ISO/IEC JTC 1/SC 27/WG 2
Cryptography and security mechanisms
Convenorship: JISC (Japan)

Replaces:	N 1369
Document type:	Standing Document
Title:	WG 2 SD7 -- Conversion functions (2nd Edition)
Status:	
Date of document:	2017-11-21
Source:	Editors (Chris Mitchell, Liqun Chen)
Expected action:	INFO
No. of pages:	1 + 48
Email of convenor:	t-chika@ipa.go.jp
Committee URL:	http://isotc.iso.org/livelink/livelink/open/jtc1sc27wg2

ISO/IEC JTC 1/SC 27/WG 2

Standing Document 7

Conversion Functions

2nd edition

20th November 2017

NOTE All conversion functions should be stated in non-italic font.

Contents

1 Introduction

Data format conversion functions are widely used in SC 27/WG 2 standards. Unfortunately, there are many inconsistencies in the existing functions appearing across the range of WG 2 standards. This document is intended to address this by providing a recommended set of conversion functions intended for use in all new SC 27/WG 2 standards and also in revised editions of existing standards.

A catalogue of conversion functions is provided in section 2, followed by details of the conversion functions appearing in existing WG 2 standards (as of 2017) in section 3. Section 4 provides a recommended set of conversion functions designed for use in all future standards.

Users of this document are recommended to use the catalogue in section 2 to identify the functions needed, and then to select appropriate functions from section 4.

2 Catalogue of conversion functions

The following table lists both the conversion functions that are currently used in SC 27/WG 2 standards and the recommended versions given in this document. In each case the name of the function is given, brief description, a list of the standards that contain a specification of the function (if applicable), together with the section numbers of this document containing descriptions of the existing functions (where they exist) and the recommended function.

Names of functions are given in a non-italic font, and this form is recommended for use in all new and revised standards. As of 2017, practice in existing standards is mixed, and a uniform approach would be beneficial.

Name	Description	Where used	Section number for existing functions	Section number for recommended functions
BS2FE	a function that converts a bit string into a field element			4.3.1
BS2I	a function that converts a bit string into an integer	ISO/IEC 9796-2 ISO/IEC 9796-3 ISO/IEC 10118-4 ISO/IEC 11770-4 ISO/IEC 14888-3 ISO/IEC 15946-1 ISO/IEC 18033-2 ISO/IEC 18370-2	3.1	4.3.2
BS2OS	a function that converts a bit string into an octet string	ISO/IEC 9796-3 ISO/IEC 15946-1 ISO/IEC 18033-2	3.2	4.3.3
EC2OS	a function that converts an elliptic curve point into an octet string	ISO/IEC 9796-3 ISO/IEC 15946-1 ISO/IEC 18033-2	3.3	4.3.4
FE2BS	a function that converts a field element into a bit string	ISO/IEC 14888-3	3.4	4.3.5
FE2I	a function that converts a field element into an integer	ISO/IEC 9796-3 ISO/IEC 11770-4 ISO/IEC 14888-3 ISO/IEC 15946-1	3.5	4.3.6
FE2OS	a function that converts a field element into an octet string	ISO/IEC 9796-3 ISO/IEC 11770-4 ISO/IEC 15946-1 ISO/IEC 18033-2	3.6	4.3.7
$GE2OS_x$	a function that converts a group element into an octet string; when the group element is a point on an elliptic curve E, this function converts the x-coordinate of the point into an octet string and ignores the y-coordinate	ISO/IEC 11770-4	3.7	4.3.8

I2BS	a function that converts an integer into a bit string	ISO/IEC 9796-2 ISO/IEC 9796-3 ISO/IEC 10118-4 ISO/IEC 14888-3 ISO/IEC 15946-1 ISO/IEC 18033-2 ISO/IEC 18370-2	3.8	4.3.9
I2EC	a function that converts an integer into an elliptic curve point	ISO/IEC 15946-1	3.9	4.3.10
I2FE	a function that converts an integer into a field element	ISO/IEC 11770-4 ISO/IEC 14888-3	3.10	4.3.11
I2OS	a function that converts an integer into an octet string	ISO/IEC 9796-3 ISO/IEC 11770-4 ISO/IEC 14888-3 ISO/IEC 15946-1 ISO/IEC 18033-2	3.11	4.3.12
I2P	a function that converts an integer into a point on an elliptic curve	ISO/IEC 11770-4 ISO/IEC 14888-3	3.12	4.3.13
IHF1	a function that converts a bit string into an integer in a specified range	ISO/IEC 18033-5	3.13	4.3.14
OS2BS	a function that converts an octet string into a bit string	ISO/IEC 9796-3 ISO/IEC 15946-1 ISO/IEC 18033-2	3.14	4.3.15
OS2EC	a function that converts an octet string into an elliptic curve point	ISO/IEC 9796-3 ISO/IEC 15946-1 ISO/IEC 18033-2	3.15	4.3.16
OS2FE	a function that converts an octet string into a field element	ISO/IEC 9796-3 ISO/IEC 15946-1 ISO/IEC 18033-2	3.16	4.3.17
OS2I	a function that converts an octet string into an integer	ISO/IEC 9796-3 ISO/IEC 11770-4 ISO/IEC 14888-3 ISO/IEC 15946-1 ISO/IEC 18033-2	3.17	4.3.18
PHF1	a function that converts a bit string into an element of an elliptic curve group (for a supersingular elliptic curve)	ISO/IEC 18033-5	3.18	4.3.19

3 Existing conversion functions

In each case, the text given is taken directly from the relevant standard, with no editing (except for a small number of minor changes to improve readability). Note that in some cases an extraneous 'P' is added to the end of the function name – in all cases these have been removed for consistency.

3.1 BS2I

This section gives the eight existing specifications of the function BS2I (Bit String to Integer conversion). Note that the definitions given in ISO/IEC 9796-2 and ISO/IEC 10118-4 do not use the name BS2I.

3.1.1 ISO/IEC 9796-2 [9796-2]

To represent a bit string $x_{l-1}\, x_{l-2} \ldots x_0$ (of length l) as an integer x, the inverse process [i.e. the function I2BS] shall be followed, i.e. x shall be the integer defined by

$$x = 2^{l-1}x_{l-1} + 2^{l-2}x_{l-2} + \ldots + 2x_1 + x_0.$$

3.1.2 ISO/IEC 9796-3 [9796-3]

The function BS2I(x) maps a bit string x to an integer value x' as follows. If $x = \langle x_{l-1}, \ldots, x_0 \rangle$ where x_0, …, x_{l-1} are bits, then the value x' is defined as

$$x' = \sum_{\substack{0 \le i < l \\ x_i = 1}} 2^i.$$

3.1.3 ISO/IEC 10118-4 [10118-4]

Converting a string to a number

During computation of the round-function, strings of bits need to be converted into integers. Where this is required, the integer shall be made equal to the number having binary representation equal to the binary string, where the left-most bit of the string is considered as the most significant bit of the binary representation.

3.1.4 ISO/IEC 11770-4 [11770-4]

Function BS2I takes a bit string $b_{l-1}\, b_{l-2} \ldots b_0$ as input and produces a non-negative integer as output. It is defined as follows:

1. Let integer y_i have the value of the bit b_i for $0 \le i \le l-1$.
2. Compute the integer $y = y_{l-1}\, 2^{l-1} + y_{l-2}\, 2^{l-2} + \ldots + y_1\, 2 + y_0$.
3. Output y.

- For example, if $l = 19$, BS2I(000 0010 1010 1100 0001) = 10945.

Note that the bit string of length zero (the empty bit string) is converted to the integer 0.

3.1.5 ISO/IEC 14888-3 [14888-3]

Input values

— g – a positive integer, indicating the length of the input string.

— $x = x_{g-1}\, x_{g-2} \ldots x_0$ – a binary string of length g.

Output value

— BS2I(g, x) = $2^{g-1}\, x_{g-1} + 2^{g-2}\, x_{g-2} + \ldots + x_0 \in \{0, 1, \ldots, 2^g - 1\}$.

3.1.6 ISO/IEC 15946-1 [15946-1]

The function BS2I(x) maps a bit string x to an integer value x', as follows. If $x = \langle x_{l-1}, \ldots, x_0 \rangle$ where x_0, …, x_{l-1} are bits, then the value x' is defined as

$$x' = \sum_{\substack{0 \le i < l \\ x_i = 1}} 2^i.$$

3.1.7 ISO/IEC 18033-2 [18033-2]

The function BS2I(x) maps a bit string x to an integer value x' as follows. If $x = \langle x_{l-1}, \ldots, x_0 \rangle$ where x_0, …, x_{l-1} are bits, then the value x' is defined as

$$x' = \sum_{\substack{0 \le i < l \\ x_i = 1}} 2^i.$$

3.1.8 ISO/IEC 18370-2 [18370-2]

The function BS2I(x) maps a bit string x to an integer value m as follows. If $x = (x_{l-1}, \ldots, x_0)$ where x_0, …, x_{l-1} are bits, then the value m is defined as $m = 2^{l-1}x_{l-1} + 2^{l-2}x_{l-2} + \ldots + 2x_1 + x_0$.

3.2 BS2OS

This section gives the three existing specifications of the function BS2OS (Bit String to Octet String conversion).

3.2.1 ISO/IEC 9796-3 [9796-3]

The function BS2OS(y) takes as input a bit string y, whose length is a multiple of 8, and outputs the unique octet string x such that y = OS2BS(x).

