



**BSI Standards Publication**

# **Information technology — Security techniques — Encryption algorithms**

---

Part 5: Identity-based ciphers

# National foreword

This British Standard is the UK implementation of ISO/IEC 18033-5:2015+A1:2021. It supersedes BS ISO/IEC 18033-5:2015, which is withdrawn.

The start and finish of text introduced or altered by amendment is indicated in the text by tags. Tags indicating changes to ISO/IEC text carry the number of the ISO amendment. For example, text altered by ISO/IEC amendment 1 is indicated by A1 A1.

The UK participation in its preparation was entrusted to Technical Committee IST/33, Information security, cybersecurity and privacy protection.

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

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

© The British Standards Institution 2021  
Published by BSI Standards Limited 2021

ISBN 978 0 539 05585 6

ICS 35.030

**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 31 December 2015.

## Amendments/corrigenda issued since publication

Date	Text affected
31 March 2021	Implementation of ISO/IEC amendment 1:2021

# INTERNATIONAL STANDARD

# ISO/IEC 18033-5

First edition  
2015-12-01

---

---

## Information technology — Security techniques — Encryption algorithms —

### Part 5: Identity-based ciphers

*Technologies de l'information — Techniques de sécurité —  
Algorithmes de chiffrement —*

*Partie 5: Chiffrements identitaires*



Reference number  
ISO/IEC 18033-5:2015(E)

ISO/IEC 2015



## **COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2015, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Ch. de Blandonnet 8 • CP 401  
CH-1214 Vernier, Geneva, Switzerland  
Tel. +41 22 749 01 11  
Fax +41 22 749 09 47  
copyright@iso.org  
www.iso.org

# Contents

Page

<b>Foreword</b>	<b>v</b>
<b>Introduction</b>	<b>vi</b>
<b>1 Scope</b>	<b>1</b>
<b>2 Normative references</b>	<b>1</b>
<b>3 Terms and definitions</b>	<b>1</b>
<b>4 Symbols, abbreviated terms and conversion functions</b>	<b>2</b>
4.1 Symbols	3
4.2 Abbreviated terms	3
4.3 Conversion functions	4
<b>5 Cryptographic transforms</b>	<b>5</b>
5.1 General	5
5.2 The function <i>IHF1</i>	5
5.3 The function <i>SHF1</i>	6
5.4 The function <i>PHF1</i>	6
5.5 The function <i>IHF2</i>	7
<b>6 General model for identity-based encryption</b>	<b>8</b>
6.1 Composition of algorithms	8
6.2 Plaintext length	8
6.3 Use of labels	9
6.4 Ciphertext format	9
6.5 IBE operation	9
<b>7 General model for identity-based hybrid encryption</b>	<b>10</b>
7.1 General	10
7.2 Identity-based key encapsulation	10
7.2.1 Composition of algorithms	11
7.2.2 Prefix-freeness	11
7.3 Data encapsulation	11
7.3.1 Composition of algorithms	11
7.4 Identity-based hybrid encryption operation	11
7.4.1 System parameters	11
7.4.2 Set up	12
7.4.3 Private key extraction	12
7.4.4 Encryption	12
7.4.5 Decryption	12
<b>8 Identity-based encryption mechanism</b>	<b>12</b>
8.1 General	12
8.2 The BF mechanism	13
8.2.1 Set up	13
8.2.2 Private key extraction	13
8.2.3 Encryption	14
8.2.4 Decryption	15
<b>9 Identity-based hybrid encryption mechanisms</b>	<b>15</b>
9.1 General	15
9.2 The SK key encapsulation mechanism	15
9.2.1 Set up	15
9.2.2 Private key extraction	16
9.2.3 Session key encapsulation	17
9.2.4 Session key de-encapsulation	17
9.3 The BB1 key encapsulation mechanism	18
9.3.1 Set up	18
9.3.2 Private key extraction	18

A1

	9.3.3	Session key encapsulation.....	19	
	9.3.4	Session key de-encapsulation.....	19	
A1	9.4	The SM9 key encapsulation mechanism.....	20	
	9.4.1	Set up.....	20	
	9.4.2	Private key extraction .....	20	
	9.4.3	Session key encapsulation .....	21	
	9.4.4	Session key de-encapsulation.....	21	A1
<b>Annex A (normative) Object identifiers .....</b>			<b>22</b>	
<b>Annex B (informative) Security considerations.....</b>			<b>24</b>	
<b>Annex C (informative) Numerical examples .....</b>			<b>25</b>	
<b>Annex D (informative) Mechanisms to prevent access to keys by third parties.....</b>			<b>41</b>	
<b>Bibliography .....</b>			<b>42</b>	

## 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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives](http://www.iso.org/directives)).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see [www.iso.org/patents](http://www.iso.org/patents)).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: [Foreword — Supplementary information](#).

