEC 15946 specifies public-key cryptographic techniques based on elliptic curves. They include the establishment of keys for secret-key systems, and digital signature mechanisms.

This part of ISO/IEC 15946 describes mechanisms for digital signatures. The mathematical background and general techniques necessary for implementing the mechanisms are described in part 1 of ISO/IEC 15946.

The scope of this part of ISO/IEC 15946 is restricted to cryptographic techniques based on elliptic curves defined over finite fields of prime power order (including the special cases of prime order and characteristic two). The representation of elements of the underlying finite field (i.e. which basis is used) is outside the scope of this part of ISO/IEC 15946.

This part of ISO/IEC 15946 does not fully specify the implementation of the techniques it defines. Thus, additional specification may be required to ensure the compatibility of products complying with this part of ISO/IEC 15946.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 15946. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 15946 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 10118 (all parts), *Information technology – Security techniques – Hash-functions*

ISO/IEC 15946-1, *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General*

3 Symbols and abbreviated terms

For the purposes of this part of ISO/IEC 15946, the symbols, terms and definitions described in ISO/IEC 15946-1 apply. In addition, the following terms and symbols are used.

3.1 Terms and definitions

3.1.1 domain parameter

[ISO/IEC14888-1] A data item which is common to and known by or accessible to all entities within the domain.

NOTE The set of domain parameters may contain data items such as hash-function identifier, elliptic curve parameters, or other parameters specifying the security policy in the domain.

3.1.2 hash-code
[ISO/IEC 10118-1] The string of bits which is the output of a hash-function.

3.1.3 hash-function
[ISO/IEC 10118-1] A function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

— for a given output, it is computationally infeasible to find an input which maps to this output; and

— for a given input, it is computationally infeasible to find a second input which maps to the same output.

NOTE Computational feasibility depends on the specific security requirements and environment.

3.1.4 message
[ISO/IEC 9796-1] A string of bits of any length.

3.1.5 randomizer
[ISO/IEC 14888-1] A secret data item produced by the signing entity in the pre-signature production process, and not predictable by other entities.

3.1.6 signature
[ISO/IEC 9796-1] The string of bits resulting from the signature process.

3.1.7 signature key
[ISO/IEC 14888-1] A secret data item specific to an entity and usable only by this entity in the signature process.

3.1.8 signature process
[ISO/IEC 14888-1] A process which takes as inputs the message, the signature key and the domain parameters, and which gives as output the signature.

3.1.9 verification key
[ISO/IEC 14888-1] A data item which is mathematically related to an entity's signature key and which is used by the verifier in the verification process.

3.1.10 verification process
[ISO/IEC 14888-1] A process which takes as input the signed message, the verification key and the domain parameters, and which gives as output the result of the signature verification: valid or invalid.

3.2 Symbols and notation

In addition to the symbols and notation defined in ISO/IEC 15946-1, the following symbols and notation are used in this part of ISO/IEC 15946:

Cert_Data	certification data
e, e'	hash code and recovered hash code respectively
k	randomizer
m	positive integer
(*r,s*) , (*r',s'*)	signature and received signature respectively
M, *M'*	message and received message respectively
len_x	length in bits of *x*
h()	hash function

4 General Model for Digital Signatures with Appendix

This part of ISO/IEC 15946 describes signature schemes based on the one way property of discrete exponentiation on elliptic curves defined over some finite prime field $\boldsymbol{F}(p)$, some finite field $\boldsymbol{F}(2^m)$ or some finite extension field of $\boldsymbol{F}(p)$.

A digital signature scheme is defined by the specification of the following processes:

— Parameter generation process;

— Signature generation process;

— Signature verification process.

4.1 Parameter Generation Process

The parameters can be divided into domain parameters and user parameters.

4.1.1 Domain Parameters

The domain parameters consist of parameters to define a finite field, parameters to define an elliptic curve over the finite field, and other public information which is common to and known by or accessible to all entities within the domain. As well as the domain parameters for a general cryptographic scheme based on elliptic curves which are specified in ISO/IEC 15946-1, the following parameters are required to be specified:

— An identifier for the digital signature scheme used;

— An identifier for the hash function $\boldsymbol{h}()$ mapping an arbitrary message to a bit string of constant length;

— The user parameter generation procedure.

NOTE One of the domain parameters specified in ISO/IEC 15946-1 is the function $\pi()$ for converting a field element into an integer. It should be noted that operation of this function is trivial when the field is $\boldsymbol{F}(p)$ or $\boldsymbol{F}(2^m)$, but is not trivial when the field is $\boldsymbol{F}(p^m)$.

4.1.2 User Parameters

Each entity has its own public and private parameters. The user parameters of the entity A consist of the following:

— The private key d_A.

— The public key P_A.

— (Optional) Other information, which is specific to the entity A, for the use in the signature generation and/or verification process.

4.1.3 Validity of Parameters

The signature verifier may require assurance that the domain parameters and public key are valid, otherwise there is no assurance of meeting the intended security even if the signature verifies. The signer may also require assurance that the domain parameters and public key are valid, otherwise an adversary may be able to generate signatures that verify.

Assurance of validity of domain parameters can be provided by one of the following:

— Selection of valid domain parameters from a published source, such as a standard.

— Generation of valid domain parameters by a trusted third party, such as a Certification Authority.

