

# Information technology — Security techniques — Digital signatures with appendix

## Part 3: Discrete logarithm based mechanisms

ICS 35.040

# National foreword

This British Standard is the UK implementation of ISO/IEC 14888-3:2006+A2:2012, incorporating corrigenda September 2007 and February 2009. It supersedes BS ISO/IEC 14888-3:2006+A1:2010, which is withdrawn.

The start and finish of text introduced or altered by corrigendum is indicated in the text by tags. Text altered by ISO/IEC corrigendum February 2009 is indicated in the text by AC1 AC1.

Amendments 1:2010 and 2:2012 have been appended at the back of this document.

The UK participation in its preparation was entrusted to Technical Committee IST/33, IT — Security techniques.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

**Compliance with a British Standard cannot confer immunity from legal obligations.**

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 29 December 2006

© The British Standards Institution 2012. Published by BSI Standards Limited 2012

ISBN 978 0 580 72151 9

## Amendments/corrigenda issued since publication

| Date           | Comments                                                                                      |
|----------------|-----------------------------------------------------------------------------------------------|
| 31 March 2008  | Implementation of ISO/IEC corrigendum September 2007 (text in <b>F.1</b> replaced)            |
| 31 August 2009 | Implementation of ISO/IEC corrigendum February 2009                                           |
| 31 August 2010 | Implementation of ISO/IEC amendment 1:2010; Amendment implemented at the back of the document |
| 31 August 2012 | Implementation of ISO/IEC amendment 2:2012                                                    |

# INTERNATIONAL STANDARD

# ISO/IEC 14888-3

Second edition  
2006-11-15

---

---

## Information technology — Security techniques — Digital signatures with appendix —

### Part 3: Discrete logarithm based mechanisms

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

*Partie 3: Mécanismes basés sur un logarithme discret*



Reference number  
ISO/IEC 14888-3:2006(E)

**AVA** Documents  
Document Sharing Hub



## Contents

Page

|                                                                     |     |
|---------------------------------------------------------------------|-----|
| Foreword.....                                                       | vi  |
| Introduction .....                                                  | vii |
| 1 Scope .....                                                       | 1   |
| 2 Normative references .....                                        | 1   |
| 3 Terms and definitions .....                                       | 2   |
| 4 Symbols .....                                                     | 2   |
| 5 General model .....                                               | 4   |
| 5.1 Parameter generation process .....                              | 4   |
| 5.1.1 Certificate-based mechanisms .....                            | 4   |
| 5.1.2 Identity-based mechanisms .....                               | 4   |
| 5.1.3 Parameter selection .....                                     | 5   |
| 5.1.4 Validity of domain parameters and verification key .....      | 5   |
| 5.2 Signature process .....                                         | 6   |
| 5.2.1 Producing the randomizer .....                                | 7   |
| 5.2.2 Producing the pre-signature .....                             | 7   |
| 5.2.3 Preparing the message for signing .....                       | 7   |
| 5.2.4 Computing the witness (the first part of the signature) ..... | 7   |
| 5.2.5 Computing the assignment .....                                | 7   |
| 5.2.6 Computing the second part of the signature .....              | 8   |
| 5.2.7 Constructing the appendix .....                               | 8   |
| 5.2.8 Constructing the signed message .....                         | 8   |
| 5.3 Verification process .....                                      | 9   |
| 5.3.1 Retrieving the witness .....                                  | 10  |
| 5.3.2 Preparing message for verification .....                      | 10  |
| 5.3.3 Retrieving the assignment .....                               | 10  |
| 5.3.4 Recomputing the pre-signature .....                           | 10  |
| 5.3.5 Recomputing the witness .....                                 | 10  |
| 5.3.6 Verifying the witness .....                                   | 10  |
| 6 Certificate-based mechanisms .....                                | 11  |
| 6.1 DSA .....                                                       | 11  |
| 6.1.1 Parameters .....                                              | 12  |
| 6.1.2 Generation of signature key and verification key .....        | 12  |
| 6.1.3 Signature process .....                                       | 12  |
| 6.1.4 Verification process .....                                    | 13  |
| 6.2 KCDSA .....                                                     | 14  |
| 6.2.1 Parameters .....                                              | 15  |
| 6.2.2 Generation of signature key and verification key .....        | 15  |
| 6.2.3 Signature process .....                                       | 15  |
| 6.2.4 Verification process .....                                    | 16  |
| 6.3 Pointcheval/Vaudenay algorithm .....                            | 17  |
| 6.3.1 Parameters .....                                              | 17  |
| 6.3.2 Generation of signature key and verification key .....        | 18  |
| 6.3.3 Signature process .....                                       | 18  |
| 6.3.4 Verification process .....                                    | 19  |
| 6.4 EC-DSA .....                                                    | 19  |
| 6.4.1 Parameters .....                                              | 20  |
| 6.4.2 Generation of signature key and verification key .....        | 20  |
| 6.4.3 Signature process .....                                       | 20  |
| 6.4.4 Verification process .....                                    | 21  |

|         |                                                                                    |    |
|---------|------------------------------------------------------------------------------------|----|
| 6.5     | EC-KCDSA .....                                                                     | 22 |
| 6.5.1   | Parameters .....                                                                   | 22 |
| 6.5.2   | Generation of signature key and verification key .....                             | 23 |
| 6.5.3   | Signature process .....                                                            | 23 |
| 6.5.4   | Verification process .....                                                         | 24 |
| 6.6     | EC-GDSA .....                                                                      | 24 |
| 6.6.1   | Parameters .....                                                                   | 25 |
| 6.6.2   | Generation of signature key and verification key .....                             | 25 |
| 6.6.3   | Signature process .....                                                            | 25 |
| 6.6.4   | Verification process .....                                                         | 26 |
| 7       | Identity-based mechanisms .....                                                    | 27 |
| 7.1     | IBS-1 .....                                                                        | 27 |
| 7.1.1   | Parameters .....                                                                   | 28 |
| 7.1.2   | Generation of master key and signature/verification key .....                      | 28 |
| 7.1.3   | Signature process .....                                                            | 28 |
| 7.1.4   | Verification process .....                                                         | 29 |
| 7.2     | IBS-2 .....                                                                        | 30 |
| 7.2.1   | Parameters .....                                                                   | 30 |
| 7.2.2   | Generation of master key and signature/verification key .....                      | 30 |
| 7.2.3   | Signature process .....                                                            | 30 |
| 7.2.4   | Verification process .....                                                         | 31 |
| Annex A | (normative) ASN.1 module .....                                                     | 33 |
| Annex B | (normative) Conversion functions (I) .....                                         | 36 |
| B.1     | Conversion from a field element to an integer ( <i>FE2I</i> ) .....                | 36 |
| B.2     | Conversion from an integer to a field element ( <i>I2FE</i> ) .....                | 36 |
| B.3     | Conversion from a field element to a bit sequence ( <i>FE2BS</i> ) .....           | 36 |
| B.4     | Conversion from a bit sequence to an integer ( <i>BS2I</i> ) .....                 | 36 |
| B.5     | Conversion from an integer to a bit sequence ( <i>I2BS</i> ) .....                 | 37 |
| B.6     | Conversion between an integer and an octet string ( <i>I2OS &amp; OS2I</i> ) ..... | 37 |
| Annex C | (informative) Conversion functions (II) .....                                      | 38 |
| C.1     | Conversion from an integer to a point ( <i>I2P</i> ) .....                         | 38 |
| Annex D | (normative) Generation of DSA domain parameters .....                              | 40 |
| D.1     | Generation of the prime <i>p</i> and <i>q</i> .....                                | 40 |
| D.2     | Generation of the generator <i>G</i> .....                                         | 41 |
| D.2.1   | Unverifiable generation of <i>G</i> .....                                          | 41 |
| D.2.2   | Verifiable generation of <i>G</i> .....                                            | 41 |
| Annex E | (informative) The Weil and Tate pairings .....                                     | 42 |
| E.1     | The functions <i>f</i> , <i>g</i> and <i>d</i> .....                               | 42 |
| E.2     | The Weil pairing .....                                                             | 43 |
| E.3     | The Tate pairing .....                                                             | 43 |
| Annex F | (informative) Numerical examples .....                                             | 45 |
| F.1     | DSA mechanism .....                                                                | 45 |
| F.1.1   | Example 1 .....                                                                    | 45 |
| F.1.2   | Example 2 .....                                                                    | 46 |
| F.2     | KCDSA mechanism .....                                                              | 48 |
| F.2.1   | Parameters .....                                                                   | 48 |