3.2.2 ISO/IEC 15946-1 [15946-1]

The function BS2OS(y) takes as input a bit string y, whose length is a multiple of 8, and outputs the unique octet string x such that y = OS2BS(x).

3.2.3 ISO/IEC 18033-2 [18033-2]

The function BS2OS(y) takes as input a bit string y, whose length is a multiple of 8, and outputs the unique octet string x such that y = OS2BS(x).

3.3 EC2OS

This section gives the three existing specifications of the function EC2OS (Elliptic Curve point to Octet String conversion).

3.3.1 ISO/IEC 9796-3 [9796-3]

The function $\text{EC2OS}_E(P, \text{fmt})$ takes as input a point P on E and a format specifier fmt, which is one of the symbolic values `compressed`, `uncompressed`, or `hybrid`. The output is an octet string EP, computed as follows:

1. If P = O, then EP = Oct(0); and
2. If $P = (x, y) \neq$ O, with compressed form $(x, \tilde{y})$, then EP = $H \,||\, X \,||\, Y$, where
   a. H is a single octet of the form $\text{Oct}(4U + C \cdot (2 + \tilde{y}))$, where
      i. U = 1 if fmt is either uncompressed or hybrid, and otherwise, U = 0, and
      ii. C = 1 if fmt is either compressed or hybrid, and otherwise, C = 0,
   b. X is the octet string $\text{FE2OS}_F(x)$, and
   c. Y is the octet string $\text{FE2OS}_F(y)$ if fmt is either uncompressed or hybrid, and otherwise Y is the null octet string.

3.3.2 ISO/IEC 15946-1 [15946-1]

The function $EC2OS_E(P, \text{fmt})$ takes as input a point P on E and a format specifier fmt, which is one of the symbolic values `compressed`, `uncompressed`, or `hybrid`. The output is an octet string EP, computed as follows:

- If $P = O_E$, then EP = Oct(0).
- If $P = (x, y) \neq O_E$, with compressed form $(x, \tilde{y})$, then EP = $H \,||\, X \,||\, Y$, where
  - H is a single octet of the form $\text{Oct}(4U + C \cdot (2 + \tilde{y}))$, where
    - $U = 1$ if fmt is either uncompressed or hybrid, and otherwise, $U = 0$,
    - $C = 1$ if fmt is either compressed or hybrid, and otherwise, $C = 0$,
  - X is the octet string $FE2OS_F(x)$,
  - Y is the octet string $FE2OS_F(y)$ if fmt is either uncompressed or hybrid, and otherwise Y is the null octet string.

3.3.3 ISO/IEC 18033-2 [18033-2]

The function $EC2OS_E(P, \text{fmt})$ takes as input a point P on E and a format specifier fmt, which is one of the symbolic values `compressed`, `uncompressed`, or `hybrid`. The output is an octet string EP, computed as follows:

- If $P = O_E$, then EP = Oct(0).
- If $P = (x, y) \neq O_E$, with compressed form $(x, \tilde{y})$, then EP = $H \,||\, X \,||\, Y$, where
  - H is a single octet of the form $\text{Oct}(4U + C \cdot (2 + \tilde{y}))$, where
    - $U = 1$ if fmt is either uncompressed or hybrid, and otherwise, $U = 0$,
    - $C = 1$ if fmt is either compressed or hybrid, and otherwise, $C = 0$,
  - X is the octet string $FE2OS_F(x)$,
  - Y is the octet string $FE2OS_F(y)$ if fmt is either uncompressed or hybrid, and otherwise Y is the null octet string.

3.4 FE2BS

This section gives the single existing specification of the function FE2BS (Field Element to Bit String conversion).

3.4.1 ISO/IEC 14888-3 [14888-3]

Input values

— r – a prime or a power of a prime.

— x – an element of the Galois field $GF(r)$.

Assumptions

— When $r = p$, where p is an odd prime:

$x \in GF(p)$ is represented as an integer in $\{0, 1, \ldots, p - 1\}$.

— When $r = p^m$, where p is an odd prime and m is an integer greater than 1:

$x \in GF(p^m)$ is represented as a p-ary string of length m:
$x = x_{m-1}\, x_{m-2} \ldots x_0$, where $x_i \in \{0, 1, \ldots, p - 1\}$ for $0 \leq i < m$.

— When $r = 2$:

$x \in GF(2)$ is represented as an integer in $\{0, 1\}$.

— When $r = 2^m$, where m is an integer greater than 1:

$x \in GF(2^m)$ is represented as a binary string of length m:
$x = x_{m-1}\, x_{m-2} \ldots x_0$, where $x_i \in \{0, 1\}$ for $0 \le i < m$.

Output value

— FE2BS(r, x) = I2BS(g, FE2I(r, x)),

where $g = 8\lceil \log_{256}(r) \rceil$

3.5 FE2I

This section gives the four existing specifications of the function FE2I (Field Element to Integer conversion).

3.5.1 ISO/IEC 9796-3 [9796-3]

The function $FE2I_F$ maps an element $a \in F$ to an integer value a', as follows. If the cardinality of F is $q = p^e$, where p is prime and $e \ge 1$, then an element a of F is an e-tuple $(a_1, \ldots, a_e)$, where $a_i \in [0 \ldots p)$ for $1 \le i \le e$, and the value a' is defined as $a' = \sum_{1 \le i \le e} a_i p^{i-1}$.

3.5.2 ISO/IEC 11770-4 [11770-4]

Let an element of a finite field $F(s^m)$ (where s is either p or 2) be represented by $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ where β_i is an integer satisfying $0 \le \beta_i \le s - 1$.

Function FE2I takes an element $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ as input and produces a non-negative integer as output. It is defined as follows:

1. Let integer y_i have the value of β_i for $0 \le i \le m - 1$.
2. Compute the integer $y = y_{m-1}\, s^{m-1} + y_{m-2}\, s^{m-2} + \ldots + y_1\, s + y_0$.
3. Output y.

3.5.3 ISO/IEC 14888-3 [14888-3]

Input values

— r – a prime or a power of a prime.

— x – an element of the Galois field $GF(r)$.

Assumptions

— When $r = p$, where p is an odd prime:

$x \in GF(p)$ is (already) represented as an integer in $\{0, 1, \ldots, p - 1\}$.

— When $r = p^m$, where p is an odd prime and m is an integer greater than 1:

$x \in GF(p^m)$ is represented as a p-ary string of length m:
$x = x_{m-1}\, x_{m-2} \ldots x_0$, where $x_i \in \{0, 1, \ldots, p - 1\}$ for $0 \le i < m$.

— When $r = 2$:

$x \in GF(2)$ is (already) represented as an integer in $\{0, 1\}$.

— When $r = 2^m$, where m is an integer greater than 1:

$x \in GF(2^m)$ is represented as a binary string of length m:
$x = x_{m-1}\, x_{m-2} \ldots x_0$, where $x_i \in \{0, 1\}$ for $0 \le i < m$.

Output value

— When $r = p$, where p is an odd prime:

$FE2I(r, x) = x \in \{0, 1, ..., p - 1\}$.

— When $r = p^m$, where p is an odd prime, m is an integer greater than 1, and x is represented as the p-ary string $x_{m-1}\, x_{m-2} \ldots x_0$:

$FE2I(r, x) = p^{m-1} x_{m-1} + p^{m-2} x_{m-2} + \ldots + x_0 \in \{0, 1, \ldots, p^m - 1\}$.

— When $r = 2$:

$FE2I(r, x) = x \in \{0, 1\}$.

— When $r = 2^m$, where m is an integer greater than 1 and x is represented as the binary string $x_{m-1}\, x_{m-2} \ldots x_0$:

$FE2I(r, x) = 2^{m-1} x_{m-1} + 2^{m-2} x_{m-2} + \ldots + x_0 \in \{0, 1, \ldots, 2^m - 1\}$.

3.5.4 ISO/IEC 15946-1 [15946-1]

The function $FE2I_F$ maps an element $a \in F$ to an integer value a', as follows. If an element a of F is identified with an m-tuple $(a_1, \ldots., a_m)$, where the cardinality of F is $q = p^m$ and $a_i \in [0, p\text{-}1]$ for $1 \le i \le m$, then the value a' is defined as $a' = \sum_{1 \le i \le m} a_i p^{i-1}$.

3.6 FE2OS

This section gives the four existing specifications of the function FE2OS (Field Element to Octet String conversion).

3.6.1 ISO/IEC 9796-3 [9796-3]

The function $FE2OS_F(a)$ takes as input an element a of the field F and outputs the octet string $I2OS(a', l)$, where $a' = FE2I_F(a)$, and l is the length in octets of $|F| - 1$, i.e., $l = \lceil \log_{256} |F| \rceil$. Thus, the output of $FE2OS_F(a)$ is always an octet string of length exactly $\lceil \log_{256} |F| \rceil$.

3.6.2 ISO/IEC 11770-4 [11770-4]

Function FE2OS takes an element $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ as input and produces an octet string y as output. It is defined as follows:

1. Convert $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ in to an integer x by using FE2I.

2. Convert x into an octet string y by using I2OS.

3.6.3 ISO/IEC 15946-1 [15946-1]

The function $FE2OS_F(a)$ takes as input an element a of the field F and outputs the octet string $I2OS(a', l)$, where $a' = FE2I_F(a)$ and $l = L(|F| - 1)$. Thus, the output of $FE2OS_F(a)$ is always an octet string of length exactly $\lceil \log_{256} |F| \rceil$.