— Validation of candidate domain parameters by a trusted third party, such as a Certification Authority.

— For the signer, generation of valid domain parameters by the signer using a trusted system.

— Validation of candidate domain parameters by the user (i.e., the signer or verifier).

Assurance of validity of a public key can be provided by one of the following:

— For the signer, generation of the public/private key pair using a trusted system.

— For the signer or verifier, validation of the public key by a trusted third party, such as a Certification Authority.

— Validation of the public key by the user (i.e., the signer or verifier).

4.2 Signature Generation Process

The following data items are required for the signature generation process:

— the domain parameters;

— the signer A's user parameters including the private key d_A;

— the message M.

For all the schemes the signature generation process consists of the following procedures:

— calculation of the message digest;

— elliptic curve computations;

— computations modulo the group order of the base point G.

The output of the signature generation process is a pair of integers (r,s) that constitutes A's digital signature of the message M.

4.2.1 Randomizer

Prior to each signature computation the signing entity must have a fresh, secret value of randomizer available. The randomizer is an integer k such that $0 < k < n$. The implementation of the signature scheme must ensure that the following two requirements are satisfied:

— The used randomizers are never disclosed, since knowledge of a randomizer and the signature generated using this randomizer can be used to compromise the private signature key.

— Randomizers are statistically unique, that is, the probability that the same randomizer is used to produce signatures for two different messages is negligible. If the same value of randomizer is used to produce signatures for two different messages, then the signature key can be deduced from the signatures.

4.3 Signature Verification Process

The following data items are required for the signature verification process:

— the domain parameters;

— the signer A's user parameters including the public key P_A but not the private key d_A;

— the received message, M';

— the received signature of M, represented as the two integers, r' and s'.

For all the schemes the signature verification process consists of some or all of the following procedures:

— signature size verification;

— calculation of the message digest;

— computations modulo the group order of the base point G;

— elliptic curve computations;

— signature checking.

If all procedures are passed, the signature is accepted by the verifier, otherwise it is rejected.

5 EC-GDSA Signature Algorithm

The EC-GDSA signature scheme is an example of a mechanism producing a digital signature with appendix.

5.1 Domain and User Parameters

The bit length of n should be greater than the output bit length of the hash function $\boldsymbol{h}()$.

The private and public keys of entity A, d_A and P_A respectively, should be produced in accordance with the procedure 7.2 defined in ISO/IEC 15946-1.

5.2 Signature Generation Process

The input to the signature process consists of:

— the domain parameters;

— the signer's private key d_A;

— the message M.

The output of the signature generation process is a pair $(r,s) \in \boldsymbol{F}(n)^* \times \boldsymbol{F}(n)^*$ that constitutes A's digital signature of the message M.

To sign a message M, A executes the following steps:

5.2.1 Calculation of the message digest

1. Compute the hash-code $e = \boldsymbol{h}(M)$.

5.2.2 Elliptic Curve Computations (Arithmetic operations in the underlying field)

2. Select a random integer k in the interval $\{1, \ldots, n-1\}$.

3. Compute the elliptic curve point $(x_1, y_1) = kG$.

5.2.3 Computations modulo the group order of G (Arithmetic operations in $\boldsymbol{F}(n)$)

4. Set $r = \pi(kG) \bmod n$.

5. Set $s = (kr - e)d_A \bmod n$.

If the signature generation process yields either $s = 0$ or $r = 0$ then the process must be repeated from step 2 with a new random value k. (But note that the probability that either $r = 0$ or $s = 0$ is negligibly small if k is chosen as described in 5.2.2.)

NOTE Since the computation of r is independent of any message to be signed, r may be precomputed and stored for a later one-time use in a signing operation.

5.3 The Signature

The pair $(r,s) \in \boldsymbol{F}(n)^* \times \boldsymbol{F}(n)^*$ constitutes A's digital signature of the message *M*.

5.4 Signature Verification Process

The signature verification process consists of four steps: signature size verification; calculation of the message digest; elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— the domain parameters;

— A's public key P_A;

— the received message, *M'*;

— the received signature of *M*, represented as the two integers, *r'* and *s'*.

To verify A's signature of message *M'*, *B* executes the following steps:

5.4.1 Signature Size Verification

1. Verify that $0 < r' < n$ and $0 < s' < n$; if not, then reject the signature.

5.4.2 Calculation of the message digest

2. Compute the hash-code $e' = \boldsymbol{h}(M')$ using the hash function $\boldsymbol{h}()$.

5.4.3 Elliptic Curve Computations

3. Compute $w = (r')^{-1} \bmod n$.

4. Compute $u_1 = e'w \bmod n$ and $u_2 = s'w \bmod n$.

5. Compute the elliptic curve point $(x_1, y_1) = u_1G + u_2P_A$.

5.4.4 Signature Checking

6. Compute $v = \pi((x_1, y_1)) \bmod n$.

If $r' = v$, then the signature shall be accepted by the verifier. If $r' \neq v$, then the signature shall be rejected by the verifier.

6 EC-DSA

The signature scheme EC-DSA is the elliptic curve analogue of the DSA signature scheme. It is an example of a mechanism producing a digital signature with appendix.

6.1 Domain and User Parameters

The bit length of *n* should be greater than the output bit length of the hash function $\boldsymbol{h}()$.