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 27, *IT Security techniques*.

ISO/IEC 18033 consists of the following parts, under the general title *Information technology — Security techniques — Encryption algorithms*:

- *Part 1: General*
- *Part 2: Asymmetric ciphers*
- *Part 3: Block ciphers*
- *Part 4: Stream ciphers*
- *Part 5: Identity-based ciphers*

Further parts may follow.

Annex A forms a normative part of this part of ISO/IEC 18033. Annex B, Annex C and Annex D are informative only.

## Introduction

Use of a public key encryption mechanism requires reliable identification of the correct public key to be used for encryption. A public key infrastructure (PKI) provides functions to give a trusted link between an entity and to enable the current status of the public key to be determined. In a PKI, a certification authority (CA) issues a certificate binding a public key to the owner's identifier together with other key specific information, e.g. the validity period. If a public key is deemed to be invalid before its expiry date, then potential users of the public key need to be notified, e.g. by the issue of a CA-signed Certificate Revocation List (CRL). The generation and distribution of certificates and CRLs poses a major management problem, which the mechanisms in this part of ISO/IEC 18033 are designed to address. On encrypting, an encryptor first obtains the CRL and checks the current status of the certificate. Then the encryptor verifies the certificate, and finally encrypts a message. Therefore, the encryptor has to be provided with some means of accessing the current CRL, and additionally it should not require excessive time and computational resources for checking the validity of a certificate whenever it encrypts a message.

Identity-based encryption (IBE) is a type of asymmetric encryption that allows a decryptor to set its public key to an arbitrary string. By setting the public key to an easily identifiable string (e.g. an e-mail address), an encryptor can gain assurance in its correctness without using a certificate. Moreover, if a short validity period can be arranged, significantly shorter than the updating period of a CRL in a conventional PKI, an encryptor can generate a ciphertext without checking the current status of the public key because revocation is unlikely to occur during such a short period. As a result IBE is expected to reduce the certificate management workload.

The use of IBE requires a Private Key Generator (PKG), which generates private keys for all decryptors using its master secret key; this contrasts with 'traditional' asymmetric encryption mechanisms, such as those specified in ISO/IEC 18033-2, in which entities generate their own public/private key pairs. As a result, use of IBE is only appropriate when it is acceptable for a third party to have decryption access to all encrypted data.

The identity-based encryption mechanisms are specified in [Clauses 8](#) and [9](#). <sup>A1</sup> The specified mechanisms are the BF identity-based encryption mechanism, the SK identity-based key encapsulation mechanism, the BB1 identity-based key encapsulation mechanism and the SM9 identity-based key encapsulation mechanism and encryption mechanisms. <sup>A1</sup>

The specifications in this part of ISO/IEC 18033 do not prescribe protocols for reliably obtaining public values, for proof of possession of a private key, or for validation of either public values or private keys.

Certain sections of [Clause 5](#), [Clause 8](#) and [Clause 9](#) of this part of ISO/IEC 18033 have been reprinted with permission from [\[7\]](#) IEEE Std 1363.3-2013 - IEEE Standard for Identity-Based Cryptographic Techniques using Pairings. Reprinted with permission from IEEE. Copyright 2013. All rights reserved.

<sup>A1</sup> The content of 9.4 follows Reference [\[8\]](#). <sup>A1</sup>

Annex A gives the assignment of object identifiers to the algorithms specified in this part of ISO/IEC 18033. Annex B describes security considerations for each specified mechanism and Annex C provides numerical examples. Annex D introduces techniques which can be used to remove the decryption capability of the PKG, and thereby reduce the level of trust required in this entity.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 18033 may involve the use of patents. The ISO and IEC take no position concerning the evidence, validity, and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout



the world. In this respect, the statements of the holders of these patent rights are registered with the ISO and IEC. Information may be obtained from the following:

Patent holder name: Nippon Telegraph and Telephone Corporation

Postal address: Licensing Group, Intellectual Property Center

9-11, Midori-cho, 3-Chome Musashino-Shi, Tokyo 180-8585 Japan

Patent holder name: IBM Corporation

Postal address: IBM Intellectual Property Licensing

North Castle Drive, Armonk, NY 10504 USA

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.

ISO ([www.iso.org/patents](http://www.iso.org/patents)) and IEC (<http://patents.iec.ch>) maintain on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

# Information technology — Security techniques — Encryption algorithms —

## Part 5: Identity-based ciphers

### 1 Scope

This part of ISO/IEC 18033 specifies identity-based encryption mechanisms. For each mechanism the functional interface, the precise operation of the mechanism, and the ciphertext format are specified. However, conforming systems may use alternative formats for storing and transmitting ciphertexts.