NOTE 1 $L(x)$ represents the length in octets of integer x or octet string x (non-negative integer).

3.6.4 ISO/IEC 18033-2 [18033-2]

The function $FE2OS_F(a)$ takes as input an element a of the field F and outputs the octet string $I2OS(a', l)$, where $a' = FE2I_F(a)$, and l is the length in octets of $|F| - 1$, i.e., $l = \lceil \log_{256} |F| \rceil$. Thus, the output of $FE2OS_F(a)$ is always an octet string of length exactly $\lceil \log_{256} |F| \rceil$.

3.7 $GE2OS_X$

This section gives the single existing specification of the function $GE2OS_X$ (Group Element to Octet String conversion).

3.7.1 ISO/IEC 11770-4 [11770-4]

Function $GE2OS_X$ takes a group element as input and produces an octet string as output. It is defined as follows:

In the DL setting, group elements are elements in $F(q)$. Let u be a group element. The output of $GE2OS_X(u)$ is computed as follows:

1. Represent u as a field element.
2. Convert the result of 1) into an octet string using FE2OS.
3. Output the result of 2).

In the EC setting, group elements are points on the elliptic curve E. Let $Q = (x_Q, y_Q)$ be a point on E, where x_Q is the x-coordinate of Q and y_Q is the y-coordinate of Q; both x_Q and y_Q are elements in $F(q)$. For the purpose of specifying mechanisms in this part of ISO/IEC 11770, the function $GE2OS_X(Q)$ converts the x-coordinate of Q to an octet string and ignores the y-coordinate of Q. The output of $GE2OS_X(Q)$ is computed as follows:

1. Represent x_Q as a field element.
2. Convert the result of 1) into an octet string using FE2OS.
3. Output the result of 2).

NOTE 1 – This conversion does not define a 1-1 mapping. For example, this conversion will associate the elliptic curve points Q and $-Q$ with the same octet string.

3.8 I2BS

This section gives the seven existing specifications of the function I2BS (Integer to Bit String conversion). Note that the definitions given in ISO/IEC 9796-2 and ISO/IEC 10118-4 do not use the name I2BS.

3.8.1 ISO/IEC 9796-2 [9796-2]

To represent a non-negative integer x as a bit string of length l (l has to be such that $2^l > x$), the integer shall be written in its unique binary representation:

$$x = 2^{l-1}x_{l-1} + 2^{l-2}x_{l-2} + \ldots + 2x_1 + x_0$$

where $0 \le x_i < 2$ (note that one or more leading digits will be zero if $x < 2^{l-1}$). The bit string shall be

$$x_{l-1}\, x_{l-2} \ldots x_0.$$

3.8.2 ISO/IEC 9796-3 [9796-3]

The function I2BS(m,l) takes as input two non-negative integers m and l, and outputs the unique bit string x of length l such that BS2I(x) = m, if such an x exists. Otherwise, the function fails.

3.8.3 ISO/IEC 10118-4 [10118-4]

Converting a number to a string

During computation of the round-function, integers need to be converted to strings of L bits. Where this is required, the string of bits shall be made equal to the binary representation of the integer, with the left-most bit of the string corresponding to the most significant bit of the binary representation.

If the resulting string of bits has less than L bits, then the string shall be left-padded with the appropriate number of zeros to make it of length L.

3.8.4 ISO/IEC 14888-3 [14888-3]

Input values

— g – a positive integer, indicating the length of the output string.

— x – an integer in $\{0, 1, ..., 2^g - 1\}$.

Output value

— I2BS(g, x) = $x_{g-1}\, x_{g-2} \ldots x_0$,

a binary string whose g components (representing a base 2 expansion of x, padded with leading zeros – if necessary – to attain the desired length) can be computed as follows:

$a := x$;

i := 0;

While ($i < g$) do {

$b := \lfloor a/2 \rfloor$;

$x_i := a - (2)(b)$;

$a := b$;

$i := i + 1$ }

3.8.5 ISO/IEC 15946-1 [15946-1]

The function I2BS(m,l) takes as input two non-negative integers m and l, and outputs the unique bit string x of length l such that BS2I(x) = m, if such an x exists. Otherwise, the function outputs an error message.

3.8.6 ISO/IEC 18033-2 [18033-2]

The function I2BS(m,l) takes as input two non-negative integers m and l, and outputs the unique bit string x of length l such that BS2I(x) = m, if such an x exists. Otherwise, the function **fails**.

3.8.7 ISO/IEC 18370-2 [18370-2]

The function I2BS(m, l) takes as input two non-negative integers m and l, and outputs the unique bit string x of length l such that BS2I(x) = m, if such an x exists. Otherwise, the function outputs an error message.

3.9 I2EC

This section gives the single existing specification of the function I2EC (Integer to Elliptic Curve point conversion).

3.9.1 ISO/IEC 15946-1 [15946-1]

Let E be an elliptic curve over an explicitly given finite field F. Primitive I2EC to convert from integers to elliptic curve points is defined as follows.

a) The function I2EC(x) takes as input an integer x.
b) Convert the integer x to an octet string X = I2OS(x,L(|F|−1)).

c) If there exists a point *P* on the curve *E* such that $EC2OS_E(P, \text{compressed}) = 03||X$, then the function outputs P, and otherwise, the function fails.

NOTE 1 The output of point *P*, if it exists, is uniquely defined.

NOTE 2 The function I2EC will fail on input *x* if there does not exist a point *P* on the curve *E* such that $EC2OS_E(P,\text{compressed}) = 03||X$.

NOTE 3 The range of the I2EC is approximately half of *E*(*F*). That is, the I2EC always outputs elliptic curve points *P* = (*x*,*y*) with compressed form (*x*, 1). It will not output either the point at infinity or an elliptic curve point *P* = (*x*,*y*) with compressed form (*x*, 0).

NOTE 4 Some applications based on elliptic curve may need a function which maps octet strings to elliptic curve points. The function I2EC is used as a component together with OS2I or a hash function.

3.10 I2FE

This section gives the two existing specifications of the function I2FE (Integer to Field Element conversion).

3.10.1 ISO/IEC 11770-4 [11770-4]

Function I2FE takes a non-negative integer *x* as input and produces an element $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ as output. It is defined as follows:

1. Write *x* in its unique *m*-digit base *s* representation:

$$x = x_{m-1}\, s^{m-1} + x_{m-2}\, s^{m-2} + \ldots + x_1\, s + x_0,$$

where $0 \le x_i < s$ (note that one or more leading digits will be zero if $x < s^{m-1}$).

2. Let β_i have the value x_i for $0 \le i \le m-1$.

3. Output $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$.

3.10.2 ISO/IEC 14888-3 [14888-3]

Input values

— *r* – a prime or a power of a prime.

— *x* – an integer in {0, 1, ..., *r* – 1}.

Assumptions

— When *r* = *p*, where *p* is an odd prime:

Elements of *GF*(*p*) are represented as integers in {0, 1, . . . , *p* – 1}.

— When $r = p^m$, where *p* is an odd prime and *m* is an integer greater than 1:

Elements of $GF(p^m)$ are represented as a *p*-ary strings of length *m*.

— When *r* = 2:

Elements of *GF*(2) are represented integers in {0, 1}.

— When $r = 2^m$, where *m* is an integer greater than 1:

Elements of $GF(2^m)$ are represented as a binary strings of length *m*.

Output value

— When *r* = *p* is an odd prime:

I2FE(r, x) = $x \in \{0, 1, \ldots, p-1\}$.

— When $r = p^m$, where p is an odd prime and m is an integer greater than 1:

I2FE(r, x) = $x_{m-1}\, x_{m-2} \ldots x_0$,
a p-ary string whose m components (representing a base p expansion of x, padded with leading zeros – if necessary – to attain the desired length) can be computed as follows:

$a := x$;

i := 0;

While ($i < m$) do {

$b := \lfloor a/p \rfloor$;

$x_i := a - (p)(b)$;

$a := b$;

$i := i + 1$ }

— When $r = 2$:

I2FE(r, x) = $x \in \{0, 1\}$.

— When $r = 2^m$, where m is an integer greater than 1:

I2FE(r, x) = $x_{m-1}\, x_{m-2} \ldots x_0$,
a binary string whose m components (representing a base 2 expansion of x, padded with leading zeros – if necessary – to attain the desired length) can be computed as follows:

$a := x$;

i := 0;

While ($i < m$) do {

$b := \lfloor a/2 \rfloor$;

$x_i := a - (2)(b)$;

$a := b$;

$i := i + 1$ }

3.11 I2OS

This section gives the five existing specifications of the function I2OS (Integer to Octet String conversion).

3.11.1 ISO/IEC 9796-3 [9796-3]

The function I2OS(m,l) takes as input two non-negative integers m and l, and outputs the unique octet string x of length l such that OS2I(x) = m, if such an x exists. Otherwise, the function fails.

3.11.2 ISO/IEC 11770-4 [11770-4]

Function I2OS takes as input a non-negative integer x, and produces the unique octet string $M_{l-1}M_{l-2} \ldots M_0$ of length l as output, where $l = \lceil \log_{256}(x+1) \rceil$ is the length in octets of x. I2OS is defined as follows:

1. Write x in its unique l-digit base 256 representation:

$$x = x_{l-1}\, 256^{l-1} + x_{l-2}\, 256^{l-2} + \ldots + x_1\, 256 + x_0,$$

where $0 \le x_i < 256$.