|              |                                                                        |     |
|--------------|------------------------------------------------------------------------|-----|
| F.2.2        | Signature key and verification key.....                                | 49  |
| F.2.3        | Per message data .....                                                 | 49  |
| F.2.4        | Signature .....                                                        | 49  |
| F.2.5        | Verification .....                                                     | 49  |
| F.3          | Pointcheval-Vaudenay mechanism.....                                    | 49  |
| F.3.1        | Parameters .....                                                       | 49  |
| F.3.2        | Signature key and verification key.....                                | 49  |
| F.3.3        | Per message data .....                                                 | 50  |
| F.3.4        | Signature .....                                                        | 50  |
| F.3.5        | Verification .....                                                     | 50  |
| F.4          | EC-DSA mechanism .....                                                 | 50  |
| F.4.1        | Example 1: Field $F_{2m}$ , $m = 191$ .....                            | 50  |
| F.4.2        | Example 2: Field $F_p$ , 192-bit Prime $P$ .....                       | 51  |
| F.5          | EC-KCDSA mechanism .....                                               | 52  |
| F.5.1        | Example 1: Field $F_{2m}$ , $m = 163$ .....                            | 52  |
| F.5.2        | Example 2: Field $F_p$ , 192-bit Prime $P$ .....                       | 53  |
| F.5.3        | Example 2: Field $F_{Pm}$ , 32-bit $P$ and $m = 5$ .....               | 54  |
| F.6          | EC-GDSA mechanism .....                                                | 55  |
| F.6.1        | Domain and User Parameters.....                                        | 55  |
| F.6.2        | Example 1: Field $F_p$ , 192-bit Prime $P$ .....                       | 55  |
| F.7          | IBS-1 mechanism .....                                                  | 56  |
| F.7.1        | Example 1: Field $F_p$ , 512-bit Prime $p$ .....                       | 56  |
| F.7.2        | Example 2: Field $F_p$ , 512-bit Prime $p$ .....                       | 58  |
| F.8          | IBS-2 mechanism .....                                                  | 60  |
| F.8.1        | Example 1: Field $F_p$ , 512-bit Prime $p$ .....                       | 60  |
| Annex G      | (informative) Comparison of the signature schemes .....                | 64  |
| G.1          | Symbols and abbreviated terms for comparing the signature schemes..... | 64  |
| G.2          | Comparison of the signature schemes .....                              | 64  |
| Annex H      | (informative) Claimed features for choosing a mechanism .....          | 66  |
| Bibliography | .....                                                                  | 67  |
| Amendment 1  | .....                                                                  | 69  |
| Amendment 2  | .....                                                                  | 100 |

## 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.

ISO/IEC 14888-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, Subcommittee SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 14888-3:1998), which has been technically revised. It also incorporates Technical Corrigendum ISO/IEC 14888-3:1998/Cor.1:2001. New mechanisms and object identifiers have been specified.

ISO/IEC 14888 consists of the following parts, under the general title *Information technology — Security techniques — Digital signatures with appendix*:

- *Part 1: General*
- *Part 2: Integer factorization based mechanisms*
- *Part 3: Discrete logarithm based mechanisms*

## Introduction

Digital signature mechanisms can be used to provide services such as entity authentication, data origin authentication, non-repudiation, and data integrity. A digital signature mechanism satisfies the following requirements.

- Given either or both of the following two things:
  - the verification key but not the signature key,
  - a set of signatures on a sequence of messages that an attacker has adaptively chosen,it should be computationally infeasible for the attacker:
  - to produce a valid signature on a new message,
  - to produce a new signature on a previously signed message, or
  - to recover the signature key.
- It should be computationally infeasible, even for the signer, to find two different messages with the same signature.

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

Digital signature mechanisms are based on asymmetric cryptographic techniques and involve three basic operations.

- A process for generating pairs of keys, where each pair consists of a private signature key and the corresponding public verification key.
- A process that uses the signature key, called the signature process.
- A process that uses the verification key, called the verification process.

There are two types of digital signature mechanisms.

- When, for a given signature key, any two signatures produced for the same message are always identical, the mechanism is said to be non-randomized (or deterministic); see ISO/IEC 14888-1.
- When, for a given message and signature key, each application of the signature process produces a different signature, the mechanism is said to be randomized.

The eight mechanisms specified in this part of ISO/IEC 14888 are all randomized.

Digital signature mechanisms can also be divided into the following two categories.

- When the whole message has to be stored and/or transmitted along with the signature, the mechanism is termed a "signature mechanism with appendix" (which is the subject of ISO/IEC 14888).
- When the whole message, or part of it, can be recovered from the signature, the mechanism is termed a "signature mechanism giving message recovery" (see ISO/IEC 9796).

Security of the digital signature mechanisms is based on unsolvable problems, i.e. problems for which, given current knowledge, finding a solution is computationally infeasible, such as the factorization problem and the discrete logarithm problem. ISO/IEC 14888-3 specifies digital signature mechanisms with appendix based on the discrete logarithm problem, and ISO/IEC 14888-2 specifies digital signature mechanisms with appendix based on the factorization problem.

NOTE – The previous version of ISO/IEC 14888 grouped identity-based mechanisms into Part 2 and certificate-based mechanisms into Part 3, with each of the two parts covering mechanisms based on both the discrete logarithm and the factorisation problems. This revision re-organizes the grouping, so that Part 2 contains integer factoring based mechanisms and Part 3 discrete logarithm based mechanisms.

This part of ISO/IEC 14888 includes eight mechanisms, two of which were in ISO/IEC 14888-3:1998, and three of which are in ISO/IEC 15946-2:2002. The Korean Certificate-based Digital Signature Algorithm (KCDSA) and two mechanisms based on pairing technology are newly added.

The mechanisms specified in this part of ISO/IEC 14888 use a collision resistant hash-function for hashing the entire message (possibly in more than one part). ISO/IEC 10118 specifies hash-functions.

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

The ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured the ISO and IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the ISO and IEC. Information may be obtained from:

*ISO/IEC JTC 1/SC 27 Standing Document 8 (SD 8) "Patent Information"*. SD 8 is publicly available at:  
<http://www.ni.din.de/sc27>

Further information is available from the identified patent-holders.

| Area | Inventors | Patent       | Issue date | Contact address       |
|------|-----------|--------------|------------|-----------------------|
| DSA  | Kravitz   | US 5 231 668 | 1993-07-27 | [no licence required] |

Attention is drawn to the possibility that some of the elements of this document 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 — Digital signatures with appendix —

## Part 3: Discrete logarithm based mechanisms

### 1 Scope

This part of ISO/IEC 14888 specifies digital signature mechanisms with appendix whose security is based on the discrete logarithm problem. This part of ISO/IEC 14888 provides

- a general description of a digital signature with appendix mechanism;
- a variety of mechanisms that provide digital signatures with appendix.

For each mechanism, this part of ISO/IEC 14888 specifies

- the process of generating a pair of keys;
- the process of producing signatures;
- the process of verifying signatures.

The verification of a digital signature requires the signing entity's verification key. It is thus essential for a verifier to be able to associate the correct verification key with the signing entity, or more precisely, with (parts of) the signing entity's identification data. This association between the signer's identification data and the signer's public verification key can either be guaranteed by an outside entity or mechanism, or the association can be somehow inherent in the verification key itself. In the former case, the scheme is said to be "certificate-based." In the latter case, the scheme is said to be "identity based." Typically, in an identity-based scheme, the verifier can derive the signer's public verification key from the signer's identification data. The digital signature mechanisms specified in this part of ISO/IEC 14888 are classified into certificate-based and identity-based mechanisms.

NOTE – For certificate-based mechanisms, various PKI standards can be used for key management. For further information, see ISO/IEC 11770-3, ISO/IEC 9594-8 (also known as X.509) and ISO/IEC 15945.

### 2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

ISO/IEC 14888-1:1998, *Information technology — Security techniques — Digital signatures with appendix — Part 1: General*

### 3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14888-1 and the following apply.

#### 3.1

##### finite commutative group

finite set  $E$  with the binary operation “ $*$ ” such that

- for all  $a, b, c \in E$ ,  $(a * b) * c = a * (b * c)$ ;
- there exists  $e \in E$  with  $e * a = a$  for all  $a \in E$ ;
- for all  $a \in E$  there exists  $b \in E$  with  $b * a = e$ ;
- for all  $a, b \in E$ ,  $a * b = b * a$ ;
- for all  $a, b \in E$ ,  $a * b \in E$ .

NOTE 1 – If  $a^0 = e$ , and  $a^{n+1} = a * a^n$  (for  $n \geq 0$ ) is defined recursively, the order of  $a \in E$  is the least positive integer  $n$  such that  $a^n = e$ .

NOTE 2 – In some cases, such as when  $E$  is the set of points on an elliptic curve, arithmetic in the finite set  $E$  is described with additive notation.

#### 3.2

##### cyclic group

group  $E$  of  $n$  elements that contains an element  $a \in E$ , called the generator, of order  $n$

#### 3.3

##### pairing

function which takes two elements,  $P$  and  $Q$ , from an elliptic curve cyclic group over a finite field,  $G_1$ , as input, and produces an element from another cyclic group over a finite field,  $G_2$ , as output, and which has the following two properties (where we assume that the cyclic groups  $G_1$  and  $G_2$  have order  $q$ , for some prime  $q$ , and for any two elements  $P, Q$ , the output of the pairing function is written as  $\langle P, Q \rangle$ ).

