IEEE P1363 / D13 (Draft Version 13). Standard Specifications for Public Key Cryptography

Annex E (Informative). Formats.

Comments and suggestions are welcome. Please contact the chair, Ari Singer, at singerar@pb.com.

ANNEX E (Informative)
Formats

E.1 Overview

As outlined in Section 4, the specifications presented in this standard are functional specifications, rather than interface specifications. Therefore, this standard does not specify how the mathematical and cryptographic objects (such as field elements, keys, or outputs of schemes) are to be represented for purposes of communication or storage. This informative Annex provides references to other relevant standards and defines some recommended primitives for that purpose. While the use of this Annex is optional, it is recommended for interoperability.

As octet strings are arguably the most common way to represent data electronically for purposes of communication, this Annex focuses on representing objects as octet strings. One way to accomplish this is to represent data structures in Abstract Syntax Notation 1 (ASN.1—see [ISO98a], [ISO98b], [ISO98c], [ISO98d]) and then to use encoding rules, such as Basic Encoding Rules, Distinguished Encoding Rules or others (see [ISO98e], [ISO98f]) to represent them as octet strings. This Annex does not specify ASN.1 constructs for use in this standard, because the generality of this standard would make such constructs very complex. It is likely that particular implementations only utilize a small part of the options available in this standard and would be better served by simpler ASN.1 constructs. When the use of ASN.1 is desired, ASN.1 constructs defined in the following standards or draft standards may be adapted for use:

— ANSI X9.42 [ANS98b] for DL key agreement
— ANSI X9.63 [ANS98f] for EC key agreement and EC encryption for key transport
— ANSI X9.57 [ANS97c] for DL DSA signatures
— ANSI X9.62 [ANS98e] for EC DSA signatures
— ANSI X9.31 [ANS98a] for IF signatures
— ANSI X9.44 [ANS98c] for IF encryption for key transport

Section E.2 gives recommendations on representing basic mathematical objects as octet strings and Section E.3 gives recommendations on representing the outputs of encryption and signature schemes as octet strings.

E.2 Representing basic data types as octet strings

When integers, finite field elements, elliptic curve points, or binary polynomials need to be represented as octet strings, it should be done as described in this Section. Other primitives for converting between different data types (including bit strings) are defined in Section 5.

E.2.1 Integers (I2OSP and OS2IP)

Integers should be converted to/from octet strings using primitives I2OSP and OS2IP, as defined in the Section 5.5.3 of the standard.

E.2.2 Finite Field Elements (FE2OSP and OS2FEP)

Finite field elements should be converted to/from octet strings using primitives FE2OSP and OS2FEP, as defined in the Section 5.5.4 of the standard.

E.2.3 Elliptic Curve Points (EC2OSP and OS2ECP)

Elliptic curve points should be converted to/from octet strings using primitives EC2OSP and OS2ECP, as defined below.

An elliptic curve point P (which is not the point at infinity O) can be represented in either *compressed* or *uncompressed* form. (For internal calculations, it may be advantageous to use other representations, *e.g.*, the projective coordinates of A.9.6. See Annex A.9.6 also for more information on point compression.) The uncompressed form of P is simply given by its two coordinates. The compressed form is presented below. The octet string format is defined to support both compressed and uncompressed points.

E.2.3.1 Compressed Elliptic Curve Points

The *compressed form* of an elliptic curve point $P \neq \mathsf{O}$ is the pair $(x_P, \tilde{y}_P)$ where x_P is the x-coordinate of P, and $\tilde{y}_P$ is a bit which is computed as follows.

(1) if the field size q is an odd prime, then $\tilde{y}_P = y_P \bmod 2$ (where y_P is taken as an integer in $[0, q - 1]$ and not as a field element). Put another way, $\tilde{y}_P$ is the rightmost bit of y_P.

(2) if the field size q is a power of 2 and $x_P = 0$, then $\tilde{y}_P = 0$.

(3) if the field size q is a power of 2 and $x_P \neq 0$, then $\tilde{y}_P$ is the rightmost bit of the field element $y_P x_P^{-1}$.

Rules (2) and (3) apply for any of the basis representations given in Section 5.3.2.

Procedures for *point decompression* (*i.e.*, recovering y_P given x_P and $\tilde{y}_P$) are given in A.12.8 (for q which is an odd prime) and A.12.9 (for q which is a power of 2).

E.2.3.2 Elliptic Curve Points as Octet Strings—EC2OSP and OS2ECP

The point O should be represented by an octet string containing a single 0 octet. The rest of this section discusses octet string representation of a point $P \neq \mathsf{O}$. Let the x-coordinate of P be x_P and the y-coordinate of P be y_P. Let $(x_P, \tilde{y}_P)$ be the compressed representation of P (see Annex E.2.3.1 for information on point compression).

An octet string PO representing P should have one of the following three formats: *compressed*, *uncompressed*, or *hybrid* (the hybrid format contains information of both compressed and uncompressed form). PO should have the following general form:

$$PO = PC \,\|\, X \,\|\, Y, \text{ where}$$

— PC is a single octet of the form $\mathtt{00000}UC\tilde{Y}$ defined as follows:
  - — Bit U is 1 if the format is uncompressed or hybrid; 0 otherwise.
  - — Bit C is 1 if the format is compressed or hybrid; 0 otherwise.
  - — Bit $\tilde{Y}$ is equal to the bit $\tilde{y}_P$ if the format is compressed or hybrid; 0 otherwise

— X is the octet string of length $\lceil \log_{256} q \rceil$ representing x_P according to FE2OSP (Section 5.5.4)

— Y is the octet string of length $\lceil \log_{256} q \rceil$ representing y_P of P according to FE2OSP (Section 5.5.4) if the format is uncompressed or hybrid; Y is an empty string if the format is compressed

The primitive that converts elliptic curve points to octet strings is called Elliptic Curve Point to Octet String Conversion Primitive or EC2OSP. It takes an elliptic curve point P, the size q of the underlying field and the desired format (compressed, uncompressed or hybrid) as input and outputs the corresponding octet string.