2. Let the octet M_i have the value x_i for $0 \le i \le l-1$.

3. Output the octet string $M_{l-1}\, M_{l-2} \ldots M_0$.

For example, I2OS(10945) = 2A C1.

3.11.3 ISO/IEC 14888-3 [14888-3]

I2OS(*h*, *x*):

Input values

— *h* – a positive integer, indicating the length of the output octet string.

— *x* – an integer in $\{0, 1, ..., 256^h - 1\}$.

Output value

— Compute a string of integers, $x_{h-1}\, x_{h-2} \ldots x_0$, where $x_i \in \{0, 1, \ldots, 255\}$ for $0 \le i < h$, representing a base 256 expansion of *x*, padded with leading zeros – if necessary – to attain length *h*. The x_i values can be computed as follows:

$a := x;$

i := 0;

While ($i < h$) do {

$b := \lfloor a/256 \rfloor;$

$x_i := a - (256)(b);$

$a := b;$

$i := i + 1$ }

— I2OS(*h*, *x*) = $M_{h-1}\, M_{h-2} \ldots M_0$,

where octet M_i is equivalent to the 8-long binary string I2BS(8, x_i).

3.11.4 ISO/IEC 15946-1 [15946-1]

The function I2OS(*m*,*l*) takes as input two non-negative integers *m* and *l*, and outputs the unique octet string *x* of length *l* in octets such that OS2I(*x*) = *m*, if such an *x* exists. Otherwise, the function outputs an error message.

3.11.5 ISO/IEC 18033-2 [18033-2]

The function I2OS(*m*,*l*) takes as input two non-negative integers *m* and *l*, and outputs the unique octet string *x* of length *l* such that OS2I(*x*) = *m*, if such an *x* exists. Otherwise, the function **fails**.

3.12 I2P

This section gives the two existing specifications of the function I2P (Integer to Point conversion).

3.12.1 ISO/IEC 14888-3 [14888-3]

NOTE This function is believed to hold the properties of the randomness and onewayness, i.e., converting an integer to a point such that the point is randomly distributed in the selected group and that given current knowledge, recovering the integer from the point is computationally infeasible. This function is also used by IEEE P1363. However, there is no formal security proof of this function published. For this reason, this function is recommended as informative.

Given a set of elliptic curve domain parameters (r, q, a_1, a_2), Function I2P operates on an integer u as input, and produces a point T of order q on the curve E over $GF(r)$ as output, which is specified as T = I2P(u). In the following specification, the operations of addition and multiplication between finite field elements follow the specification in ISO/IEC 15946-1, and the operation of KDF1 follows the specification in ISO/IEC 18033-2.

1. Set v = BS2I(8 $\lceil \log_{256}(r) \rceil$, $KDF1_{H_2}$ (I2OS($\lceil \log_{256}(u) \rceil$, u), length in bytes of representation of r)) mod r. If v = 0, output "invalid" and stop.

2. Set $\lambda = u \bmod 2$.

3. If r is a prime ($r = p$) and the curve E is $Y^2 = X^3 + a_1X + a_2$ defined over $GF(p)$, compute the point T in the following way:

   a. Let x have the value of v.

   b. Compute the field element $c = x^3 + a_1x + a_2 \bmod p$. If $c = 0$, output "invalid" and stop.

   c. Find a square root d of c modulo p (i.e., an integer d with $0 < d < p$ such that $d^2 = c \bmod p$) or determine that no such square roots exist.

      i. To determine the existence of the square root, compute $\delta = c^{(p-1)/2} \bmod p$. If $\delta = 1$, d exists, otherwise d does not exist.

      ii. If $\delta \neq 1$, compute $u = u + 1$ and go to Step a.

      iii. If $\delta = 1$, find d.

      NOTE The operation of finding two field elements d such that $d^2 = c \bmod p$ is given in [4] and [23]. In order to make the result unique, if the application of this mechanism has a specific requirement on which value should be chosen, follow the requirement. Otherwise, a smallest absolute value modulo p is recommended.

      iv. Set $y = (\text{I2FE}(r, p-1))^{\lambda} \times d$.

      v. Set point $T = (x, y)$, compute $T = [\#E/q]T$, and output T.

4. If r is an odd prime power ($r = p^m$, $p > 2$, $m \geq 2$) and the curve E is $Y^2 = x^3 + a_1x^2 + a_2$ (when $p = 3$) and $Y^2 = x^3 + a_1x + a_2$ (when $p > 3$) defined over $GF(p^m)$, compute the point T in the following way:

   a. Set x = I2FE(r, v).

   b. If ($p = 3$), set $c = x^3 + a_1x^2 + a_2$ in $GF(p^m)$. If $c = 0$, output "invalid" and stop.

   c. If ($p > 3$), set $c = x^3 + a_1x + a_2$ in $GF(p^m)$. If $c = 0$, output "invalid" and stop.

   d. Find a square root d of c in $GF(p^m)$ (i.e., a $GF(p^m)$ element d such that $d^2 = c$ in $GF(p^m)$) or determine that no such square roots exist. If the result is no such square roots existing, Set $u = u + 1$ and go to Step a.

      NOTE The operations of determining the existence of and finding a square root of a field element are given in [4] and [23]. In order to make the result unique, if the application of this mechanism has a specific requirement on which value should be chosen, follow the requirement. Otherwise, compare the maximum degrees of the results, and select d with smaller degree. If the maximum degree of the values are same, then

select d with smaller absolute value coefficient of the degree. If both the degree and the coefficient are same, then compare the second largest degree and select d with smaller absolute value coefficient of the degree. Repeat this process until the unique d is selected.

e. Set $y = (\text{I2FE}(r, p-1))^{\lambda} \times d$.

f. Set point $T = (x, y)$, compute $T = [\#E/q]T$, and output T.

5. If r is a prime power of 2 ($r = 2^m$, $m \geq 2$) and the curve E is $Y^2 + XY = X^3 + a_1X^2 + a_2$ defined over $GF(2^m)$, compute the point T in the following way:

a. Set $x = \text{I2FE}(r, v)$.

b. Set $c = x + a_1 + a_2x^{(-2)}$ in $GF(2^m)$. If $c = 0$, output "invalid" and stop.

c. Find a field element d satisfying $d^2 + d \equiv c$ in $GF(2^m)$ or determine that no such integers exist. If the result is no such integers existing, Set $u = u + 1$ and go to Step a.

NOTE The operations of determining the existence of and finding a field element d such that $d^2 + d = c$ in $GF(2^m)$ is given in [4] and [23]. In order to make the result unique, if the application of this mechanism has a specific requirement on which value should be chosen, follow the requirement. Otherwise, compare the maximum degrees of the results, and select d with smaller maximum degree. If the maximum degree of two values are same, then compare the second largest degree and select d with smaller second largest degree. Repeat this process until the unique d is selected.

d. Set $y = (d + \text{I2FE}(r, \lambda)) \times x$.

e. Set point $T = (x, y)$, compute $T=[\#E/q]T$, and output T.

3.12.2 ISO/IEC 11770-4 [11770-4]

Given a set of EC domain parameters (E, q, p, m, r, k, a_1, a_2), Function I2P operates on an integer u as input, and produces a point T on the curve E over $F(q)$ as output, which is specified as $T = \text{I2P}(u)$. In the following specification, the operations of addition and multiplication between finite field elements follow the specification in ISO/IEC 15946-1.

1. Set $v = \text{BS2I}(H(\text{I2OS}(u)))$ mod q.

— If $v = 0$, output "invalid" and stop.

2. Set $\lambda = u$ mod 2.

3. If q is prime ($q = p$) and the curve E is $Y^2 = X^3 + a_1X + a_2$ defined over $F(p)$, compute the point T in the following way:

(a) Set $x = v$.

(b) Compute the field element $\alpha = x^3 + a_1x + a_2$ mod p.

— If $\alpha = 0$, output "invalid" and stop.

(c) Find a square root β of α modulo p (i.e., an integer β with $0 < \beta < p$ such that $\beta^2 = \alpha$ mod p) or determine that no such square roots exist.

— To determine the existence of the square root, compute $\delta = \alpha^{(p-1)/2}$ mod p. If $\delta = 1$, β exists, otherwise β does not exist.

— If $\delta \neq 1$, compute $u = u + 1 \bmod p$ and go to Step 1.

— If $\delta = 1$, find β.

NOTE – The operation of finding a field element β such that $\beta^2 = \alpha \bmod p$ is given in [ANSI X9.62] and [IEEE P1363].

(d) Set $y = (p - 1)^{\lambda} \times \beta$.

(e) Set point $T = (x, y)$ and output T.

4. If q is even ($q = 2^m$) and the curve E is $Y^2 + XY = X^3 + a_1X^2 + a_2$ defined over $F(2^m)$, compute the point T in the following way:

(a) Set x = I2FE(v).

(b) Set $\alpha = x + a_1 + a_2x^{(-2)}$ in $F(2^m)$.

(c) Find a field element β satisfying $\beta^2 + \beta \equiv \alpha$ in $F(2^m)$ or determine that no such integers exist. If the result is no such integers existing, Set $u = u + 1 \bmod q$ and go to Step 1.