The private and public keys of entity A, d_A and P_A respectively, should be produced in accordance with the procedure 7.1 defined in ISO/IEC 15946-1.

6.2 Signature Generation Process

The input to the signature process consists of:

— the domain parameters;

— the signer's private key d_A;

— the message M.

The output of the signature generation process is a pair $(r,s) \in \boldsymbol{F}(n)^* \times \boldsymbol{F}(n)^*$ that constitutes A's digital signature of the message M.

To sign a message M, A executes the following steps:

6.2.1 Calculation of the message digest

1. Compute the hash-code $e = \boldsymbol{h}(M)$.

6.2.2 Elliptic Curve Computations (Arithmetic operations in the underlying field)

2. Select a random integer k in the interval $\{1, \ldots, n-1\}$.

3. Compute the elliptic curve point $(x_1, y_1) = kG$.

6.2.3 Computations modulo the group order of G. (Arithmetic operations in $\boldsymbol{F}(n)$)

4. Set $r = \pi(kG) \bmod n$.

5. Compute k^{-1} in $\boldsymbol{F}(n)$.

6. Compute $s = (d_A r + e)\, k^{-1} \bmod n$.

If the signature generation process yields either $s = 0$ or $r = 0$ then the process of signature generation must be repeated with a new random value k. (But note that the probability that either $r = 0$ or $s = 0$ is negligibly small if k is chosen as described in clauses 6.2.2 and 4.2.1)

NOTE Since the computation of r is independent of any message to be signed, r may be precomputed and stored for a later one-time use in a signing operation.

6.3 The Signature

The pair $(r,s) \in \boldsymbol{F}(n)^* \times \boldsymbol{F}(n)^*$ constitutes A's digital signature of the message M.

6.4 Signature Verification Process

The signature verification process consists of four steps: signature size verification, calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— the domain parameters;

— A's public key P_A;

— the received message, M';

— the received signature of M, represented as the two integers, r' and s'.

To verify A's signature of message M', B executes the following steps:

6.4.1 Signature Size Verification

1. Verify that $0 < r' < n$ and $0 < s' < n$; if not, then reject the signature.

6.4.2 Calculation of the message digest

2. Compute the hash-code $e' = \boldsymbol{h}(M')$ using the hash function $\boldsymbol{h}()$.

6.4.3 Elliptic Curve Computations

3. Compute $w = (s')^{-1} \bmod n$.

4. Compute $u_1 = e'w \bmod n$ and $u_2 = r'w \bmod n$.

5. Compute the elliptic curve point $(x_1, y_1) = u_1 G + u_2 P_A$.

6.4.4 Signature Checking

6. Compute $v = \pi((x_1, y_1)) \bmod n$.

If $r' = v$, then the signature shall be accepted by the verifier. If $r' \neq v$, then the signature shall be rejected by the verifier.

7 EC-KCDSA

The signature scheme EC-KCDSA is the elliptic curve analogue of the KCDSA signature scheme. It is an example of a mechanism producing a digital signature with appendix.

7.1 Domain and User Parameters

The bit length of n should be greater than or equal to the output bit length of the hash function $\boldsymbol{h}()$.

The private and public keys of entity A, d_A and P_A respectively, should be produced in accordance with the procedure 7.3 defined in ISO/IEC 15946-1.

The entity A has a hash-code z_A of *Cert_Data* which is public information. Here *Cert_Data* denotes the certification data of A, which contains at least A's distinguished identifier, public key P_A and all of the domain parameters.

NOTE Assuming the certificate includes all the above, the most straightforward implementation would be for *Cert_Data* to be the certificate itself.

7.2 Signature Generation Process

The input to the signature process consists of:

— the domain parameters;

— the signer's private key d_A;

— the signer's hash-code z_A of the *Cert_Data*;

— the message M.

The output of the signature generation process is a pair (r, s) that constitutes A's digital signature of the message M. The first part r of the signature is a hash-code and the second part s is a positive integer less than n.

To sign a message M, A executes the following steps:

7.2.1 Calculation of the message digest

1. Compute the hash-code $e = \boldsymbol{h}(z_A || M)$.

7.2.2 Elliptic Curve Computations (Arithmetic operations in the underlying field)

2. Select a random integer k in the interval $\{1, \ldots, n-1\}$.

3. Compute the elliptic curve point $(x_1, y_1) = kG$.

4. Set c to be the converted byte string of x_1.

5. Compute the hash-code $r = \boldsymbol{h}(kG) = \boldsymbol{h}(c)$.

NOTE Since the computation of r is independent of any message to be signed, r may be precomputed and stored for a later one-time use in a signing operation.

7.2.3 Computations modulo the group order of G (Arithmetic operations in $F(n)$)

6. Compute $w = r$ XOR e. If $w \geq n$, then $w = w - n$.

7. Compute $s = d_A(k - w) \bmod n$.

If the signature generation process yields $s = 0$ then the process of signature generation must be repeated with a new random k. (But note that the probability that $s = 0$ is negligibly small if k is chosen as described in clauses 7.2.1 and 4.2.1)

7.3 The Signature

The pair (r, s) constitutes A's digital signature of the message M.

7.4 Signature Verification Process

The signature verification process consists of four steps: signature size verification, calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— the domain parameters;

— A's public key P_A;

— A's hash-code z_A of the *Cert_Data*;

— the received message M';

— the received signature of M represented as the two integers, r' and s'.