- Bilinearity: If  $P, P_1, P_2, Q, Q_1, Q_2$  are elements of  $G_1$  and  $a$  is an integer satisfying  $1 \leq a \leq q - 1$ , then
$$\langle P_1 + P_2, Q \rangle = \langle P_1, Q \rangle * \langle P_2, Q \rangle,$$
$$\langle P, Q_1 + Q_2 \rangle = \langle P, Q_1 \rangle * \langle P, Q_2 \rangle, \text{ and}$$
$$\langle [a]P, Q \rangle = \langle P, [a]Q \rangle = \langle P, Q \rangle^a.$$
- Non-degeneracy: If  $P$  is a non-identity element of  $G_1$ ,  $\langle P, P \rangle \neq 1$ .

#### 3.4

##### Trusted Key Generation Centre KGC

trusted third party, which, in an identity-based signature mechanism, generates a private signature key for each signing entity

### 4 Symbols

|              |                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------|
| $a  b$       | concatenation of $a$ and $b$ , in the order specified                                            |
| $a \oplus b$ | bitwise exclusive OR of $a$ and $b$                                                              |
| $a_1, a_2$   | elliptic curve coefficients                                                                      |
| $(A, B, C)$  | a permutation of $(S, T_1, T_2)$ , which specifies the functionality of the signature mechanisms |
| $D$          | a parameter which specifies the relationship between the signature key and the verification key  |
| $E$          | an elliptic curve defined by two elliptic curve coefficients, $a_1$ and $a_2$                    |

|                  |                                                                                                                                                                                                                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\mathcal{E}$    | a finite commutative group; in the mechanisms based on a multiplicative group, elements of $\mathcal{E}$ are in $\mathbb{Z}_p^*$ ; in the mechanisms based on an additive group of elliptic curve points, elements of $\mathcal{E}$ are the points on the elliptic curve $E$ over $GF(r)$                      |
| $\#\mathcal{E}$  | the cardinality of $\mathcal{E}$ ; in the mechanisms based on a multiplicative group, $\#\mathcal{E}$ is $p - 1$ ; in the mechanisms based on an additive group of elliptic curve points, $\#\mathcal{E}$ is the number of points on the elliptic curve $E$ over $GF(r)$                                       |
| $\gcd(N_1, N_2)$ | the greatest common divisor of integers $N_1$ and $N_2$                                                                                                                                                                                                                                                        |
| $G$              | an element of order $q$ in $\mathcal{E}$                                                                                                                                                                                                                                                                       |
| $GF(r)$          | the Galois field of cardinality $r$                                                                                                                                                                                                                                                                            |
| $G_1$            | a cyclic group of prime order $q$ ; elements of $G_1$ are points on an elliptic curve over $GF(r)$                                                                                                                                                                                                             |
| $G_2$            | a cyclic group of prime order $q$ ; elements of $G_2$ are elements of a finite field $GF(r)$                                                                                                                                                                                                                   |
| $H_1$            | a hash-function that converts a data string into an element in $G_1$ (first express the data string as an integer and then convert the integer into a point on $E$ over $GF(r)$ by using the $I2P$ function, see Annex C for details)                                                                          |
| $h, H_2$         | hash-functions, i.e. one of the mechanisms specified in ISO/IEC 10118                                                                                                                                                                                                                                          |
| $ID$             | a data string containing an identifier of the signer, used in Mechanisms IBS-1 and IBS-2                                                                                                                                                                                                                       |
| $m$              | an embedding degree (or extension degree)                                                                                                                                                                                                                                                                      |
| $[n]P$           | multiplication operation that takes a positive integer $n$ and a point $P$ on the curve $E$ as input and produces as output another point $Q$ on the curve $E$ , where $Q = [n]P = P + P + \dots + P$ added $n - 1$ times. The operation satisfies $[0]P = 0_E$ (the point at infinity), and $[-n]P = [n](-P)$ |
| $P$              | a generator of $G_1$ which is used in Mechanisms IBS-1 and IBS-2                                                                                                                                                                                                                                               |
| $p$              | a prime number or a prime power                                                                                                                                                                                                                                                                                |
| $q$              | a divisor of $\#\mathcal{E}$ and the order of $G_1$ and $G_2$ , which is a prime number                                                                                                                                                                                                                        |
| $r$              | the size of $GF(r)$ ; in the mechanisms based on an additive group of elliptic curve points, $r$ is a prime power, $p^m$ , for some prime $p \geq 2$ and integer $m \geq 1$ .                                                                                                                                  |
| $T$              | the assignment                                                                                                                                                                                                                                                                                                 |
| $T_1$            | the first part of the assignment $T$                                                                                                                                                                                                                                                                           |
| $T_2$            | the second part of the assignment $T$                                                                                                                                                                                                                                                                          |
| $U$              | KGC's master private key, which is a randomly chosen integer used in Mechanisms IBS-1 and IBS-2                                                                                                                                                                                                                |
| $V$              | KGC's master public key, which is an element of $G_1$ used in Mechanisms IBS-1 and IBS-2                                                                                                                                                                                                                       |
| $\mathbb{Z}_N^*$ | the set of integers $U$ with $0 < U < N$ and $\gcd(U, N) = 1$ , with arithmetic defined modulo $N$                                                                                                                                                                                                             |
| $\alpha$         | the bit-length of a prime number (or prime power) $p$                                                                                                                                                                                                                                                          |
| $\beta$          | the bit-length of a prime number $q$                                                                                                                                                                                                                                                                           |
| $\gamma$         | the output bit-length of hash-functions $h$ and $H_2$                                                                                                                                                                                                                                                          |
| $\Pi_x$          | x-coordinate of $\Pi$                                                                                                                                                                                                                                                                                          |
| $0_E$            | the point at infinity on the elliptic curve $E$                                                                                                                                                                                                                                                                |
| $< >$            | a bilinear and non-degenerate pairing                                                                                                                                                                                                                                                                          |

## 5 General model

### 5.1 Parameter generation process

#### 5.1.1 Certificate-based mechanisms

##### 5.1.1.1 Generation of domain parameters

For digital signature mechanisms based on discrete logarithms, the set of domain parameters includes the following parameters:

- $\mathcal{E}$ , a finite commutative group;
- $q$ , a prime divisor of  $\#\mathcal{E}$ ;
- $G$ , an element of order  $q$  in  $\mathcal{E}$

In the group  $\mathcal{E}$ , multiplicative notation is used. It is worthwhile to note that the particular signature mechanism chosen may place additional constraints on the choice of  $\mathcal{E}$ ,  $q$ , and  $G$ .

##### 5.1.1.2 Generation of signature key and verification key

A signature key of a signing entity is a secretly generated random or pseudo-random integer  $X$  such that  $0 < X < q$ . The corresponding public verification key  $Y$  is an element of  $\mathcal{E}$  and is computed as

$$Y = G^{X^D},$$

where  $D$  is a parameter defined by the mechanism to be used. The value of  $D$  is one of two values, -1 and 1.

NOTE – An implementation is still considered compliant if it excludes a few integers from consideration as possible  $X$  values. For example, the value 1 can be excluded because this value results in the user's verification key being the generator,  $G$ , which is easily detectable.

#### 5.1.2 Identity-based mechanisms

##### 5.1.2.1 Notation

The two identity-based mechanisms specified in clause 7 are both based on the use of pairings over elliptic curve groups. As a result we use additive group notation here and throughout clause 7.

##### 5.1.2.2 Generation of domain parameters

For the identity-based digital signature mechanisms based on discrete logarithms, the set of domain parameters includes the following parameters:

- $G_1$ , a cyclic group of prime order  $q$ ;
- $G_2$ , a cyclic group of prime order  $q$ ;
- $P$ , a generator of  $G_1$ ;
- $q$ , a prime number - the cardinality of  $G_1$  and  $G_2$

##### 5.1.2.3 Generation of master key

A master private key of a KGC is a secretly generated random or pseudo-random integer  $U$  such that  $0 < U < q$ . The corresponding master public key  $V$  is an element of  $G_1$  and is computed as

$$V = [U]P.$$

#### 5.1.2.4 Generation of signature key and verification key

A signature key of a signing entity is an element of  $G_1$  and is computed by the KGC as

$$X = [U]Y,$$

where  $U$  is the KGC's master private key and  $Y$  is the public verification key generated from an identity string  $ID$  and a hash-function  $H_1$ , i.e.,  $Y = H_1(ID)$ .

#### 5.1.3 Parameter selection

##### 5.1.3.1 Selecting parameter size

The bit-lengths of parameters for typical security levels are shown in Table 1. The minimum recommended security level is  $2^{80}$ .

NOTE – Security level means the number of steps in the best known attack on a cryptographic primitive. If  $2^{80}$  steps are required in the best known attack on a hash-function, the security level of the hash-function is  $2^{80}$ . For a comprehensive analysis of parameter sizes, see Silverman [27], and Lenstra and Verheul [22].