NOTE – The operations of determining the existence of and finding a field element β such that $\beta^2 + \beta = \alpha$ in $F(2^m)$ is given in [ANSI X9.62] and [IEEE P1363].

EDITOR'S NOTE – The word 'integers' in the text of 4(c) is incorrect – it should refer to 'field elements'.

(d) Set $y = (\beta +$ I2FE(λ)$) \times x$.

(e) Set point $T = (x, y)$ and output T.

3.13 IHF1

This section gives the single existing specification of the function IHF1 (Bit string to integer in a specific range conversion function).

3.13.1 ISO/IEC 18033-5 [18033-5]

IHF1 is based on four hash-functions specified in ISO/IEC 10118-3, namely SHA-224, SHA-256, SHA-384 and SHA-512. It inputs a string of bits and outputs an integer in a specified range.

Input:

— A string $str \in \{0,1\}^*$

— A security parameter $\kappa \in \{112,128,192,256\}$

— An integer n, $0 < n < 2^{4\kappa}$

Output:

— An integer v, $0 \leq v < n$.

Operation: Perform the following steps.

(a) If $\kappa = 112$ then let H be SHA-224;

else if $\kappa = 128$ then let H be SHA-256;

else if $\kappa = 192$ then let H be SHA-384;
else if $\kappa = 256$ then let H be SHA-512.

(b) Let h_0 be an all-zero bit string of length 2κ.

(c) Let $t_1 = h_0 \,||\, str$.

(d) Let $h_1 = H(t_1)$.

(e) Let v_1 = BS2IP(h_1).

(f) Let $t_2 = h_1 \,||\, str$.

(g) Let $h_2 = H(t_2)$.

(h) Let a_2 = BS2IP(h_2).

(i) Let $v_2 = 2^{2\kappa} v_1 + a_2$.

(j) Output v_2 mod n.

3.14 OS2BS

This section gives the three existing specifications of the function OS2BS (Octet String to Bit String conversion function).

3.14.1 ISO/IEC 9796-3 [9796-3]

The function OS2BS(*x*) takes as input an octet string *x* and outputs *x*, which is also a bit string.

3.14.2 ISO/IEC 15946-1 [15946-1]

The function OS2BS(*x*) takes as input an octet string *x*, interprets it as a bit string *y* (in the natural way) and outputs the bit string *y*.

3.14.3 ISO/IEC 18033-2 [18033-2]

The function OS2BS(*x*) takes as input an octet string $x = \langle x_1,\ldots,x_l \rangle$ and outputs the bit string $y = x_1 \,||\, \ldots \,||\, x_l$.

3.15 OS2EC

This section gives the three existing specifications of the function OS2EC (Octet String to Elliptic Curve point conversion function).

3.15.1 ISO/IEC 9796-3 [9796-3]

The function $OS2EC_E$(EP) takes as input an octet string EP. If there exists a point *P* on the curve *E* and a format specifier fmt such that $EC2OS_E(P, \text{fmt}) = \text{EP}$, then the function outputs *P* (in uncompressed form), and otherwise, the function fails. Note that the point *P*, if it exists, is uniquely defined, and so the function $OS2EC_E$(EP) is well defined.

3.15.2 ISO/IEC 15946-1 [15946-1]

The function $OS2EC_E$(EP) takes as input an octet string EP. If there exists a point P on the curve E and a format specifier fmt such that $EC2OS_E(P, fmt) = EP$, then the function outputs P (in uncompressed form), and otherwise, the function fails. Note that the point P, if it exists, is uniquely defined, and so the function $OS2EC_E$(EP) is well defined.

3.15.3 ISO/IEC 18033-2 [18033-2]

The function $OS2EC_E$(EP) takes as input an octet string EP. If there exists a point P on the curve E and a format specifier fmt such that $EC2OS_E(P, fmt) = EP$, then the function outputs P (in uncompressed form), and otherwise, the function fails. Note that the point P, if it exists, is uniquely defined, and so the function $OS2EC_E$(EP) is well defined.

3.16 OS2FE

This section gives the three existing specifications of the function OS2FE (Octet String to Field Element conversion function).

3.16.1 ISO/IEC 9796-3 [9796-3]

The function $OS2FE_F(x)$ takes as input an octet string x, and outputs the (unique) field element $a \in F$ such that $FE2OS_F(a) = x$, if such an a exists, and otherwise fails.

3.16.2 ISO/IEC 15946-1 [15946-1]

The function $OS2FE_F(x)$ takes as input an octet string x, and outputs the (unique) field element $a \in F$ such that $FE2OS_F(a) = x$, if such an a exists, and otherwise fails.

NOTE $OS2FE_F(x)$ fails if and only if either x does not have length exactly $\lceil \log_{256} |F| \rceil$, or $OS2I(x) \geq |F|$.

3.16.3 ISO/IEC 18033-2 [18033-2]

The function $OS2FE_F(x)$ takes as input an octet string x, and outputs the (unique) field element $a \in F$ such that $FE2OS_F(a) = x$, if any such a exists, and otherwise **fails**. Note that $OS2FE_F(x)$ **fails** if and only if either x does not have length exactly $\lceil \log_{256} |F| \rceil$, or $OS2I(x) \geq |F|$.

3.17 OS2I

This section gives the five existing specifications of the function OS2I (Octet String to Integer conversion function).

3.17.1 ISO/IEC 9796-3 [9796-3]

The function $OS2I(x)$ takes as input an octet string, and outputs the integer $BS2I(OS2BS(x))$.

3.17.2 ISO/IEC 11770-4 [11770-4]

Function OS2I takes an octet string $M_{l-1}\, M_{l-2} \ldots M_0$ as input and produces a non-negative integer y as output. It is defined as follows:

1. Let integer y_i have the value of the octet M_i for $0 \leq i \leq l-1$.
2. Compute the integer $y = y_{l-1}\, 256^{l-1} + y_{l-2}\, 256^{l-2} + \ldots + y_1\, 256 + y_0$.
3. Output y.

For example, OS2I(2A C1) = 10945.

Note that the octet string of length zero (the empty octet string) is converted to the integer 0 and vice versa.

3.17.3 ISO/IEC 14888-3 [14888-3]

OS2I(*h*, *M*):

Input values

— h – a positive integer, indicating the length of the input octet string.

— $M = M_{h-1} M_{h-2} \ldots M_0$ – an octet string of length h.

Assumption

— For $0 \le i < h$, M_i is represented as an 8-long binary string.

Output value

— Compute the string of integers, $x_{h-1} x_{h-2} \ldots x_0$,

where x_i = BS2I(8, M_i) $\in \{0, 1, \ldots, 255\}$ for $0 \le i < h$.

OS2I(h, x) = $256^{h-1} x_{h-1} + 256^{h-2} x_{h-2} + \ldots + x_0 \in \{0, 1, \ldots, 256^h - 1\}$.

3.17.4 ISO/IEC 15946-1 [15946-1]

The function OS2IP(x) takes as input an octet string x, and outputs the integer BS2I(OS2BS(x)).

3.17.5 ISO/IEC 18033-2 [18033-2]

The function OS2I(x) takes as input an octet string, and outputs the integer BS2I(OS2BS(x)).

3.18 PHF1

This section gives the single existing specification of the function PHF1 (Bit string to supersingular elliptic curve group element conversion function).

3.18.1 ISO/IEC 18033-5 [18033-5]

The function PHF1 returns an element of an elliptic curve group $E(GF(q))[p]$ for a supersingular elliptic curve $E/GF(q): y^2 = x^3 + b$ or $E/GF(q): y^2 = x^3 + ax$.

Input:

— A string $str \in \{0,1\}^*$

— A security parameter $\kappa \in \{112,128,192,256\}$

— A flag j taking the values 0 or 1 which defines a supersingular elliptic curve, with $j = 0$ representing the elliptic curve $E/GF(q): y^2 = x^3 + b$ and $j = 1$ representing the elliptic curve $E/GF(q): y^2 = x^3 + ax$.

— A prime q with $q = 2 (\mathrm{mod}\, 3)$ when $j = 0$ or $q = 3 (\mathrm{mod}\, 4)$ when $j = 1$ that defines the finite field $GF(q)$.

— An integer $a, 0 < a < q$ if $j = 1$ or an integer $b, 0 < b < q$ if $j = 0$

— A prime p with $p \mid \#E(GF(q))$ and $p^2 \nmid \#E(GF(q))$ for elliptic curve E defined by the flag j

Output:

— An element of $E(GF(q))[p]$ for the selected elliptic curve.

Operation: Use the following steps.

(a) Let $r = (q+1)/p$.

(b) If *j=0* then perform the following steps:

(1) Let $y = \mathrm{IHF1}(str, q, \kappa)$.

(2) Let $x = (y^2 - b)^{(2q-1)/3} (\mathrm{mod} q)$.

(3) Let $Q = (x, y)$.

(c) Else if *j=1* perform the following steps:

(1) Let $x = \mathrm{IHF1}(str, q, \kappa)$.

(2) Let $z = x^3 + ax (\mathrm{mod} q)$.

(3) If the Jacobi symbol $(z/q) = +1$ then perform the following steps:

(i) Let $y = z^{(q+1)/4} (\mathrm{mod} q)$.

(ii) Let $Q = (x, y)$.

(4) If the Jacobi symbol $(z/q) = -1$ then perform the following steps:

(i) Let $y = (-z)^{(q+1)/4} (\mathrm{mod} q)$.