To verify A's signature of message M', B executes the following steps:

7.4.1 Signature Size Verification

1. Verify that $0 < s' < n$ and $len_{r'} \leq len_{h()}$; if not, then reject the signature.

7.4.2 Calculation of the message digest

2. Compute the hash-code $e' = \boldsymbol{h}(z_A||M')$.

7.4.3 Elliptic Curve Computation

3. Compute $w' = r'$ XOR e'. If $w' \geq n$, then $w' = w' - n$.

4. Compute $(x_1', y_1') = s'P_A + w'G$.

7.4.4 Signature Checking

5. Set c to be the converted byte string of x_1'.

6. Compute $v = \boldsymbol{h}(c)$.

If $r' = v$, then the signature shall be accepted by the verifier. If $r' \neq v$, then the signature shall be rejected by the verifier.

NOTE As with the public key P_A, if a non-authentic hash-code z_A is used as an input, the signature verification process will automatically fail.

Annex A
(informative)

Comparison

In this annex, the properties of the three signature algorithms, EC-GDSA, EC-DSA and EC-KCDSA, are compared. Table 1 shows the comparison of the descriptions and conditions, and Table 2 shows the number of operations required to generate a signature or to verify it.

As shown in Table 2, EC-DSA and EC-GDSA use $\pi(\ )$ to convert an elliptic curve point to an integer, but EC-KCDSA uses the hash function $\boldsymbol{h}(\ )$ instead of $\pi(\ )$ (see 7.2.2). The hash function costs more than the conversion function. However, the portion of hash computation in total computation is very small. Furthermore, the use of a hash function in EC-KCDSA allows for a proof of security in the random oracle model (see reference [5]). Moreover, the use of two hash functions in EC-KCDSA allows for a proof of security if one of the hash functions is a random oracle and the other is collision-resistant (see [6]). In the signature generation step and verifying step, EC-KCDSA has no computation of a multiplicative inverse mod n. This computation would take very little time in the overall workload of signing and verifying on most general purpose computers, but it may be quite expensive in a limited computing environment such as smart cards. Therefore, in some environments EC-KCDSA may be more computationally efficient than EC-DSA. The EC-GDSA provides the user with an improved signing procedure, which has no computation of a multiplicative inverse mod n, nor a hash function to convert elliptic curves to integers. The verification is computational equivalent to the verification of the EC-DSA.

EC-DSA has been endorsed by the US federal government, and the Korean government is expected to endorse EC-KCDSA.

Table 1 – Comparison of descriptions and operations

	EC-DSA	EC-GDSA	EC-KCDSA
Security Parameters	n, $\boldsymbol{h}(\ )$		
Condition of n	$n \geq 2^{\lvert h(\ )\rvert}$	$n \geq 2^{\lvert h(\ )\rvert}$	$n > 2^{\lvert h(\ )\rvert - 1}$
Private key	$d_A \in \{1, \cdots, n-1\}$		
Public key Computation	$P_A = d_A G$	$P_A = (d_A^{-1} \bmod n)G$	$P_A = (d_A^{-1} \bmod n)G$
Signature Generation	$k \in \{1, \cdots, n-1\}$ $r = \pi(kG) \bmod n$ $s = (d_A r + \boldsymbol{h}(M))k^{-1} \bmod n$	$k \in \{1, \cdots, n-1\}$ $r = \pi(kG) \bmod n$ $s = (kr - \boldsymbol{h}(M))d_A \bmod n$	$k \in \{1, \cdots, n-1\}$ $r = \boldsymbol{h}(kG)$ $s = d_A(k - r \text{ XOR } \boldsymbol{h}(z_A\|\|M)) \bmod n$
Signature Size	$0 < r < n$, $0 < s < n$	$0 < r < n$, $0 < s < n$	$0 < s < n$, $0 \leq r < 2^{\lvert h(\ )\rvert}$
Signature Verification	$u_1 = s'^{-1}\boldsymbol{h}(M') \bmod n$ $u_2 = s'^{-1} r' \bmod n$ $\pi(u_1 G + u_2 P_A) \bmod n = r'$?	$u_1 = r'^{-1}\boldsymbol{h}(M') \bmod n$ $u_2 = r'^{-1} s' \bmod n$ $\pi(u_1 G + u_2 P_A) \bmod n = r'$?	$e' = r' \text{ XOR } \boldsymbol{h}(z_A\|\|M') \bmod n$ $\boldsymbol{h}(s'P_A + e'G) = r'$?

Table 2 – Comparison of the number of operations

Process	Operation	EC-DSA	EC-GDSA	EC-KCDSA
Signature Generation	$\boldsymbol{h}(\ )$	1	1	2
	$\pi(\ )$	1	1	0
	$k^{-1} \bmod n$	1	0	0
	multiplication in $\boldsymbol{Z}_n$	2	2	1
	addition (or subtraction) in $\boldsymbol{Z}_n$	1	1	1
	scalar multiplication of a curve point	1	1	1
Signature Verification	$\boldsymbol{h}(\ )$	1	1	2
	$\pi(\ )$	1	1	0
	s^{-1} (or r^{-1}) $\bmod n$	1	1	0
	multiplication in $\boldsymbol{Z}_n$	2	2	0
	scalar multiplication of a curve point	2	2	2
	addition of curve points	1	1	1

Annex B
(informative)

Examples

B.1 Numerical example for EC-GDSA

All numbers in this example are given in hexadecimal notation.