### 2 Normative references

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

ISO/IEC 18033-1, *Information technology — Security techniques — Encryption algorithms — Part 1: General*

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

ISO/IEC 18033-3, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

### 3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 18033-1 and the following apply.

#### 3.1

##### **decryptor**

entity which decrypts ciphertexts

#### 3.2

##### **encryptor**

entity which encrypts plaintexts

#### 3.3

##### **hybrid encryption**

encryption performed using a hybrid cipher

#### 3.4

##### **identifier**

object that represents something and enables one to identify it

#### 3.5

##### **identity string**

string that represents an identity

**ISO/IEC 18033-5:2015+A1:2021(E)****3.6****identity-based cipher**

asymmetric cipher in which the encryption algorithm takes an arbitrary string as a public key

**3.7****identity-based hybrid cipher**

cipher which is both a hybrid cipher and an identity-based cipher

**3.8****identity-based key encapsulation mechanism**

key encapsulation mechanism for which the encryption process takes an arbitrary string as a public key

**3.9****master-public key**

public value uniquely determined by the corresponding master-secret key

**3.10****master-secret key**

secret value used by the private key generator to compute private keys for an IBE algorithm

**3.11****private key extraction algorithm**

method used by the private key generator to compute private keys for an IBE algorithm

**3.12****private key generator**

entity or function which generates a set of private keys

**3.13****public key encryption**

encryption performed using an asymmetric cipher

**3.14****string**

ordered sequence of symbols

**3.15****set up**

process by which the system parameters for an IBE algorithm are selected

**3.16****set up algorithm**

process which generates a master-secret key and the corresponding master-public key, together with some part of the system parameters

**3.17****system parameters**

parameters for cryptographic computation including a selection of a particular cryptographic scheme or function from a family of cryptographic schemes or functions, or from a family of mathematical spaces

**3.18****trusted third party**

security authority, or its agent, trusted by other entities with respect to security related activities

**4 Symbols, abbreviated terms and conversion functions**

For the purposes of this part of ISO/IEC 18033, the symbols and abbreviated terms given in ISO/IEC 18033-1 and the following apply.

## 4.1 Symbols

$[a, \dots, b]$	the set of integers $\{x : a \leq x < b\}$ .
$\tilde{x} \oplus \tilde{y}$	if $\tilde{x}$ and $\tilde{y}$ are bit/octet strings of the same length, the bit-wise exclusive-or (XOR) of the two strings.
$\langle x_1, \dots, x_l \rangle$	a tuple $x_1, \dots, x_l$ of elements.
$\tilde{x}    \tilde{y}$	if $\tilde{x}$ and $\tilde{y}$ are bit/octet strings, the concatenation of the two strings $\tilde{x}$ and $\tilde{y}$ , resulting in the string consisting of $\tilde{x}$ followed by $\tilde{y}$ .
$\gcd(a, b)$	for integers $a$ and $b$ , the greatest common divisor of $a$ and $b$ , i.e., the largest positive integer that divides both $a$ and $b$ (or 0 if $a = b = 0$ ).
$a b$	a relation between integers $a$ and $b$ that holds if and only if $a$ divides $b$ i.e., there exists an integer $c$ such that $b = ac$ .
$a \nmid b$	a relation between integers $a$ and $b$ that holds if and only if $a$ does not divide $b$ i.e., there does not exist any integer $c$ such that $b = ac$ .
$a \equiv b \pmod{n}$	for a non-zero integer $n$ , a relation between integers $a$ and $b$ that holds if and only if $a$ and $b$ are congruent modulo $n$ , i.e., $n (a - b)$ .
$a \pmod{n}$	for integer $a$ and positive integer $n$ , the unique integer $r \in [0, \dots, n)$ such that $r \equiv a \pmod{n}$ .
$a^{-1} \pmod{n}$	for integer $a$ and positive integer $n$ , such that $\gcd(a, n) = 1$ , the unique integer $b \in [0, \dots, n)$ such that $ab \equiv 1 \pmod{n}$ .
$GF(q)$	the finite field containing $q$ elements, where $q$ is a power of a prime.
$E / GF(q)$	an elliptic curve defined over the field $GF(q)$ .
$E(GF(q))$	the additive group of points on the elliptic curve $E / GF(q)$ .
$E(GF(q))[n]$	the subgroup of $E(GF(q))$ consisting of all points of order $n$ .
$\#E(GF(q))$	the number of points of an elliptic curve defined over the field $GF(q)$ .
$\lceil x \rceil$	smallest integer greater than or equal to the real number $x$ . $\lceil x \rceil$

## 4.2 Abbreviated terms

$CT$	ciphertext, an octet string.
DEM	data encapsulation mechanism.
IBE	identity-based encryption.
IBhE	identity-based hybrid encryption.
$ID$	octet string uniquely assigned to a decryptor.
$ID_b$	binary representation of $ID$ .
$K$	session key for DEM.
$\kappa$	security parameter.