(ii) Let $Q = (-x, y)$.

(d) Return rQ.

4 Recommended conversion functions

4.1 General

This part of SD7 provides recommended conversion functions. For 15 of the 18 types of conversion function specified in section 3, together with a further type (BS2FE) added for completeness, a single recommended function is provided. These functions are recommended for adoption in all SC 27/WG 2 standards. No recommended function is provided in the other three cases, for the reasons explained in the relevant sections.

Names of functions are given in a non-italic font, and this form is recommended for use in all new and revised standards.

4.2 Symbols and terminology

The following symbols and terminology are used in defining and describing the properties of the recommended conversion functions.

4.2.1 Terminology

string ordered sequence of symbols (e.g. bits, numerals or octets)

4.2.2 Symbols

0^n	string of zeros of length n, for a non-negative integer n
$\log_2 x$	logarithm to the base 2 of the non-zero value x
a mod b	for an integer a and a positive integer b, $c = a$ mod b is the smallest non-negative integer such that a-c is a multiple of b
truncate(y)	for a bit string y, truncate(y) is the unique bit string obtained from y by deleting all the leftmost (most significant) zeros from y, so that truncate(y) has leftmost (most significant) bit equal to 1 or is the empty string
$a\|\|b$	for strings a and b, $a\|\|b$ is the string made up of the symbols in a (in the order defined in a) followed by the symbols in b (in the order defined in b)
$\|s\|$	length of a string s, i.e. the number of symbols in s, where $\|s\|=0$ if s is empty
$\lceil x \rceil$	for a real number x, $\lceil x \rceil$ represents the unique integer c such that $c \geq x > c$-1
$\lfloor x \rfloor$	for a real number x, $\lfloor x \rfloor$ represents the unique integer c such that $c \leq x < c$+1
$[a,b]$	for integers a, b such that $a \leq b$, the set of all integers c satisfying $a \leq c \leq b$
$<b_7 b_6 \ldots b_0>$	an octet made up of the eight bits $b_7 b_6 \ldots b_0$, where b_7 is the most significant (leftmost) bit, and b_0 is the least significant (rightmost) bit

4.2.3 Conventions

octets	octets are written either as strings of eight bits, e.g. <0010 1110>, where the leftmost (first) bit is the most significant, or for convenience as a pair of hexadecimal digits (e.g. 2E).

4.3 Conversion functions

A single recommended specification for each of the 18 types of conversion function contained in existing SC 27/WG 2 standards, together with an additional function not currently specified, is now given. Each function is presented in the following format:

Purpose

The objective of the conversion function is briefly specified.

Input(s)

The input(s) to the function are listed.

Output

The output from the function is given.

Operation

The means by which the output is computed from the input(s) is specified.

Example

Where appropriate, one or more examples of the function are given.

Consistency

The consistency of the proposed function with the existing functions specified in the relevant part of section 3 is described. If an inconsistency exists, a recommendation is given as to what changes are required in order to replace the current conversion function with the recommended version, except where such changes are obvious.

Dependencies

Functions used in the definition of the function, and which therefore must also be defined in any standard adopting the conversion function.

Relationship to other conversion functions

The relationships of the conversion function to other functions in the set are described (where appropriate).

4.3.1 BS2FE (bit string to field element conversion)

Purpose

This function takes as input a bit string and outputs an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1].

Input

- y a bit string

Output

- s = BS2FE(y) an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1]

Operation

x = BS2I(y);

if $x \geq r$ then terminate and output an error message;

s = I2FE(x)

Consistency

This function has not previously been defined.

Dependencies

- BS2I (see section 4.3.2)
- I2FE (see section 4.3.11)

Relationship to other conversion functions

BS2FE has an inverse relationship to FE2BS, i.e.

- BS2FE(FE2BS(s)) = s for all field elements s; and
- for any bit string y, either BS2FE(y) terminates with an error or FE2BS(BS2FE(y)) = y.

4.3.2 BS2I (bit string to integer conversion)

Purpose

This function takes as input a bit string and outputs the integer for which this bit string is the binary representation.

Input

- *y* a bit string

Output

- *x* = BS2I(*y*) a non-negative integer

Operation

Let $g = |y|$;

If $g = 0$ then $x = 0$ else

{

Let $y = b_{g-1}b_{g-2} \ldots b_0$; [where b_i is a bit, $0 \le i \le g-1$]

$$x = \sum_{i=0}^{g-1} b_i 2^i$$

}

Example

BS2I(000 0010 1010 1100 0001) = 10945

Consistency

The function as defined above is consistent with the definitions in section 3.1, with the following minor difference.

- The function defined in ISO/IEC 14888-3 (see section 3.1.5) has two inputs – the bit string itself and its length. However, in all other respects the definition is consistent with the definition given here.

  If the recommended function BS2I is adopted in this document, the input length value *g* should be removed.

Dependencies

None.

Relationship to other conversion functions

BS2I has an inverse relationship to I2BS, i.e.:

- BS2I(I2BS(*x*,*g*)) = *x* for all non-negative integers *x* and all integers *g* satisfying $g \ge \log_2(x+1)$; and
- I2BS(BS2I(*y*),|*y*|) = *y* for all bit strings *y*.

4.3.3 BS2OS (bit string to octet string conversion)

Purpose

This function takes as input a bit string *y* and outputs an equivalent octet string of length $\lceil |y|/8 \rceil$.

Input

- *y* a bit string

Output

- *z* = BS2OS(*y*) an octet string

Operation

g = |*y*|;

if (*g* mod 8) > 0 then $y = 0^{8-(g \bmod 8)} || y$;

$h = \lceil g/8 \rceil$;

If $y = b_{8h-1}b_{8h-2} \ldots b_0$, then let $z = o_{h-1}o_{h-2} \ldots o_0$ be the octet string of length *h* such that $o_i = <b_{8i+7}b_{8i+6} \ldots b_{8i}>$, $0 \le i \le h-1$

Example

BS2OS(000 0010 1010 1100 0001) = <0000 0000> <0010 1010> <1100 0001> = 00 2A C1

Consistency

The above specification is believed to be consistent with the functions defined in section 3.2.

Dependencies

None.

Relationship to other conversion functions

BS2OS has an inverse relationship to OS2BS, i.e.:

- BS2OS(OS2BS(*z*)) = *z* for all octet strings *z*; and
- truncate(OS2BS(BS2OS(*y*))) = truncate(*y*) for all bit strings *y*.

4.3.4 EC2OS (elliptic curve point to octet string conversion)

Purpose

This function takes as inputs a point *P* on an elliptic curve and a format specifier `fmt`, and outputs an octet string representing the point.

Inputs

- *P* a point on an elliptic curve *E* over a field *F* of characteristic *p*
- `fmt` a format specifier which must be one of `compressed`, `uncompressed`, or `hybrid`

Output

- *EP* = EC2OS(*P*, `fmt`) an octet string

Assumptions

As described in ISO/IEC 15946-1, a point *P* on an elliptic curve *E* is either 0_E or a pair (*x*, *y*), where *x*, *y* ∈ *F*. If *P* = (*x*, *y*) then it can be represented in one of three forms:

- *uncompressed*: in which case the representation is simply the pair (*x*, *y*);
- *compressed*: in which case the representation is the pair (*x*, $\tilde{y}$), where $\tilde{y} \in \{0,1\}$, and the relationship between $\tilde{y}$ and (*x*, *y*) is as defined in ISO/IEC 15946-1;
- *hybrid*: in which case the representation is the triple (*x*, $\tilde{y}$, *y*), where $\tilde{y}$ is as above.

Operation

If $P = 0_E$, then *EP* = I2OS(0,1);

If $P = (x, y) \neq 0_E$, with compressed form (*x*, $\tilde{y}$), then *EP* = *H* || *X* || *Y*, where

- *H* is a single octet of the form I2OS(4*U* + (2 + $\tilde{y}$)*C*,1), where
  - *U* = 1 if `fmt` is either `uncompressed` or `hybrid`; otherwise, *U* = 0; and
  - *C* = 1 if `fmt` is either `compressed` or `hybrid`; otherwise, *C* = 0;
- *X* is the octet string FE2OS(*x*); and
- *Y* is the octet string FE2OS(*y*) if `fmt` is either `uncompressed` or `hybrid`; otherwise *Y* is the null octet string.

Consistency

The function as defined above is consistent with the definitions in section 3.3.

Dependencies

- FE2OS (see section 4.3.7)
- I2OS (see section 4.3.12)

Relationship to other conversion functions

EC2OS has an inverse relationship to OS2EC, i.e.:

- OS2EC(EC2OS(*P*, `fmt`)) = *P*, for every point *P* on an elliptic curve.

4.3.5 FE2BS (field element to bit string conversion)

Purpose

This function takes as input an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1], and outputs a bit string.

Input

- s an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1]

Output

- y = FE2BS(s) a bit string

Operation

Let $g = 8\lceil \log_{256}(r) \rceil$

y = I2BS(FE2I(s), g)

Consistency

The function as defined above is consistent with the definitions in section 3.4, with the following minor difference.

- The function defined in ISO/IEC 14888-3 (see section 3.4.1) has two inputs – the field element itself and the number of elements in the finite field (r). However, in all other respects the definition is consistent with the definition given here.

  If the recommended function FE2BS is adopted in this document, the input field size r should be removed.