B.1.1 Domain and User Parameters

B.1.1.1 Domain Parameters

The finite field $\boldsymbol{F}(p)$ is given by the prime number

p = D148F03F 28C5981C 59D0A732 DF3C94F0 DD0F4405 4C8320AF

Every element $\alpha \in \boldsymbol{F}(p)$ is represented as an integer $0 \le \alpha \le p-1$.

The elliptic curve $\boldsymbol{E}$ over $\boldsymbol{F}(p)$ is

$\boldsymbol{E}: Y^2 = X^3 + a \cdot X + b$,

where

a = 578EC8B0 4D37D261 C37D4472 4C22CC4D 53854A60 82204CDC and

b = 128A1341 26C047E7 D24E3EFA E77B93D1 3C66A4BE 53388217.

Then #($\boldsymbol{E}$) is computed and factored as:

#($\boldsymbol{E}$) = D148F03F 28C5981C 59D0A734 20C0337C 203CACD4 0C7DD24B

$= 3 \cdot n$

where n is given by :

n = 45C2FABF B841DD5E C89AE266 B595667E B5698EF1 597F4619.

A point $G = (x_G, y_G)$ on $\boldsymbol{E}$ that generates the cyclic subgroup of $\boldsymbol{E}$ of prime order n is given by:

x_G = 89565C06 A278E3CE 5BC36D7D F76521F9 E8A13D8B 359DB4CC

y_G = 5FB2293B 428E873C 3A7AD24B 65569F80 83ABDCA9 A406EE2E.

This example uses RIPEMD-160 as the hash function $\boldsymbol{h}(\ )$, which has an output length of 160 bits.

B.1.1.2 User Parameters

The keys for EC-GDSA are produced as follows:

A selects the random integer t in the interval $\{1, \ldots, n-1\}$. In this example we take

t = 268FE05E 70EEFB70 877B1729 061976EF 931F170A C169C230.

Hence, A's private key is

$d_A = t^{-1} \bmod n$ = 40F95B49 A3B1BF55 311A56DF D3B5061E E1DF6439 84D41E35

and his or her public key is the point $P_A = tG = (P_{Ax}, P_{Ay})$ on E, where

P_{Ax} = 0B1A7C6E F6F2FEC9 718F9CA6 D3D2A1CC 8BD95EBD 35340816 and

P_{Ay} = 6286423F BB294E91 3006F359 A5BAF501 058ACB5C 7FF79349.

B.1.2 Signature Generation Process

In order to sign the ASCII-coded message M = 'message digest', A executes the following steps:

B.1.2.1 Calculation of the Message Digest

1. Using the RIPEMD-160 algorithm, A obtains the hash-code

   e = $\boldsymbol{h}(M)$ = 5D0689EF 49D2FAE5 72B881B1 23A85FFA 21595F36.

B.1.2.2 Elliptic Curve Computations (Arithmetic Operations in the Underlying Field *F*(p))

2. A selects a random integer k in the interval {1, … , n – 1}. For this example, A chooses

   k = 19E489 19EA2B66 D69ADF2B 8110B35B 358DAB4D 185D4D85.

3. A computes the elliptic curve point $(x_1, y_1) = kG$:

   x_1 = 1DF6E5EA 619D2CE8 B172B6BE AB159B8E 71A25CF0 B2CB5B19

   y_1 = 999FF1D3 5EB3E01D EDE93288 AF40FD45 4146C160 5681A7AA.

B.1.2.3 Computations modulo the group order of *G* (Arithmetic operations in *F*(*n*))

A computes the numbers r and s:

4. $r = x_1 \bmod n$ = 1DF6E5EA 619D2CE8 B172B6BE AB159B8E 71A25CF0 B2CB5B19

5. s = $(kr - e)d_A \bmod n$ = 3C73FA01 5B0EFF1F D8AEB482 BD15FA58 DCB5F62E 46527403.

B.1.3 The Signature

The pair (r, s) constitutes A's digital signature of the message M.

B.1.4 Signature Verification Process

To verify whether a pair (r', s') of elements from $F(n)$ may constitute A's signature of the message M, B executes the following steps:

B.1.4.1 Signature Size Verification

1. B checks if $0 < r' < n$ and $0 < s' < n$. Taking $r' = r$ and $s' = s$, this is the case. If not the pair (r', s') is rejected by B.

B.1.4.2 Calculation of Message Digest

2. B computes the hash-code

$e' = \boldsymbol{h}(M) =$ 5D0689EF 49D2FAE5 72B881B1 23A85FFA 21595F36.