It is not necessary to select  $\alpha$ ,  $\beta$ , and  $\gamma$  having same security level; the security level of an implemented signature scheme is the lowest among security levels of parameters.

**Table 1 — Parameter sizes according to the security level**

| Security level | $2^{80}$ | $2^{112}$ | $2^{128}$ | $2^{192}$ | $2^{256}$ |
|----------------|----------|-----------|-----------|-----------|-----------|
| $\alpha$       | 1024     | 2048      | 3072      | 7680      | 15360     |
| $\beta$        | 160      | 224       | 256       | 384       | 512       |
| $\gamma$       | 160      | 224       | 256       | 384       | 512       |

##### 5.1.3.2 Selecting hash-function

Selection of hash-functions should be based on those standardized in ISO/IEC 10118-3. That is,  $h$  and  $H_2$  are one of the mechanisms specified in ISO/IEC 10118-3 and  $H_1$  converts a data string obtained by using one of the mechanisms specified in ISO/IEC 10118-3 into an element in  $G_1$  (see Annex B for details).

NOTE 1 – The hash-functions used in this part of ISO/IEC 14888 should be collision-resistant.

NOTE 2 – The security strength for the selected hash-function must meet or exceed the security strength of the parameters in key generation. The relation between the security levels of a hash-function and parameters is shown in the above clause.

Furthermore, implementations that verify digital signatures shall have a way of securely determining which hash-function was used by the signer. Otherwise an attacker might be able to convince a verifier to use a different weaker, hash-function and thus bypass the intended security level.

#### 5.1.4 Validity of domain parameters and verification key

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

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

- Selection of valid domain parameters from a trusted published source, such as a standard;
- Generation of valid domain parameters by a trusted third party, such as a CA or a KGC;
- Validation of candidate domain parameters by a trusted third party, such as a CA or a KGC;
- For the signer, generation of valid domain parameters by the signer using a trusted system; and
- Validation of candidate domain parameters by the user (i.e., the signer or verifier).

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

- For the signer, generation of the public verification/private signature key pair using a trusted system;
- For the signer or verifier, validation of the public verification key by a trusted third party, such as a CA or a KGC; and
- Validation of the public verification key by the user (i.e., the signer or verifier).

NOTE 1 – Validation of domain parameters and keys is required. However, how to achieve this is outside the scope of this document.

NOTE 2 – The method of authenticating the signer is dependent on the real applications, which is out of the scope of this document.

## **5.2 Signature process**

All of the signature mechanisms in this part of ISO/IEC 14888 make use of a randomizing value  $K$ , which is used (along with the message) to produce a witness  $R$  (the first part of the signature) and an assignment  $(T_1, T_2)$ . The signature for the message is the pair  $(R, S)$  where  $S$  (the second part of the signature) is computed as the solution of a signature equation.

In the certificate-based mechanisms, specified in Clause 6, the signature equation is

$$AK + BX^D + C \equiv 0 \pmod{q},$$

given that  $(A, B, C)$  is a permutation of  $(S, T_1, T_2)$ ,  $X$  is the private signature key and  $D$  is a parameter depending on the particular mechanism.

In the identity-based mechanisms, specified in Clause 7, the signature equation is

$$[K]A + [U^D]B + C \equiv 0_E \text{ (in } G_1\text{)}.$$

Given that  $(A, B, C)$  is a permutation of  $(S, T_1, T_2)$ ,  $U$  is the master private key and  $D$  is a parameter depending on the particular mechanism.

The permutation will be specified or agreed upon when setting up the signature system.

The signature process and the formation of a signed message consist of eight stages (See Figure 1):

- Producing the randomizer
- Producing the pre-signature
- Preparing the message for signing
- Computing the witness

- Computing the assignment (it is not necessary to compute the assignment in the identity-based mechanisms)
- Computing the second part of the signature
- Constructing the appendix
- Constructing the signed message.

In this process, the signing entity makes use of its private signature key, its public verification key (optional) and the domain parameters.

### 5.2.1 Producing the randomizer

For each signature, the signing entity freshly generates a secret randomizer which is an integer  $K$  with  $0 < K < q$ . The output of this stage is  $K$ , which shall be kept secret and destroyed safely after use.

NOTE 1 – The randomizer  $K$  can be considered as an ephemeral key.

NOTE 2 – For the same rationale of 5.1.1.2, an implementation is still considered compliant if it excludes a few integers from consideration as possible  $K$  values.

### 5.2.2 Producing the pre-signature

The inputs to this stage are the randomizer  $K$  and optionally signature key  $X$ , with which the signing entity computes the pre-signature,  $\Pi$ , by using  $K$  and public parameters as input. In the certificate-based mechanisms specified in Clause 6, it is computed as

$$\Pi = G^K$$

in  $\mathcal{E}$ . In the identity-based mechanisms specified in Clause 7, it is individually specified in the mechanisms. The output of this stage is the pre-signature,  $\Pi$ .

### 5.2.3 Preparing the message for signing

In the process of preparing the message, one of  $M_1$  and  $M_2$  becomes message  $M$ , the other becomes empty.

### 5.2.4 Computing the witness (the first part of the signature)

The variables to this stage are the pre-signature  $\Pi$  from 5.2.2 and  $M_1$  from 5.2.3. The values of these variables are taken as inputs to the witness function. The output of the witness function is the witness  $R$ . The witness function is specified in the mechanisms.

### 5.2.5 Computing the assignment

The inputs to the assignment function are the first part of the signature, which is the witness  $R$  from 5.2.4,  $M_2$  from 5.2.3, and, optionally, the verification key  $Y$ . The output of the assignment function is assignment  $T = (T_1, T_2)$ . In the certificate-based mechanisms specified in Clause 6,  $T_1$  and  $T_2$  are integers such that

$$0 < |T_1| < q, 0 < |T_2| < q.$$

In the identity-based mechanisms specified in Clause 7,  $T_1$  and  $T_2$  are elements of  $G_1$ . It is not necessary to compute  $T$  in the identity-based mechanisms.

### 5.2.6 Computing the second part of the signature

The inputs to this stage are the randomizer  $K$  from 5.2.1, the signature key  $X$ , the assignment  $T = (T_1, T_2)$  from 5.2.5, the permutation  $(A, B, C)$  of  $(S, T_1, T_2)$ , a variable  $D$  in 5.1.2 and domain parameter  $q$  as specified in 5.1.1.1 and 5.1.2.1.

In the certificate-based mechanisms, the signing entity forms the signature equation

$$AK + BX^D + C \equiv 0 \pmod{q},$$

and solves the signature equation for  $S$ , the second part of the signature, where  $0 < S < q$ .

In the identity-based mechanisms, the signing entity solves the signature equation for  $S$ , the second part of the signature such that  $S \in G_1$ . This solution satisfies the signature equation

$$[K]A + [U^D]B + C \equiv 0_E \text{ (in } G_1\text{)}.$$

The pair  $(R, S)$  will be called the signature,  $\Sigma$ .

### 5.2.7 Constructing the appendix

The appendix is constructed from the signature and an optional text field, *text*, as  $((R, S), \text{text})$ . The text field could include a certificate that cryptographically ties the public verification key to the identification data of the signing entity.

NOTE – As indicated in ISO/IEC 14888-1, depending on the application, there are different ways of forming the appendix and appending it to the message. The general requirement is that the verifier is able to relate the correct signature to the message. For successful verification, it is also essential that prior to the verification process, the verifier is able to associate the correct verification key with the signature.

### 5.2.8 Constructing the signed message

The signed message is obtained by the concatenation of message  $M$  and the appendix, i.e.,  $M \parallel ((R, S), \text{text})$ .