Dependencies

- FE2I (see section 4.3.6)
- I2BS (see section 4.3.9)

Relationship to other conversion functions

FE2BS has an inverse relationship to BS2FE, i.e.

- BS2FE(FE2BS(s)) = s for all field elements s; and
- for any bit string y, either BS2FE(y) terminates with an error, or FE2BS(BS2FE(y)) = y.

4.3.6 FE2I (field element to integer conversion)

Purpose

This conversion function takes as input an element of a finite field of p^m elements (for a prime p), and outputs a non-negative integer in the range [0, p^m-1].

Input

- s an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1]

Output

- x = FE2I(s) an integer in the range [0, r-1]

Operation

Let $s = a_{m-1}a_{m-2} \ldots a_0$, where $0 \le a_i \le p-1$, $0 \le i \le m-1$

$$x = \sum_{i=0}^{m-1} a_i p^i$$

Consistency

This function is believed to be largely consistent with those described in section 3.5, with the following minor differences.

- The functions defined in ISO/IEC 9796-3 (see section 3.5.1) and ISO/IEC 15946-1 (see section 3.5.5) work the opposite way round, that is they both compute x as:
$$x = \sum_{i=0}^{m-1} a_{m-1-i} p^i$$
- The function defined in 14888-3 (see section 3.5.3) has two inputs – the field element itself and the number of elements in the finite field (r). However, in all other respects the definition is consistent with the definition given here.

  If the recommended function FE2I is adopted in this document, the input length value r should be removed.

Dependencies

None.

Relationship to other conversion functions

FE2I is the inverse function to I2FE, i.e.

- I2FE(FE2I(s)) = s, for every element s of the finite field of r elements;
- FE2I(I2FE(x)), for every integer x in the range [0, r-1].

4.3.7 FE2OS (field element to octet string conversion)

Purpose

This function takes as input an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1], and outputs an octet string.

Input

- s an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1]

Output

- z = FE2OS(s) an octet string

Operation

Let $h = \lceil \log_{256}(r) \rceil$

z = I2OS(FE2I(s), h)

Consistency

The function as defined above is consistent with the definitions in section 3.6, with the following minor difference.

- The function defined in ISO/IEC 9796-3 (see section 3.6.1) has two inputs – the field element itself and a parameter relating to the number of elements in the finite field. However, in all other respects the definition is consistent with the definition given here.

  If the recommended function FE2OS is adopted in this document, the input parameter relating to the field size should be removed.

Dependencies

- FE2I (see section 4.3.6)
- I2OS (see section 4.3.12)

Relationship to other conversion functions

FE2OS has an inverse relationship to OS2FE, i.e.

- OS2FE(FE2OS(s)) = s for all field elements s; and
- for any octet string z, either OS2FE(z) terminates with an error, or FE2OS(OS2FE(z)) = z.

4.3.8 GE2OS$_x$ (group element to octet string conversion)

Purpose

This function takes as input a group element *GE* and a setting index *SI*, where if *SI* indicates the Discrete Logarithm (DL) setting *GE* is an element of a finite field and if *SI* indicates the Elliptic Curve (EC) setting *GE* is an elliptic curve point, and outputs an octet string.

Input

- *SI* {DL, EC}
- *GE*
  - If *SI* = *DL*, *GE* is an element of a finite field of $r = p^m$ elements, for a prime *p*, represented as a string of *m* integers in the range [0, *p*-1]
  - If *SI* = *EC*, *GE* is a point on an elliptic curve, which is either 0_E or a pair (*x*, *y*), where *x*, *y* are two elements of a finite field of $r = p^m$ elements, for a prime *p*, each represented as a string of *m* integers in the range [0, *p*-1]

Output

- *z* = GE2OS$_x$(*SI, GE*) an octet string

Operation

If *SI* = *DL*, *z* = FE2OS(*GE*);

Else, if *GE* = 0_E, then *z* = I2OS(0,1);

Else, if *GE* = (*x*, *y*) ≠ 0_E, then *z* = FE2OS(*x*)

Consistency

The function as defined above is consistent with the definitions in section 3.6, with the following two minor differences.

- The function defined in ISO/IEC 11770-4 (see section 3.6.1) has only one input – a group element without a setting index.
- The function defined in ISO/IEC 11770-4 (see section 3.6.1) does not explicitly deal with the case if *GE* = 0_E, then *z* = I2OS(0).

However, in all other respects the definition is consistent with the definition given here. If the recommended function GE2OS$_x$ is adopted in this document, the corresponding changes are required.

Dependencies

- FE2OS (see section 4.3.6)
- I2OS (see section 4.3.12)

Relationship to other conversion functions

GE2OS$_x$ has an inverse relationship to OS2FE when *SI* = *DL*, i.e.

- OS2FE(GE2OS$_x$(*DL, GE*)) = *GE* for all field elements *GE*; and
- for any octet string *z*, either OS2FE(*z*) terminates with an error, or GE2OS$_x$(*DL*, OS2FE(*z*)) = *z*.

When *SI* = *EC*, such an inverse relationship does not exist, and the conversion function does not define a 1-1 mapping between an elliptic curve point and an octet string, because the *y* value of the curve point is ignored in the conversion operation.

4.3.9 I2BS (integer to bit string conversion)

Purpose

This conversion function takes as input a non-negative integer and outputs a bit string corresponding to its binary representation.

Inputs

- x a non-negative integer
- g a non-negative integer [the desired length for the output bit string]

Output

- y = I2BS(x, g) a bit string of length g

Operation

If $g < \log_2(x+1)$ then terminate and output an error message

Let y equal the empty string

While $x \neq 0$

{

$y = x \bmod 2 \,||\, y$

$x = \lfloor x/2 \rfloor$

}

$y = 0^{g-|y|} \,||\, y$

Examples

I2BS(10945, 19) = 000 0010 1010 1100 0001

I2BS(10945, 14) = 10 1010 1100 0001

Consistency

The function as defined above is consistent with the definitions in section 3.8, with the following minor differences.

- In the function defined in ISO/IEC 14888-3 the order of the two inputs is reversed (see section 3.8.4).
- The functions defined in ISO/IEC 9796-2 (see sections 3.8.1) have only one input – the integer to be converted. The length of the output bit string is constrained to be as small as possible, i.e. so that $g = \lceil \log_2(x+1) \rceil$. However, in all other respects the definition is consistent with the definition given here.

  If the recommended function I2BS is adopted in these documents, the existing function I2BS(x) should be replaced with I2BS(x, $\lceil \log_2(x+1) \rceil$).

Dependencies

None.

Relationship to other conversion functions

I2BS has an inverse relationship to BS2I, i.e.:

- BS2I(I2BS(x, g)) = x for all non-negative integers x and all integers g satisfying $g \geq \log_2(x+1)$; and
- I2BS(BS2I(y), $|y|$) = y for all bit strings y.

4.3.10 I2EC (integer to elliptic curve point conversion)

Purpose

This function takes as input a non-negative integer and outputs a point on an elliptic curve to which this integer corresponds (or an error if there is no such point).

Inputs

- *x* a non-negative integer

Output

- *P* = I2EC(*x*) a point on an elliptic curve *E* over a field *F* of characteristic *p* and containing $r=p^m$ elements

Assumptions

As described in ISO/IEC 15946-1, a point *P* on an elliptic curve *E* is either 0_E or a pair (*x*, *y*), where *x*, *y* $\in$ *F*; if *P* = (*x*, *y*) then it can be represented in one of three forms:

- *uncompressed*: in which case the representation is simply the pair (*x*, *y*);
- *compressed*: in which case the representation is the pair (*x*, $\tilde{y}$), where $\tilde{y} \in \{0,1\}$, and the relationship between $\tilde{y}$ and (*x*, *y*) is as defined in ISO/IEC 15946-1;
- *hybrid*: in which case the representation is the triple (*x*, $\tilde{y}$, *y*), where $\tilde{y}$ is as above.

Operation

Let $h = \lceil \log_{256}(r) \rceil$

let *P* = OS2EC(I2OS(3,1)||I2OS(*x*,*h*))

Consistency

The function as defined above is consistent with the definition in section 3.9.

Dependencies

- OS2EC (see section 4.3.4)
- I2OS (see section 4.3.12)

4.3.11 I2FE (integer to field element conversion)

Purpose

This conversion function takes as input a non-negative integer in the range $[0,p^m-1]$ (for a prime p) and outputs an element of a finite field of p^m elements.

Input

- x an integer in the range $[0, r-1]$, where $r = p^m$

Output

- s = I2FE(x) an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range $[0, p-1]$

Operation

Let s equal the empty string

While $x \neq 0$

{

$s = x \bmod p \,||\, s$

$x = \lfloor x/p \rfloor$

}

Consistency

This function is believed to be consistent with those described in section 3.10.

Dependencies

None.

Relationship to other conversion functions

I2FE is the inverse function to FE2I, i.e.

- I2FE(FE2I(s)) = s, for every element s of the finite field of r elements;
- FE2I(I2FE(x)) = x, for every integer x in the range $[0, r-1]$.

4.3.12 I2OS (integer to octet string conversion)

Purpose

This conversion function takes as input a non-negative integer and outputs an octet string containing its binary representation.