B.1.4.3 Computation modulo the Group Order of *G*

3. B computes

$w = (r')^{-1} \mod n =$ 32286A32 6113B981 E4F02FA5 B3DD1543 DFD7E4B3 BE52C818

as well as

4. $u_1 = e'w \mod n =$ 30FACBBF 228AE56B 6BCDA2BD AC060CE8 A883364D 9D7649BD and

$u_2 = s'w \mod n =$ 151B7F51 E121E95E 1925FEDB 06D3C0FD 707A5660 DCAC166A

B.1.4.4 Elliptic Curve Computations

5. B computes the point $u_1G + u_2P_A = (x_1, y_1)$ on ***E.***

The corresponding values are

$x_2 =$ B4BD89B4 B8E50D30 4D9ABAD4 F43E9D2C E9984735 A453581B

$y_2 =$ 1DE5984E 331612A0 7389A693 E0B0B6C4 158965B3 922871AF

$x_3 =$ 4BB49DB2 43C387EF BA94CA85 864FC6FD 4A7E8395 CC6F84EA

$y_3 =$ 84834E69 26A627B7 71D69843 1D962FCA 840481FE DB3CFE98

$x_1 =$ 1DF6E5EA 619D2CE8 B172B6BE AB159B8E 71A25CF0 B2CB5B19

$y_1 =$ 999FF1D3 5EB3E01D EDE93288 AF40FD45 4146C160 5681A7AA, where

$u_1G = (x_2, y_2)$ and $u_2P_A = (x_3, y_3)$.

B.1.4.5 Signature Checking

6. Finally, B computes

$v = \pi((x_1, y_1)) \mod n = x_1 \mod n$

$=$ 1DF6E5EA 619D2CE8 B172B6BE AB159B8E 71A25CF0 B2CB5B19.

Since $r' = v$, the signature is accepted by the verifier.

B.2 Numerical Example for EC-DSA

In this example over a finite field of characteristic 2, all integers are represented to base 10 and all field elements are represented as octet strings.

B.2.1 Domain and User Parameters

B.2.1.1 Domain Parameters

The field $\boldsymbol{F}(2^{191})$ is generated by the irreducible polynomial:

$$f = x^{191} + x^{9} + 1.$$

The curve is $\boldsymbol{E}: y^2+xy=x^3+ax^2+b$ over $\boldsymbol{F}(2^{191})$, where:

$a =$ 2866537B 67675263 6A68F565 54E12640 276B649E F7526267,

$b =$ 2E45EF57 1F00786F 67B0081B 9495A3D9 5462F5DE 0AA185EC.

Select a base point $G = (x_G, y_G)$:

$x_G =$ 36B3DAF8 A23206F9 C4F299D7 B21A9C36 9137F2C8 4AE1AA0D

$y_G =$ 765BE734 33B3F95E 332932E7 0EA245CA 2418EA0E F98018FB

G has prime order n:

$n =$ 1569275433846670190958947355803350458831205595451630533029.

B.2.1.2 User Parameters

A's private key d_A:

$d_A =$ 1275552191113212300012030439187146164646146646466749494799.

A's public key P_A: $P_A = d_A G = (x_1, y_1)$:

$x_1 =$ 5DE37E75 6BD55D72 E3768CB3 96FFEB96 2614DEA4 CE28A2E7

$y_1 =$ 55C0E0E0 2F5FB132 CAF416EF 85B229BB B8E13520 03125BA1

B.2.2 Signature Generation Process

The input to the signature process consists of:

- the domain parameters given above;
- the signer's private key d_A, given above;
- the message M = "abc"

B.2.2.1 Calculation of the message digest

1. Compute the hash-code $e = \boldsymbol{h}(M)$:

SHA-1 is applied to M to get:

e = SHA-1(M) = 968236873715988614170569073515315707566766479517

B.2.2.2 Elliptic Curve Computations (Arithmetic operations in the underlying field)

2. Select a random integer k in the interval {1, … , n – 1}.

k = 1542725565216523985789236956265265265235675811949404040041

3. Compute the elliptic curve point (x_1, y_1) = kG.

x_1 = 438E5A11 FB55E4C6 5471DCD4 9E266142 A3BDF2BF 9D5772D5

y_1 = 2AD603A0 5BD1D177 649F9167 E6F475B7 E2FF590C 85AF15DA.

B.2.2.3 Computations modulo the group order of G. (Arithmetic operations in $F(n)$)

4. Set $r = \pi(kG) \bmod n$.

Convert x_1 to an integer $\bar{x}_1$:

$\bar{x}_1$ = 165646981701154173431466964073025487882844318698669706107 7

Set $r = \bar{x}_1 \bmod n$.

r = 871943831648715433557222849269044199972375915350665280 48

Note that $r \neq 0$, OK.

5. Compute k^{-1} in $F(n)$, and compute $s = (d_A r + e)\, k^{-1} \bmod n$.

s = 308992691965804947361541664549085895292153777025772063598

Note that $s \neq 0$, OK.

B.2.3 The Signature

The pair $(r,s) \in F(n)^* \times F(n)^*$ constitutes A's digital signature of the message M.

r = 871943831648715433557222849269044199972375915350665280 48,

s = 308992691965804947361541664549085895292153777025772063598.

B.2.4 Signature Verification Process

B.2.4.1 Signature Size Verification

1. Clearly, both $0 < r' < n$ and $0 < s' < n$.

B.2.4.2 Calculation of the message digest

2. Compute the hash-code $e' = \boldsymbol{h}(M')$ using the hash function $\boldsymbol{h}()$.

SHA-1 is applied to M′ to get:

e' = SHA-1(M′) = 968236873715988614170569073515315707566766479517.