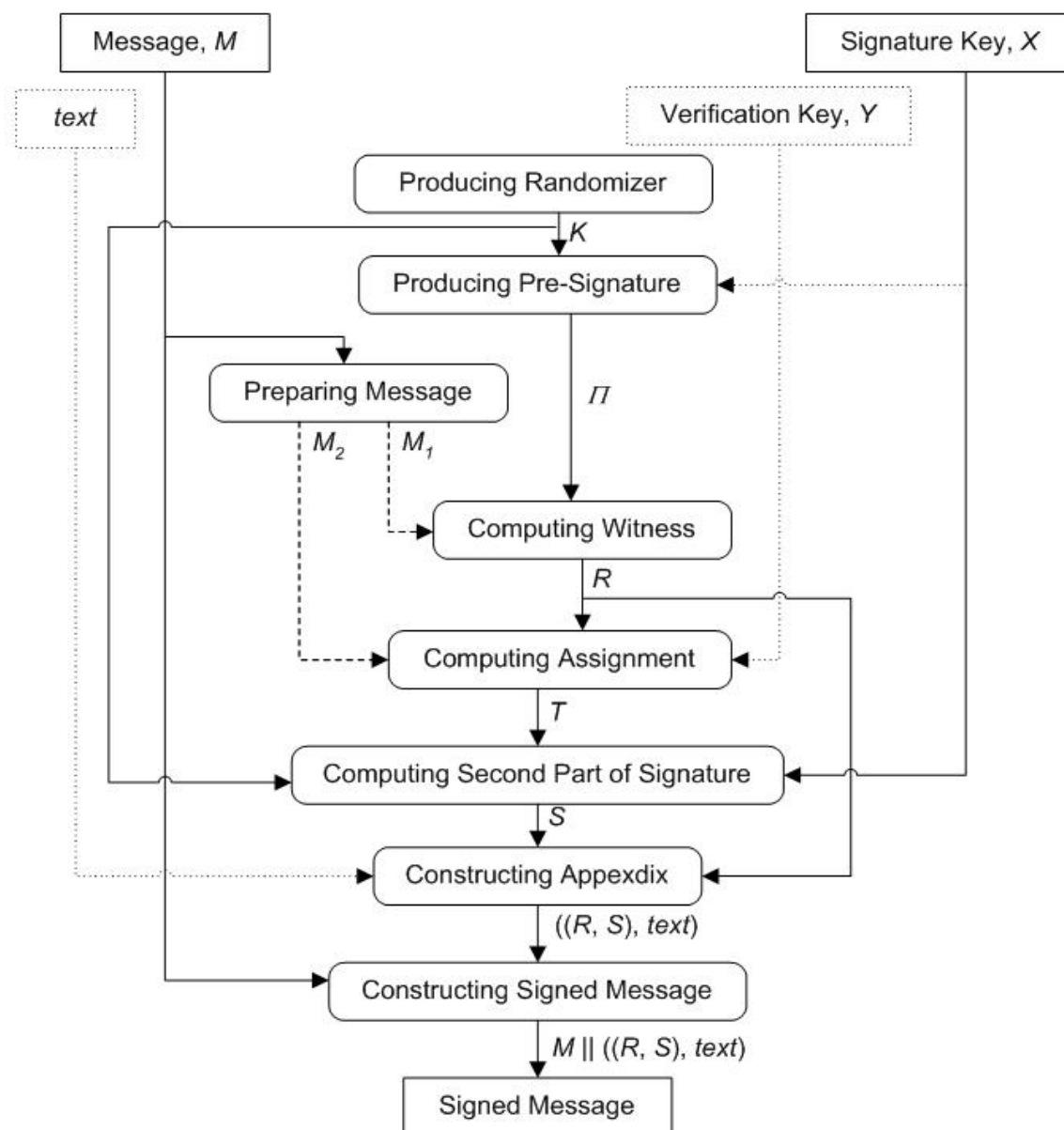


Figure 1 — Signature process with randomized witness (one of  $M_1$  and  $M_2$  is  $M$ , the other is empty)

### 5.3 Verification process

The verification process consists of six stages (See Figure 2).

- Retrieving the witness
- Preparing message for verification
- Retrieving the assignment (it is optional to compute the assignment in the identity-based mechanisms)
- Recomputing the pre-signature
- Recomputing the witness
- Verifying the witness.

In this process, the verifier makes use of the signer's verification key, KGC's master public key (only for the identity-based mechanisms specified in Clause 7) and the domain parameters.

### 5.3.1 Retrieving the witness

The verifier retrieves the signature  $(R, S)$  from the appendix, and divides it into the witness  $R$  and the second part of the signature  $S$ . Also, the verifier checks the range or the bit length of the signature elements,  $R$  and  $S$ , according to the rule specified by each signature process. If the predefined rule is violated, the signature shall be rejected.

### 5.3.2 Preparing message for verification

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

### 5.3.3 Retrieving the assignment

This stage is identical to 5.2.5. The inputs to the assignment function consist of the witness  $R$  from 5.3.1,  $M_2$  from 5.3.2, and (optionally) the verification key  $Y$ . The assignment  $T = (T_1, T_2)$  is recomputed as the output from the assignment function. In the identity-based mechanisms, it is not necessary to recompute  $T$ .

### 5.3.4 Recomputing the pre-signature

The inputs to this stage are the set of domain parameters, the verification key  $Y$ , the assignment  $T = (T_1, T_2)$  from 5.3.3, the second part of the signature  $S$  from 5.3.1, and optionally  $R$  from 5.3.1. The verifier assigns to the coefficients  $(A, B, C)$  the values  $(S, T_1, T_2)$  according to the order specified by the signature function, and in the certificate-based mechanisms, computes the element  $\overline{II}$ .

In the certificate-based mechanisms,  $\overline{II}$  is computed in  $\mathcal{E}$  as

$$\overline{II} = Y^m G^n,$$

where  $m = -A^{-1}B \bmod q$  and  $n = -A^{-1}C \bmod q$ .

In the identity-based mechanisms, it is individually specified in the mechanisms.

### 5.3.5 Recomputing the witness

The computations at this stage are the same as in 5.2.4. The verifier executes the witness function. The inputs are  $\overline{II}$  from 5.3.4 and  $M_1$  from 5.3.2. The output is the recomputed witness,  $\overline{R}$ .

In Mechanism IBS-2 specified in Clause 7.2, the process of recomputing the witness is computing two verification functions, instead of computing  $\overline{R}$ .

### 5.3.6 Verifying the witness

The signature is verified if the recomputed witness,  $\overline{R}$ , from 5.3.5 is equal to  $R$  from 5.3.1.

In Mechanism IBS-2 specified in Clause 7.2, the process of verifying the witness involves checking whether the two verification function values computed in 5.3.5 are identical, instead of verifying whether  $R = \overline{R}$  holds.

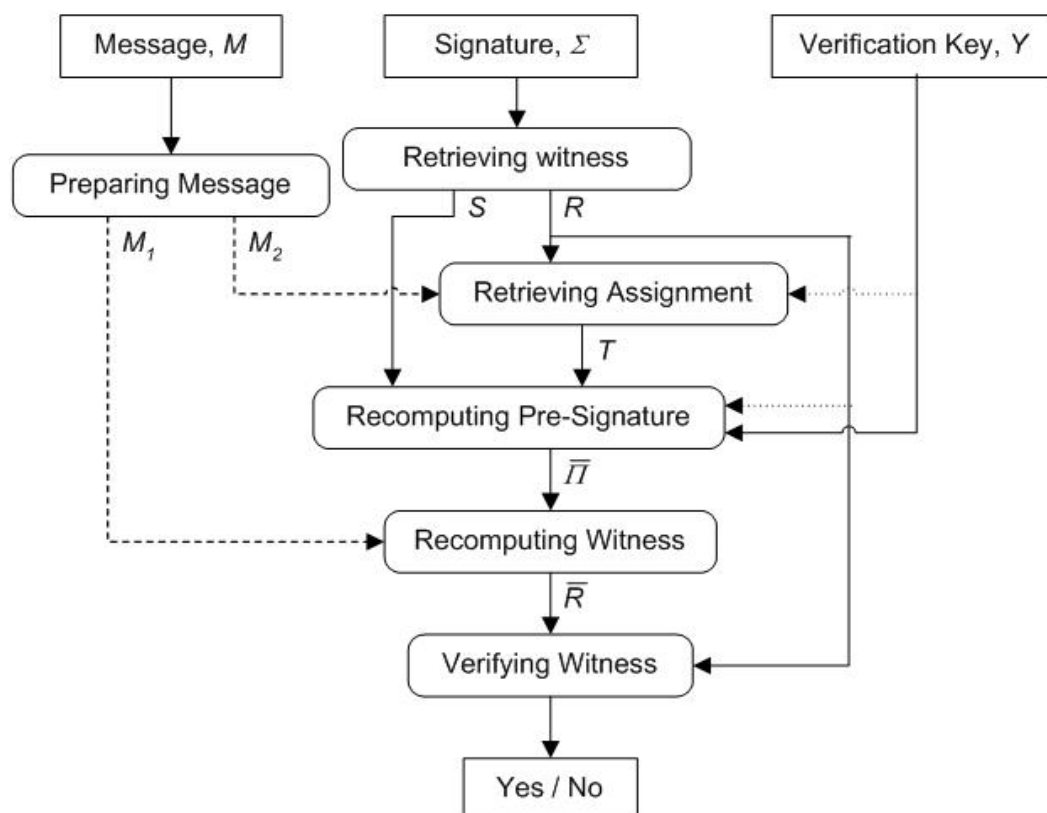


Figure 2 — Verification process with a randomized witness

## 6 Certificate-based mechanisms

This clause specifies six certificate-based mechanisms. These make use of arithmetic in the multiplicative group  $\mathbf{Z}_p^*$  or the additive group elliptic curve points. The mechanisms using arithmetic in the multiplicative group  $\mathbf{Z}_p^*$ , are the Digital Signature Algorithm (DSA), the Korean Certificate-based Digital Signature Algorithm (KCDSA) and the Pointcheval/Vaudenay algorithm. The mechanisms using arithmetic in the additive group of elliptic curve points are the Elliptic Curve DSA (EC-DSA), the Elliptic Curve KCDSA (EC-KCDSA) and the Elliptic Curve German Digital Signature Algorithm (EC-GDSA).