The primitive that converts octet strings to elliptic curve points is called Octet String to Elliptic Curve Point Conversion Primitive or OS2ECP. It takes the octet string and the field size q as inputs and outputs the corresponding elliptic curve point, or “error.” It should use OS2FEP to get x_P. It should use OS2FEP to get y_P if the format is uncompressed. It should use point decompression (see Annex E.2.3.1) to get y_P if the format is compressed. It can get y_P by either of these two means if the format is hybrid. It should output “error” in the following cases:

— if the first octet is 00000000 and the octet string length is not 1
— if the first octet is 00000100, 00000110, or 00000111 and the octet string length is not $1+ 2 \lceil \log_{256} q \rceil$
— if the first octet is 00000010, 00000011 and the octet string length is not $1 + \lceil \log_{256} q \rceil$
— if the first octet is any value other than the six values listed above
— if an invocation of OS2FEP outputs “error”
— if an invocation of the point decompression algorithm outputs “error”

NOTE—The first five bits of the first octet *PC* are reserved and may be used in future formats defined in an addendum to this standard or in future versions. It is essential that they be set to 0 and checked for 0 in order to distinguish this format from other formats.

E.2.4 Polynomials over *GF* (2) (PN2OSP and OS2PNP)

Polynomials over *GF* (2) should be converted to/from octet string using primitives PN2OSP and OS2PNP, as defined below.

The coefficients of a polynomial $p(t)$ over *GF* (2) are elements of *GF* (2) and are therefore represented as bits: the element zero of *GF* (2) is represented by the bit 0, and the element 1 of *GF* (2) is represented by the bit 1 (see Section 5.5.4). Let e be the degree of $p(t)$ and

$$p(t) = a_e\, t^{e} + a_{e-1}\, t^{e-1} + \ldots + a_1\, t + a_0,$$

where $a_e = 1$. To represent $p(t)$ as an octet string, the bits representing its coefficients should be concatenated into a single bit string: $a = a_e \parallel a_{e-1} \parallel \ldots \parallel a_1 \parallel a_0$. The bit string a should then be converted into an octet string using BS2OSP (see Section 5.5.2).

The primitive that converts polynomials over *GF* (2) to octet strings is called Polynomial to Octet String Conversion Primitive or PN2OSP. It takes a polynomial $p(t)$ as input and outputs the octet string.

The primitive that converts octet strings to polynomials over *GF* (2) called Octet String to Polynomial Conversion Primitive or OS2PNP. It takes the octet string as input and outputs the corresponding polynomial over *GF* (2). Let l be the length of the input octet string. OS2PNP should use OS2BSP (see Section 5.5.2) with the octet string and the length $8l$ as inputs. It should output “error” if OS2BSP outputs “error.” The output of OS2BSP should be parsed into $8l$ coefficients, one bit each. Note that at most 7 leftmost coefficients may be zero. The leftmost zero coefficients should be discarded.

E.3 Representing outputs of schemes as octet strings

The signature and encryption schemes in this standard do not produce octet strings as their outputs. When a scheme output (e.g., a signature) needs to be represented as an octet string, it may be done as defined in this section.

E.3.1 Output Data Format for DL/ECSSA

For DL/ECSSA, the output of the signature generation function (see Section 10.2.2) is a pair of integers (c, d). Let r denote the order of the generator (g or G) in the DL or EC settings (see Sections 6.1 and 7.1), and let $l = \lceil \log_{256} r \rceil$ (i.e., l is the length of r in octets). The output (c, d) may be formatted as an octet string as follows: convert the integers c and d to octet strings C and D, respectively, of length l octets each, using the primitive I2OSP, and output the concatenation $C \| D$. To parse the signature, split the octet string into two components C and D, of length l each, and convert them to integers c and d, respectively, using OS2IP. Note that it is essential that both C and D be of length l, even if it means that they have leading zero octets.

NOTE—The output of DL/ECSSA may also be formatted according to the following method, described in more detail in X9.57 [ANS97c] and X9.62 [ANS98e]. Combine c and d into an ASN.1 structure [ISO98a] and encode the structure using some encoding rules, such as Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) [ISO98e].

E.3.2 Output Data Format for IFSSA

For IFSSA, the output of the signature generation function (see Section 10.3.2) is an integer s. Let $k = \lceil \log_{256} n \rceil$ denote the length of the modulus n in octets in the IF setting (see Section 8.1). The output s may be formatted as an octet string by simply converting it to an octet string of length k octets using the primitive I2OSP.

E.3.3 Output Data Format for IFES

For IFES, the output of the encryption function (see Section 11.2.2) is an integer g. Let $k = \lceil \log_{256} n \rceil$ denote the length of the modulus n in octets in the IF setting (see Section 8.1). The output g may be formatted as an octet string by simply converting it to an octet string of length k octets using the primitive I2OSP.