B.2.4.3 Elliptic Curve Computations

3. Compute $w = (s')^{-1} \bmod n$.

w = 952933666850866331568782284754801289889992082635386177703

4. Compute $u_1 = e'w \bmod n$ and $u_2 = r'w \bmod n$.

u_1 = 124888640715470785402243451608406250330179237436 0994400066

u_2 = 527017380977534012168222466016199849611971141652753464154

5. Compute the elliptic curve point $(x_1, y_1) = u_1G + u_2P_A$.

x_1 = 438E5A11 FB55E4C6 5471DCD4 9E266142 A3BDF2BF 9D5772D5

y_1 = 2AD603A0 5BD1D177 649F9167 E6F475B7 E2FF590C 85AF15DA.

B.2.4.4 Signature Checking

6. Compute $v = \pi((x_1, y_1)) \bmod n$.

Convert x_1 to an integer $\bar{x}_1$:

$\bar{x}_1$ = 165646981701154173431466964073025487882844318 6986697061077

Compute $v = \bar{x}_1 \bmod n$:

v = 871943831648715433557222849269044199972375915350665280 48

$v = r'$. OK.

B.3 Numerical example of EC-KCDSA

B.3.1 Numerical example of EC-KCDSA over F(p)

B.3.1.1 Domain parameters

The finite field $\boldsymbol{F}(p)$ is selected as follows:

the prime p, p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF.

the elliptic curve $\boldsymbol{E}$ over $\boldsymbol{F}(p)$ is $\boldsymbol{E}(a, b)$: $Y^2 = X^3 + a \cdot X + b$. where

a = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFC and

b = 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1.

NOTE All the numbers in this example are given in hexadecimal notation. And all integers of long length are denoted as 32-bit blocks.

Then #(***E***) is computed:

#(***E***) = 1 FFFFFFFF FFFFFFFF FFFFFFFF 33BDF06C 28D79363 69A45062.

Here, we select a prime number n of 192 bit length as:

n = #(***E***)/2 = FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831.

A point $G = (x_G, y_G)$ on ***E***, generating a cyclic group of prime order n, is represented as:

x_G = 188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012,

y_G = 7192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811.

This example uses SHA-1 as the hash function ***h***( ), with output length of 160 bits.

B.3.1.2 User Parameters

The keys for EC-KCDSA are produced as follows:

A's private key d_A, d_A = 444811A3 23E03C28 A34CD859 EE2FF1A3 4D1AAF3C B0B5603B,

$d_A^* = d_A^{-1}$ mod n = F2AC455F E54B1E72 27B2272A E97F86A0 F3B76603 090BA248.

A's public key $P_A = d_A^* G = (P_{Ax}, P_{Ay})$, where

P_{Ax} = 793C9E6E F7CF74C4 CB8FFB6F 3A2C1A9F E9AEBBB2 8AA7451A,

P_{Ay} = B0823C74 7BE23AF0 B170AFB8 13239437 789A03AA 9C526783.

The entity A has a hash-code z_A of *Cert_Data* which is public information.

z_A = A9993E3647 06816ABA3E 25717850C2 6C9CD0D89D

B.3.1.3 Signature Generation

To sign a message *M,* A executes the following steps:

B.3.1.3.1 Calculation of Message Digest

In this example, the message is as the following text sentence :

M = This is a test message!

= 54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 6D 65

73 73 61 67 65 21 (as ASCII form)

1. Compute the hash-code $e = \boldsymbol{h}(z_A || M)$.

$z_A||M$ = A9 99 3E 36 47 06 81 6A BA 3E 25 71 78 50 C2 6C 9C D0 D8 9D 54 68

69 73 20 69 73 20 61 20 74 65 73 74 20 6D 65 73 73 61 67 65 21

e = 73C93524C0 EE82E9CF4D CD20EF0991 62FF634A2A

B.3.1.3.2 Elliptic Curve Computations (Arithmetic operations in the underlying field)

2. Select a random integer k in the interval {1, … , n – 1}.

   k = 4B19A072 5424CD33 10B02D8C 8416C98D 64C618BF E935597D

3. Compute the elliptic curve point $(x_1, y_1) = kG$.

   x_1 = DE3DA2DF 1AFC3446 BB5CAA85 0A978D21 19246CDC 0B197E6C

   y_1 = 15DD6151 225CED60 29D3531F 33092512 89B124EB B15D172B

4. Set c to be the converted byte string of x_1.

   c = DE 3D A2 DF 1A FC 34 46 BB 5C AA 85 0A 97 8D 21 19 24 6C DC 0B 19 7E 6C

5. Compute the hash-code $r = \boldsymbol{h}(kG) = \boldsymbol{h}(c)$.

   r = 3CA29800D4 25FCAA51CC B209B4ED5D 6C35210822

B.3.1.3.3 Computations modulo the group order of G (Arithmetic operations in $F(n)$)

6. Compute $w = r$ XOR e.

   w = 4F6BAD2414 CB7E439E81 7F295BE4CC 0ECA424208

7. If $w \geq n$, then $w = w - n$.

   An integer is converted from octet string w as following, which is less than n.

   w = 4F6BAD24 14CB7E43 9E817F29 5BE4CC0E CA424208

8. Compute $s = d_A(k - w) \bmod n$.

   s = F5C7441A FCE560BD F503A1B9 D234B660 4DC49172 CF9918C1

B.3.1.4 The Signature

The pair (r, s) constitutes A's digital signature of the message M.