In elliptic curve arithmetic, an elliptic curve point is represented as affine coordinates. That is, an elliptic curve point  $H$  has two coordinates: x-coordinate,  $H_x$ , and y-coordinate,  $H_y$ . Elliptic curves for EC-DSA, EC-KCDSA, and EC-GDSA are restricted to non-singular curves.

The hash-function identifier can be used for binding the signature mechanism and the hash-function.

### 6.1 DSA

DSA (Digital Signature Algorithm) is a signature mechanism with  $\mathbf{E} = \mathbf{Z}_p^*$ ,  $p$  a prime, and  $q$  a prime dividing  $p - 1$ . The parameter  $D$  of DSA 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 witness function is defined by the formula

$$R = H \bmod q,$$

and the assignment function by the formula

$$(T_1, T_2) = (-R, -H),$$

where  $H = h(M)$  is the hash-code of message  $M$ , converted to an integer according to the conversion rule given in Annex B.

NOTE – This mechanism is taken from the U.S. National Institute of Standards and Technology (NIST) Federal Information Processing Standards Publication 186-3 (FIPS PUB 186-3). The notation here has been changed slightly from FIPS PUB 186-3 to conform with the notation used elsewhere in this part of ISO/IEC 14888.

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

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

Thus the signature equation becomes

$$SK - RX - H \equiv 0 \pmod{q}.$$

### 6.1.1 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  $(\alpha, h)$  are allowed in DSA, which are (1024, SHA-1), (2048, SHA-224), (2048, SHA-256), and (3072, SHA-256). Corresponding  $\beta$  should be selected according to  $\alpha$  in Section 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. If compatibility with the NIST Federal standard is not required then the parameters  $p$  and  $q$  can be generated by using the prime generation techniques given in ISO/IEC 18032.

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

NOTE 2 – It is recognized that DSA possesses an unfavourable property in which an attack can be mounted where collisions on the underlying hash-functions can be found with a complexity of  $2^{74}$ ,  $2^{101}$ , and  $2^{114}$  as compared to  $2^{80}$ ,  $2^{112}$  and,  $2^{128}$ , respectively in the most secure case [30]. This attack though is easily detectable. For users who may still wish to avoid this property, it can be prevented by using the mechanisms of 6.2, 6.3 and 6.5.

NOTE 3 – 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.

### 6.1.2 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

$$Y = G^X \pmod{p}.$$

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.

### 6.1.3 Signature process

#### 6.1.3.1 Producing the randomizer

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

### 6.1.3.2 Producing the pre-signature

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

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

### 6.1.3.3 Preparing the message for signing

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

### 6.1.3.4 Computing the witness

The signing entity computes  $R = II \bmod q$  where the witness is simply a function of the pre-signature. Thus,

$$R = (G^K \bmod p) \bmod q.$$

### 6.1.3.5 Computing the assignment

The signing entity computes the assignment  $(T_1, T_2) = (-R, -BS2I(H))$  where  $H = h(M_2)$  is the hash-code of message  $M_2$ . The conversion function,  $BS2I$ , is given in Annex B.

### 6.1.3.6 Computing the second part of the signature

The signature is  $(R, S)$  where  $R$  is computed in 6.1.3.4, and

$$S = (K^{-1} (BS2I(h(M_2)) + XR)) \bmod q.$$

The value of  $h(M_2)$  is an  $\gamma$ -bit string output of the appropriate hash-function in 6.1.1. For use in computing  $S$ , this string shall be converted to an integer.

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).

### 6.1.3.7 Constructing the appendix

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

### 6.1.3.8 Constructing the signed message

A signed message is the concatenation of a message,  $M$ , and the appendix.

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

## 6.1.4 Verification process

Prior to verifying the signature of a signed message, it is necessary that the verifier has trusted copies of  $p$ ,  $q$ ,  $G$  and  $Y$ .

### 6.1.4.1 Retrieving the witness

The verifier retrieves the witness  $R$  and the second part of the signature  $S$  from the appendix. And, the verifier checks to see that  $0 < R < q$  and  $0 < S < q$ . If either condition is violated the signature shall be rejected.

### 6.1.4.2 Preparing the message for verification

The verifier retrieves  $M_2 = M$  from the signed message.  $M_1$  is empty.

### 6.1.4.3 Retrieving the assignment

This stage is identical to 6.1.3.5. The inputs to the assignment function consist of the witness  $R$  from 6.1.4.1 and  $M_2$  from 6.1.4.2. The assignment  $T = (T_1, T_2)$  is recomputed as output from the assignment function, 6.1.3.5.

### 6.1.4.4 Recomputing the pre-signature

The inputs to this stage are domain parameters, verification key  $Y$ , assignment  $T = (T_1, T_2)$  from 6.1.4.3 and second part of the signature  $S$  from 6.1.4.1. The verifier obtains a recomputed value  $\overline{II}$  of the pre-signature using the formula

$$\overline{II} = Y^{-S^{-1}T_1 \bmod q} G^{-S^{-1}T_2 \bmod q} \bmod p.$$

### 6.1.4.5 Recomputing the witness

The computations at this stage are the same as in 6.1.3.4. The verifier executes the witness function. The input is  $\overline{II}$  from 6.1.4.4. Note that  $M_1$  is empty. The output is the recomputed witness  $\overline{R}$  such that  $\overline{R} = \overline{II} \bmod q$ .

### 6.1.4.6 Verifying the witness

The verifier compares the recomputed witness,  $\overline{R}$  from 6.1.4.5 to the value of  $R$  from 6.1.4.1. If  $\overline{R} = R$ , then the signature is valid.

## 6.2 KCDSA

KCDSA (Korean Certificate-based Digital Signature Algorithm) is a signature mechanism with  $\mathcal{E} = \mathbb{Z}_p^*$ ,  $p$  a prime, and  $q$  a prime dividing  $p - 1$ . Verification key  $Y$  is  $G^{X^{-1}}$ ; that is, the parameter  $D$  is -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 witness function is defined by the formula

$$R = h(I2BS(II)).$$

Domain parameters shall indicate the employed hash-function. The assignment function is defined by the formula

$$(T_1, T_2) = (V, -1),$$

where  $V = BS2I(R \oplus H) \bmod q$ . The value  $H$  is the hash-code from the public key  $Y$  and message  $M$ .

NOTE – This mechanism is taken from the Korean Telecommunications Technology Association Standard (TTAS.KO-12.0001/R1), 20 Dec 2000. The notation here has been changed slightly from TTAS.KO-12.0001/R1 to conform with notation used elsewhere in this part of ISO/IEC 14888.

The coefficients  $(A, B, C)$  of the KCDSA signature equation are set as,

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

Thus the signature equation becomes

$$-K + SX^{-1} + V \equiv 0 \pmod{q}.$$

### 6.2.1 Parameters

|          |                                                                     |
|----------|---------------------------------------------------------------------|
| $\alpha$ | $1024+256i$ , for $i$ an integer $0 \leq i \leq 4$                  |
| $\beta$  | $160+32j$ , for $j$ an integer $0 \leq j \leq 3$                    |
| $p$      | a prime, where $2^{\alpha-1} < p < 2^\alpha$                        |
| $q$      | a prime divisor of $p - 1$ , where $2^{\beta-1} < q < 2^\beta$      |
| $F$      | an integer such that $1 < F < p - 1$ and $F^{(p-1)/q} \bmod p > 1$  |
| $G$      | $F^{(p-1)/q} \bmod p$ , an element of order $q$ in $\mathbb{Z}_p^*$ |
| $l$      | the input block size (in bits) of the selected hash-function $h$    |
|          | Hash-function identifier or OID with specified hash-function        |

The integers,  $p$ ,  $q$ ,  $G$ , and  $l$ , 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 KCDSA public parameters.

### 6.2.2 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

$$Y = G^{X^{-1}} \bmod p.$$

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.

### 6.2.3 Signature process

#### 6.2.3.1 Producing the randomizer

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

#### 6.2.3.2 Producing the pre-signature

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

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

#### 6.2.3.3 Preparing the message for signing

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

#### 6.2.3.4 Computing the witness

The signing entity computes  $R = h(I2BS(\Pi))$ , where the output of  $H$  is the hash-code of the bit string of length  $\alpha$  converted from the pre-signature  $\Pi$ . The witness is simply a function of the pre-signature. Thus,

$$R = h(I2BS(G^K \bmod p)).$$

To compute  $R$ , an integer  $I$  shall be converted to a bit string of length  $\alpha$ . The conversion rule,  $I2BS$ , is given in Annex B.

#### **6.2.3.5 Computing the assignment**

The signing entity computes the assignment  $(T_1, T_2) = (V, -1)$  where  $V = BS2I(R \oplus H) \bmod q$ , where  $H = h(Y' || M_2)$  is the hash-code of the concatenation of  $Y' = I2BS(Y \bmod 2^l)$  and message  $M_2$ . The value of  $Y'$  is a bit string of length  $l$ . In computing  $V$ , the bit string  $R \oplus H$  shall be converted to an integer before modulo reduction with respect to  $q$ .