Inputs

- *x* a non-negative integer
- *h* a non-negative integer [the desired length for the output octet string]

Output

- *z* = I2OS(*x*, *h*) an octet string of length *h*

Operation

g = 8**h*;

z = BS2OS(I2BS(*x*, *g*))

Example

I2OS(10945, 3) = <0000 0000> <0010 1010> <1100 0001> = 00 2A C1

I2OS(10945, 2) = <0010 1010> <1100 0001> = 2A C1

Consistency

The function as defined above is consistent with the definitions in section 3.11, with the following minor differences.

- In the function defined in ISO/IEC 14888-3 the order of the two inputs is reversed (see section 3.11.3)
- The functions defined in ISO/IEC 11770-4 (see sections 3.11.2) have only one input – the integer to be converted. In both cases the length of the output octet string is constrained to be as small as possible, i.e. so that $h = \lceil \log_2(x+1)/8 \rceil$. However, in all other respects the definition is consistent with the definition given here.

  If the recommended function I2OS is adopted in these documents, the existing function I2OS(*x*) should be replaced with I2OS(*x*, $\lceil \log_2(x+1)/8 \rceil$).

Dependencies

- BS2OS (see section 4.3.3)
- I2BS (see section 4.3.9)

Relationship to other conversion functions

I2OS has an inverse relationship to OS2I, i.e.:

- OS2I(I2OS(*x*, *h*)) = *x* for all non-negative integers *x* and all integers *h* satisfying h ≥ $\lceil \log_2(x+1)/8 \rceil$; and
- I2OS(OS2I(*z*), |*z*|) = *z* for all octet strings *z*.

4.3.13 I2P

The functions defined in section 3.12 appear at first sight to fulfil the same purpose as function I2EC (see section 4.3.10), i.e. they convert an integer to an elliptic curve point. However, whereas I2EC has the property that either it is invertible or it fails, the I2P functions do not have this property. The I2P functions are not strictly conversion functions and are used in completely different circumstances to I2EC. We therefore do not provide a recommended I2P function in this document.

4.3.14 IHF1

The function defined in section 3.13 appears to be very special purpose, and a function of precisely the same characteristics seems unlikely to be required in another standard. As a result we do not provide a recommended function in this document.

4.3.15 OS2BS (octet string to bit string conversion)

Purpose

This function takes as input an octet string *z* and outputs an equivalent bit string of length 8|*z*|.

Input

- *z* an octet string

Output

- *y* = OS2BS(*z*) a bit string

Operation

h = |*z*|;

g = 8**h*;

If $z = o_{h-1}o_{h-2} \ldots o_0$ and $o_i = \langle b_{8i+7}, b_{8i+6} \ldots b_{8i} \rangle$, $0 \le i \le h-1$, then let $y = b_{g-1}b_{g-2} \ldots b_0$

Example

OS2BS(<0000 0000> <0010 1010> <1100 0001>) = OS2BS(00 2A C1)
= 0000 0000 0010 1010 1100 0001

Consistency

The above specification is believed to be consistent with the functions defined in section 3.14.

Dependencies

None.

Relationship to other conversion functions

OS2BS has an inverse relationship to BS2OS, i.e.:

- BS2OS(OS2BS(*z*)) = *z* for all octet strings *z*; and
- truncate(OS2BS(BS2OS(*y*))) = truncate(*y*) for all bit strings *y*.

4.3.16 OS2EC (octet string to elliptic curve point conversion)

Purpose

This function takes as input an octet string *EP*, and outputs a point *P* on an elliptic curve which this string represents.

Inputs

- *EP* an octet string

Output

- *P* = OS2EC(*EP*) a point on an elliptic curve *E* over a field *F* of characteristic *p*

Assumptions

As described in ISO/IEC 15946-1, a point *P* on an elliptic curve *E* is either 0_E or a pair (*x*, *y*), where *x*, *y* ∈ *F*; if *P* = (*x*, *y*) then it can be represented in one of three forms:

- *uncompressed*: in which case the representation is simply the pair (*x*, *y*);
- *compressed*: in which case the representation is the pair (*x*, $\tilde{y}$), where $\tilde{y} \in \{0,1\}$, and the relationship between $\tilde{y}$ and (*x*, *y*) is as defined in ISO/IEC 15946-1;
- *hybrid*: in which case the representation is the triple (*x*, $\tilde{y}$, *y*), where $\tilde{y}$ is as above.

Operation

Divide EP into two substrings, i.e. *EP* = *H* || *Z*, where *H* is a single octet and *Z* may be empty;

If OS2I(*H*) is not one of 0, 2, 3, 4, 6, 7 then terminate with an error message;

If OS2I(*H*)= 0 then:

let *P* = 0_E;

If OS2I(*H*) = 2 or 3 then:

let *x* = OS2FE(*Z*);

let $\tilde{y}$ = OS2I(H)-2;

derive *y* from (*x*, $\tilde{y}$) as defined in ISO/IEC 15946-1; if no such *y* exists, terminate and fail;

let *P* = (*x*, *y*);

If OS2I(*H*) = 4 then:

divide *Z* into two equal octet strings *X*, *Y* where *Z* = *X* || *Y*;

let *x* = OS2FE(*X*);

let *y* = OS2FE(*Y*);

if (*x*, *y*) is not a point on *E*, terminate and fail;

let *P* = (*x*, *y*);

If OS2I(*H*) = 6 or 7 then:

let $\tilde{y}$ = OS2I(*H*)-6;

divide *Z* into two equal octet strings *X*, *Y* where *Z* = *X* || *Y*;

let *x* = OS2FE(*X*);

let *y* = OS2FE(*Y*);

if $\tilde{y}$ and (*x*, *y*) do not satisfy the relationship defined in ISO/IEC 15946-1, terminate and fail;

if (*x*, *y*) is not a point on *E* terminate and fail;

let *P* = (*x*, *y*)

Consistency

The function as defined above is consistent with the definitions in section 3.15.

Dependencies

- OS2FE (see section 4.3.17)
- OS2I (see section 4.3.18)

Relationship to other conversion functions

OS2EC has an inverse relationship to EC2OS, i.e.:

- OS2EC(EC2OS(*P*, `fmt`)) = *P*, for every point *P* on an elliptic curve.

4.3.17 OS2FE (octet string to field element conversion)

Purpose

This function takes as input an octet string and outputs an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1].

Input

- z an octet string

Output

- s = OS2FE(z) an element of a finite field of $r = p^m$ elements, for a prime p, represented as a string of m integers in the range [0, p-1]

Operation

x = OS2I(z);

if $x \geq r$ then terminate and output an error message;

s = I2FE(x)

Consistency

This function is believed to be consistent with those defined in section 3.16.

Dependencies

- OS2I (see section 4.3.18)
- I2FE (see section 4.3.11)

Relationship to other conversion functions

OS2FE has an inverse relationship to FE2OS, i.e.

- OS2FE(FE2OS(s)) = s for all field elements s; and
- for any octet string z, either OS2FE(z) terminates with an error or FE2OS(OS2FE(z)) = z.

4.3.18 OS2I (octet string to integer conversion)

Purpose

This conversion function takes as input an octet string and outputs the non-negative integer whose binary representation equates to the contents of the octet string.

Input

- *z* an octet string

Output

- *x* = OS2I(*z*) a non-negative integer

Operation

x = BS2I(OS2BS(*z*))

Example

OS2I(<0010 1010> <1100 0001>) = OS2I(2A C1) = 10945

Consistency

The function as defined above is consistent with the definitions in section 3.17, with the following minor difference.

- The function defined in ISO/IEC 14888-3 (see section 3.17.3) has two inputs – the octet string itself and its length. However, in all other respects the definition is consistent with the definition given here.

  If the recommended function OS2I is adopted in this document, the input length parameter should be removed.

Dependencies

- BS2I (see section 4.3.2)
- OS2BS (see section 4.3.15)

Relationship to other conversion functions

OS2I has an inverse relationship to I2OS, i.e.:

- OS2I(I2OS(*x*, *h*)) = *x* for all non-negative integers *x* and all integers *h* satisfying h ≥ $\lceil \log_2(x+1)/8 \rceil$; and
- I2OS(OS2I(*z*), |*z*|) = *z* for all octet strings *z*.

4.3.19 PHF1

The function defined in section 3.18 appears to be very special purpose, and a function of precisely the same characteristics seems unlikely to be required in another standard. As a result, we do not provide a recommended function in this document.

5 Bibliography

[9796-2] ISO/IEC 9796-2:2010, *Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms*.

[9796-3] ISO/IEC 9796-3:2006, *Information technology — Security techniques — Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms*.

[10118-4] ISO/IEC 10118-4:1998, *Information technology – Security techniques – Hash-functions – Part 4: Hash-functions using module arithmetic*.

[11770-4] ISO/IEC 11770-4:2006, *Information technology — Security techniques — Key management — Part 4: Mechanisms based on weak secrets*.

[14888-3] ISO/IEC 14888-3:2016, *Information technology — Security techniques — Digital signatures with appendix — Part 3: Discrete logarithm based mechanisms*.

[15946-1] ISO/IEC 15946-1:2016, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 1: General*.

[18033-2] ISO/IEC 18033-2:2006, *Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers*.

[18033-5] ISO/IEC 18033-5:2015, *Information technology — Security techniques — Encryption algorithms — Part 5: Identity-based ciphers*.

[18370-2] ISO/IEC 18370-2:2016, *Information technology — Security techniques —Blind digital signatures — Part 2: Discrete logarithm based mechanisms*.