B.3.1.5 Signature Verification

To verify A's signature of message M', B executes the following steps:

B.3.1.5.1 Signature Size Verification

1. Verify that $0 < s' < n$ and $len_{r'} \leq len_{h(\ )}$; if not, then reject the signature.

B.3.1.5.2 Calculation of Message Digest

2. Compute the hash-code $e' = \boldsymbol{h}(z_A||M')$.

$z_A||M'$ = A9 99 3E 36 47 06 81 6A BA 3E 25 71 78 50 C2 6C 9C D0 D8 9D 54 68
69 73 20 69 73 20 61 20 74 65 73 74 20 6D 65 73 73 61 67 65 21

e' = 73C93524C0 EE82E9CF4D CD20EF0991 62FF634A2A

B.3.1.5.3 Elliptic Curve Computation

3. Compute $w' = r'$ XOR e'.

w' = 4F6BAD2414 CB7E439E81 7F295BE4CC 0ECA424208

4. If $w' \geq n$, then $w' = w' - n$.

The integer is w' converted from octet string w' as following, which is less than n.

w' = 4F6BAD24 14CB7E43 9E817F29 5BE4CC0E CA424208

5. Compute $(x'_1, y'_1) = s'P_A + w'G$.

x'_1 = DE3DA2DF 1AFC3446 BB5CAA85 0A978D21 19246CDC 0B197E6C

y'_1 = 15DD6151 225CED60 29D3531F 33092512 89B124EB B15D172B

NOTE If $s'P_A = (x_2, y_2)$ and $e'G = (x_3, y_3)$ are calculated separately, they are given as:

x_2 = B1D67075 359880F2 F3FD1C75 3413B0DE 1F41ECFD 089C7B71

y_2 = FB03A83D 5E30E26B 9C918D99 7B7003DE 2CD8B59E 2E52408B

x_3 = 1CA4CC26 A04588D7 6DE6E806 CE062446 D090844B DB329EE7

y_3 = 49E7E6D7 0038610F C94FB9CC 63FBFAAC 7FBA8546 6E41C506

B.3.1.5.4 Signature Checking

6. Set c' to be the converted byte string of x'_1.

c' = DE 3D A2 DF 1A FC 34 46 BB 5C AA 85 0A 97 8D 21 19 24 6C DC 0B 19 7E 6C

7. Compute $v = \boldsymbol{h}(x'_1) = \boldsymbol{h}(c')$.

v = 3CA29800D4 25FCAA51CC B209B4ED5D 6C35210822

Since $r' = v$, the signature is accepted by the verifier.

B.3.2 Numerical example of EC-KCDSA over $F(2^m)$

B.3.2.1 Domain parameters

The finite field $F(2^m)$ is selected as follows:

the integer m, $m = 163$, and

a monic irreducible polynomial $f(x)$ of degree m of $F(2^m)$ over $F(2)$, $f(x) = x^{163} + x^8 + x^2 + x + 1$.

Thus, every element $\alpha \in F(2^m)$ is represented as a polynomial $\alpha = a_{m-1}x^{m-1} + ... + a_1x + a_0$ where $a_i \in \{0, 1\}$, and identified as a bit string $a_{m-1}a_{m-2} ... a_1a_0$. We denote the bit string $a_{m-1}a_{m-2} ...a_1a_0$ as 32-bit blocks.

NOTE All the following numbers in this example are given in hexadecimal notation. And all integers of long length are denoted as 32-bit blocks.

The elliptic curve E over $F(2^m)$ is $E(a, b) : Y^2 + XY = X^3 + a X^2 + b$, where

a = 7 2546B543 5234A422 E0789675 F432C894 35DE5242 and

b = 0 C9517D06 D5240D3C FF38C74B 20B6CD4D 6F9DD4D9.

Then #(E) is computed:

#(E) = 8 00000000 00000000 0003CC1F 9104398E 9B5D5F82.

Here, we select a prime number n of 160 bit length as:

n = #(E)/2 = 4 00000000 00000000 0001E60F C8821CC7 4DAEAFC1.

A point $G = (x_G, y_G)$ on E, generating a cyclic group of prime order n, is represented as:

x_G = 7 AF699895 46103D79 329FCC3D 74880F33 BBE803CB

y_G = 1 EC23211B 5966ADEA 1D3F87F7 EA5848AE F0B7CA9F.

This example uses RIPEMD-160 as the hash function $h( )$, with output length of 160 bits.

B.3.2.2 User Parameters

The keys for EC-KCDSA are produced as follows:

A's private key d_A, d_A = 533FC16B FF07B2FE 15E761BB FB5F0B63 3FD43D42.

$d_A^* = d_A^{-1} \bmod n$ = 3 F6D9FC34 5DA92523 8D441C66 96C5283D 13C4346F.

A's public key $P_A = d_A^* G = (P_{Ax}, P_{Ay})$, where

P_{Ax} = 1 A143BF83 23444956 687C03CA 8DD68CCC 9509DA33

P_{Ay} = 3 55D548DB 894D5D47 1F813F2E E3E692D1 75B9B6FD.

The entity A has a hash-code z_A of *Cert_Data* which is public information.

z_A = A9993E3647 06816ABA3E 25717850C2 6C9CD0D89D