NOTE -  $Y'$  is a fixed value for a user, Thus this value can be kept as a user parameter.

#### **6.2.3.6 Computing the second part of the signature**

The signature is  $(R, S)$  where  $R$  is computed in 6.2.3.4, and

$$S = X(K - V) \bmod q.$$

#### **6.2.3.7 Constructing the appendix**

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

#### **6.2.3.8 Constructing the signed message**

A signed message is the concatenation of a message,  $M$ , and the appendix.

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

### **6.2.4 Verification process**

Prior to verifying the signature of a signed message, it is necessary that the verifier has trusted copies of  $p$ ,  $q$  and  $G$ .

The verifier also acquires the necessary data items for the verification process: for example, the verification key  $Y$  (see ISO/IEC 14888-1:1998, clause 9 for additional required data items).

#### **6.2.4.1 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 the bit length of  $R$  is equal to the output bit length of the employed hash-function  $h$  and  $0 < S < q$ . If either condition is violated the signature shall be rejected.

#### **6.2.4.2 Preparing the message for verification**

The verifier retrieves  $M_2 = M$  from the signed message.  $M_1$  is empty.

#### **6.2.4.3 Retrieving the assignment**

This stage is identical to 6.2.3.5. The inputs to the assignment function consist of the witness  $R$  from 6.2.4.1 and  $M_2$  from 6.2.4.2. The assignment  $T = (T_1, T_2)$  is recomputed as output from the assignment function, 6.2.3.5.

#### 6.2.4.4 Reccomputing the pre-signature

The inputs to this stage are domain parameters, verification key  $Y$ , assignment  $T = (T_1, T_2)$  from 6.2.4.3 and second part of the signature  $S$  from 6.2.4.1. The verifier obtains a recomputed value  $\overline{II}$  of the pre-signature using the formula

$$\overline{II} = Y^{S \bmod q} G^{T_1 \bmod q} \bmod p.$$

#### 6.2.4.5 Reccomputing the witness

The computations at this stage are the same as in 6.2.3.4. The verifier executes the witness function. The input is  $\overline{II}$  from 6.2.4.4. Note that  $M_1$  is empty. The output is the recomputed witness  $\overline{R}$ .

#### 6.2.4.6 Verifying the witness

The verifier compares the recomputed witness,  $\overline{R}$  from 6.2.4.5 to the value of  $R$  from 6.2.4.1. If  $\overline{R} = R$ , then the signature is valid.

### 6.3 Pointcheval/Vaudenay algorithm

The method of Pointcheval/Vaudenay is a variant of the DSA algorithm, with  $\mathcal{E} = \mathbb{Z}_p^*$ ,  $p$  a prime, and  $q$  a prime divisor of  $p - 1$ . 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 witness is defined by the formula

$$R = II \bmod q,$$

and the assignment function is defined by the formula

$$(T_1, T_2) = (-R, -H)$$

where  $H = h(I2BS(R)||M)$  is the hash-code of the concatenation of the witness  $R$  and the message  $M$ . Note that the computation of  $T_2$  above requires the conversion of the hash-code to an integer. The conversion function is given in Annex B.

NOTE – This mechanism is based on the algorithm designed by D. Pointcheval and S. Vaudenay in [26].

The coefficients  $(A, B, C)$  of the Pointcheval/Vaudenay signature equation are set as follows

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

Thus the signature equation becomes

$$SK - RX - H \equiv 0 \pmod{q}.$$

#### 6.3.1 Parameters

$p$  a prime

$q$  a prime divisor of  $p - 1$

$F$  an integer such that  $1 < F < p - 1$  and  $F^{(p-1)/q} \bmod p > 1$

$G$   $F^{(p-1)/q} \bmod p$

Hash-function identifier or OID with specified hash-function

NOTE – It is recommended that all users check the proper generation of the public parameters.

### 6.3.2 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.

### 6.3.3 Signature process

#### 6.3.3.1 Producing the randomizer

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

#### 6.3.3.2 Producing the pre-signature

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

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

#### 6.3.3.3 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.,  $M_2 = M$ .

#### 6.3.3.4 Computing the witness

The signing entity computes  $R = II \bmod q$  where the witness is simply a function of the pre-signature. Thus,

$$R = (G^K \bmod p) \bmod q.$$

#### 6.3.3.5 Computing the assignment

The signing entity computes the assignment  $(T_1, T_2) = (-R, -BS2I(H))$ , where  $H = h(I2BS(R)||M_2)$  is the hash-code of the concatenation of the witness and message  $M_2$ . Before the concatenation, the witness shall be converted to a bit string of length  $|p|$ .

#### 6.3.3.6 Computing the signature

The signature is  $(R, S)$  where  $R$  is computed in 6.3.3.4, and

$$S = K^{-1}(BS2I(H) + XR) \bmod q.$$

#### 6.3.3.7 Constructing the appendix

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

#### 6.3.3.8 Constructing the signed message

A signed message is the concatenation of a message,  $M$ , and the appendix.

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

### 6.3.4 Verification process

Prior to verifying the signature of a signed message, it is necessary that the verifier has trusted copies of  $p$ ,  $q$  and  $G$ .

The verifier also acquires the necessary data items for the verification process: for example, the verification key  $Y$  (see ISO/IEC 14888-1:1998, clause 9 for additional required data items).

#### 6.3.4.1 Retrieving the witness

The verifier retrieves the witness  $R$  and the second part of the signature  $S$  from the appendix. And, the verifier checks to see that  $0 < R < q$  and  $0 < S < q$ . If either condition is violated the signature shall be rejected.

#### 6.3.4.2 Preparing the message for verification

The verifier retrieves  $M_2 = M$  from the signed message.  $M_1$  is empty.

#### 6.3.4.3 Retrieving the assignment

This stage is identical to 6.3.3.5. The inputs to the assignment function consist of the witness  $R$  from 6.3.4.1 and  $M_2$  from 6.3.4.2. The assignment  $T = (T_1, T_2)$  is recomputed as output from the assignment function, 6.3.3.5.

#### 6.3.4.4 Recomputing the pre-signature

The inputs to this stage are domain parameters, verification key  $Y$ , assignment  $T = (T_1, T_2)$  from 6.3.4.3 and second part of the signature  $S$  from 6.3.4.1. The verifier obtains a recomputed value  $\overline{II}$  of the pre-signature by computing it using the formula

$$\overline{II} = Y^{-S^{-1}T_1 \bmod q} G^{-S^{-1}T_2 \bmod q} \bmod p.$$

#### 6.3.4.5 Recomputing the witness

The computations at this stage are the same as in 6.3.3.4. The verifier executes the witness function. The inputs are  $\overline{II}$  from 6.3.4.4. Note that  $M_1$  is empty. The output is the recomputed witness  $\overline{R}$ .

#### 6.3.4.6 Verifying the witness

The verifier compares the recomputed witness,  $\overline{R}$ , from 6.3.4.5 to the value of  $R$  from 6.3.4.1. If  $\overline{R} = R$ , then the signature is valid.

## 6.4 EC-DSA

EC-DSA (Elliptic Curve Digital Signature Algorithm) is an elliptic curve analogue of the DSA algorithm. The coefficients  $(A, B, C)$  of the EC-DSA signature equation are set as follows

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

where  $(T_1, T_2) = (-R, -H)$  and  $H = h(M)$  is the truncated hash-code of message  $M$ , converted to an integer according to the conversion rule given in Annex B. The hash-function  $h$  is one of SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 described in ISO/IEC 10118-3.

NOTE – This mechanism is based on the algorithm described in [5].

Verification key  $Y$  is  $[X]G$ ; 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 witness function is defined by the formula

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

The conversion rule,  $FE2I$ , is given in Annex B.

Thus the signature equation becomes

$$SK - RX - H \equiv 0 \pmod{q}.$$

#### **6.4.1 Parameters**

|       |                                            |
|-------|--------------------------------------------|
| $F$   | a finite field                             |
| $E$   | an elliptic curve group over field $F$     |
| $\#E$ | the cardinality of $E$                     |
| $q$   | a prime divisor of $\#E$                   |
| $G$   | a point on the elliptic curve of order $q$ |

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

NOTE 1 – It is recommended that all users check the proper generation of the EC-DSA public parameters according to X9.62 [5].

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.

#### **6.4.2 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.

#### **6.4.3 Signature process**

##### **6.4.3.1 Producing the randomizer**

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

##### **6.4.3.2 Producing the pre-signature**

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

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

##### **6.4.3.3 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.,  $M_2 = M$ .

#### 6.4.3.4 Computing the witness

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

#### 6.4.3.5 Computing the assignment

The signing entity computes the hash-code; if the output bit length of the selected hash-function is larger than  $\lceil \log_2 q \rceil$ ,  $H$  is set to the leftmost  $\lceil \log_2 q \rceil$  bits of  $h(M_2)$ . Otherwise,  $H$  is  $h(M_2)$ . Afterwards,  $H$  is converted to integer according to conversion rule,  $BS2I$ , in Annex B. The assignment  $(T_1, T_2)$  is  $(-R, -BS2I(H))$ .

#### 6.4.3.6 Computing the second part of the signature

The signature is  $(R, S)$  where  $R$  is computed in 6.4.3.4, and

$$S = (K^{-1}(XR + H)) \bmod q.$$

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).

#### 6.4.3.7 Constructing the appendix

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

#### 6.4.3.8 Constructing the signed message

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

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

### 6.4.4 Verification process

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

#### 6.4.4.1 Retrieving the witness

The verifier retrieves the witness  $R$  and the second part of the signature  $S$  from the appendix. And, the verifier first checks to see that  $0 < R < q$  and  $0 < S < q$ ; if either condition is violated the signature shall be rejected.

#### 6.4.4.2 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  $M_2 = M$ .

#### 6.4.4.3 Retrieving the assignment

This stage is identical to 6.4.3.5. The inputs to the assignment function consist of the witness  $R$  from 6.4.4.1 and  $M_2$  from 6.4.4.2. The assignment  $T = (T_1, T_2)$  is recomputed as output from the assignment function, 6.4.3.5.

#### 6.4.4.4 Recomputing the pre-signature

The inputs to this stage are system parameters, verification key  $Y$ , assignment  $T = (T_1, T_2)$  from 6.4.4.3 and second part of the signature  $S$  from 6.4.4.1. The verifier obtains a recomputed value  $\overline{II}$  of the pre-signature by computing it using the formula

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

#### 6.4.4.5 Recomputing the witness

The computations at this stage are the same as in 6.4.3.4. The verifier executes the witness function. The input is  $\overline{II}$  from 6.4.4.4. The output is the recomputed witness  $\overline{R}$ .

#### 6.4.4.6 Verifying the witness

The verifier compares the recomputed witness,  $\overline{R}$  from 6.4.4.5 to the retrieved version of  $R$  from 6.4.4.1. If  $\overline{R} = R$ , then the signature is verified.

### 6.5 EC-KCDSA

EC-KCDSA (Elliptic Curve Korean Certificate-based Digital Signature Algorithm) is a signature mechanism with verification key  $Y = [X^{-1}]G$ ; 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 witness function is defined by the formula

$$R = h(FE2BS(I_x)).$$

Domain parameters shall indicate the employed hash-function. The assignment function is defined by the formula

$$(T_1, T_2) = (V, -1),$$

where  $V = BS2I((R \oplus H) \bmod q)$ . The value  $H$  is the hash-code from public key  $Y$  and message  $M$ .

NOTE – This mechanism is taken from the Korean Telecommunications Technology Association Standard (TTAS.KO-12.0015), 19 Dec 2001. The notation here has been changed slightly from TTAS.KO-12.0015 to conform with notation used elsewhere in this part of ISO/IEC 14888.

NOTE – The domestic standard for EC-KCDSA has been revised. This is the reason of the slight difference between the definition of EC-KCDSA in this document and the definition in 15946-2.

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

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

Thus the signature equation becomes

$$-K + SX^{-1} + V \equiv 0 \pmod{q}.$$

#### 6.5.1 Parameters

|       |                                                                  |
|-------|------------------------------------------------------------------|
| $I$   | the input block size (in bits) of the selected hash-function $h$ |
| $F$   | a finite field                                                   |
| $E$   | an elliptic curve group over field $F$                           |
| $\#E$ | the cardinality of $E$                                           |
| $q$   | a prime divisor of $\#E$                                         |
| $G$   | 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.

NOTE – It is recommended that all users check the proper generation of the public parameters.

### 6.5.2 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

$$Y = [X^{-1}]G.$$

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.

### 6.5.3 Signature process

#### 6.5.3.1 Producing the randomizer

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

#### 6.5.3.2 Producing the pre-signature

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

$$I_x = [K]G.$$

#### 6.5.3.3 Preparing the message for signing

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

#### 6.5.3.4 Computing the witness

The signing entity computes  $R = h(FE2BS(I_x))$ , where output of  $h$  is the hash-code of  $I_x$ . The witness is simply a function of the pre-signature. To compute  $R$ , the field element  $I_x$  shall be converted to a bit string. The conversion rule,  $FE2BS$ , is given in Annex B.

#### 6.5.3.5 Computing the assignment

The signing entity computes the assignment  $(T_1, T_2) = (V, -1)$  where  $V = BS2I(R \oplus H) \bmod q$ , where  $H = h(Y' || M_2)$  is the hash-code of the concatenation of  $Y'$  and message  $M_2$ . The value of  $Y'$  is the leftmost  $l$  bits of a bit sequence  $FE2BS(Y_x) || FE2BS(Y_y)$ . In computing  $V$ , the bit string  $R \oplus H$  shall be converted to an integer before modulo reduction with respect to  $q$ .

NOTE -  $Y'$  is a fixed value for a user, Thus this value can be kept as a user parameter.

#### 6.5.3.6 Computing the second part of the signature

The signature is  $(R, S)$  where  $R$  is computed in 6.5.3.4, and

$$S = X(K - V) \bmod q.$$

#### 6.5.3.7 Constructing the appendix

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

#### 6.5.3.8 Constructing the signed message

A signed message is the concatenation of a message,  $M$ , and the appendix.

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

#### 6.5.4 Verification process

Prior to verifying the signature of a signed message, it is necessary that the verifier has trusted copies of  $E$ ,  $q$  and  $G$ .

The verifier also acquires the necessary data items for the verification process: for example, the verification key  $Y$  (see ISO/IEC 14888-1:1998, clause 9 for additional required data items).

##### 6.5.4.1 Retrieving the witness

The verifier retrieves the witness  $R$  and the second part of the signature  $S$  from the appendix. And, the verifier first checks to see that the bit length of  $R$  is equal to the output bit length of the employed hash-function  $h$  and  $0 < S < q$ . If either condition is violated the signature shall be rejected.

##### 6.5.4.2 Preparing the message for verification

The verifier retrieves  $M_2 = M$  from the signed message.  $M_1$  is empty.

##### 6.5.4.3 Retrieving the assignment

This stage is identical to 6.5.3.5. The inputs to the assignment function consist of the witness  $R$  from 6.5.4.1 and  $M_2$  from 6.5.4.2. The assignment  $T = (T_1, T_2)$  is recomputed as output from the assignment function, 6.5.3.5.

##### 6.5.4.4 Recomputing the pre-signature

The inputs to this stage are domain parameters, verification key  $Y$ , assignment  $T = (T_1, T_2)$  from 6.5.4.3 and the second part of the signature  $S$  from 6.5.4.1. The verifier obtains a recomputed value  $\overline{II}$  of the pre-signature using the formula

$$\overline{II} = [S \bmod q]Y + [T_1 \bmod q]G.$$

##### 6.5.4.5 Recomputing the witness

The computations at this stage are the same as in 6.5.3.4. The verifier executes the witness function. The input is  $\overline{II}$  from 6.5.4.4. Note that  $M_1$  is empty. The output is the recomputed witness  $\overline{R}$ .

##### 6.5.4.6 Verifying the witness

The verifier compares the recomputed witness  $\overline{R}$  from 6.5.4.5 to the value of  $R$  from 6.5.4.1. If  $\overline{R} = R$ , then the signature is valid.

#### 6.6 EC-GDSA

EC-GDSA (Elliptic Curve German Digital Signature Algorithm) is a signature mechanism with verification key  $Y = [X^1]G$ ; 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 witness function is defined by the formula

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

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

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

where  $(T_1, T_2) = (-R, H)$  and  $H$  is the hash-code of the message  $M$ .

Thus the signature equation becomes

$$-RK + SX^{-1} + H \equiv 0 \pmod{q}.$$

NOTE – EC-GDSA stands for Elliptic Curve German Digital Signature Algorithm (H. Erwin, and S. Pascale. “Digital Signature Scheme EC-GDSA,” German Federal Office for Information Security, December 2005).

### 6.6.1 Parameters

|       |                                                              |
|-------|--------------------------------------------------------------|
| $F$   | a finite field                                               |
| $E$   | an elliptic curve group over field $F$                       |
| $\#E$ | the cardinality of $E$                                       |
| $q$   | a prime divisor of $\#E$                                     |
| $G$   | 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.

NOTE – It is recommended that all users check the proper generation of the public parameters.

### 6.6.2 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  $-1$ . Thus, the corresponding public verification key  $Y$  is

$$Y = [X^{-1}]G.$$

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.

### 6.6.3 Signature process

#### 6.6.3.1 Producing the randomizer

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

#### 6.6.3.2 Producing the pre-signature

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

$$R = [K]G.$$

#### 6.6.3.3 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.,  $M_2 = M$ .

#### 6.6.3.4 Computing the witness

The signing entity computes  $R = FE2I(I_x) \bmod q$  where the witness is simply a function of the pre-signature.