**ISO/IEC 18033-5:2015+A1:2021(E)**

<b>KEM</b>	key encapsulation mechanism.
<b><i>L</i></b>	label, an octet string.
<b><i>mpk</i></b>	master-public key of IBE.
<b><i>Msg</i></b>	plaintext, an octet string.
<b><i>Msg<sub>b</sub></i></b>	binary representation of <i>Msg</i> .
<b><i>msk</i></b>	master-secret key of IBE.
<b><i>parms</i></b>	system parameters of IBE.
<b>PKG</b>	private key generator.
<b><i>sk<sub>ID</sub></i></b>	private key corresponding to <i>ID</i> of IBE.

**4.3 Conversion functions**

The following conversion functions are given in ISO/IEC 18033-2.

<b><i>BS2IP</i></b>	bit string to integer conversion primitive.
<b><i>BS2OSP</i></b>	bit string to octet string conversion primitive.
<b><i>EC2OSP</i></b>	elliptic curve to octet string conversion primitive.
<b><i>FE2OSP</i></b>	field element to octet string conversion primitive.
<b><i>FE2IP</i></b>	field element to integer conversion primitive.
<b><i>I2BSP</i></b>	integer to bit string conversion primitive.
<b><i>I2OSP</i></b>	integer to octet string conversion primitive.
<b><i>OS2ECP</i></b>	octet string to elliptic curve conversion primitive.
<b><i>OS2FEP</i></b>	octet string to field element conversion primitive.
<b><i>OS2IP</i></b>	octet string to integer conversion primitive.
<b><i>OS2BSP</i></b>	octet string to bit string conversion primitive.
<b><i>Oct(m)</i></b>	the octet whose integer value is <i>m</i> .
<b><i>Len(n)</i></b>	the number of octets of an integer <i>n</i> .

## 5 Cryptographic transforms

### 5.1 General

**A1** The schemes specified in this document make use of four cryptographic transformations, *IHF1*, *SHF1*, *PHF1* and *IHF2* as specified below. **A1** These transformations make use of hash-functions specified in ISO/IEC 10118-3.

**A1** Annex A lists the object identifiers which shall be used to identify the algorithms specified in this document.

Annex B describes security considerations for each specified mechanism.

Annex C provides numerical examples.

Annex D introduces techniques which can be used to remove the decryption capability of the PKG, and thereby reduce the level of trust required in this entity. **A1**

### 5.2 The function *IHF1*

*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  $2\kappa$ .
- c) Let  $t_1 = h_0 || str$ .
- d) Let  $h_1 = H(t_1)$ .
- e) Let  $v_1 = BS2IP(h_1)$ .

**ISO/IEC 18033-5:2015+A1:2021(E)**

- 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  $v_2 \bmod n$ .

**5.3 The function SHF1**

Returns an  $n$ -bit string that is based on a cryptographic hash function applied to an input string.

Input:

- A string  $str \in \{0,1\}^*$
- A security parameter  $\kappa \in \{112, 128, 192, 256\}$
- An integer  $n, n > 0$

Assumptions: The string  $str$  is within the allowed range of values for inputs to the relevant hash function. The integer  $n$  has the property that  $n \leq 4\kappa$ .

Output:

- A string  $v \in \{0,1\}^n$

Operation: Use the following steps.

- Output  $I2BSP(IHF1(str, 2^n, \kappa))$ .

**5.4 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  $E/GF(q): y^2 = x^3 + ax$ . There are other types of pairing-friendly elliptic curves for which *PHF1* is not suitable.

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  $q \equiv 2 \pmod{3}$  when  $j=0$  or  $q \equiv 3 \pmod{4}$  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  $p \nmid \#E(GF(q))$  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  $y = IHF1(str, q, \kappa)$ .
  - 2) Let  $x = (y^2 - b)^{(2q-1)/3} \pmod{q}$ .
  - 3) Let  $J = (x, y)$ .
- c) Else if  $j=1$  perform the following steps:
  - 1) Let  $x = IHF1(str, q, \kappa)$ .
  - 2) Let  $z = x^3 + ax \pmod{q}$ .
  - 3) If the Jacobi symbol  $(z/q) = +1$  then perform the following steps:
    - i) Let  $y = z^{(q+1)/4} \pmod{q}$ .
    - ii) Let  $J = (x, y)$ .
  - 4) If the Jacobi symbol  $(z/q) = -1$  then perform the following steps:
    - i) Let  $y = (-z)^{(q+1)/4} \pmod{q}$ .
    - ii) Let  $J = (-x, y)$ .
- d) Return  $rJ$ .

## **5.5 The function $IHF2$**

$IHF2$  is based on the key derivation function  $KDF2$  defined in ISO/IEC 18033-2.  $KDF2(x, l)$  parameterized by a cryptographic hash function takes an octet string  $x$  and a non-negative integer  $l$  as input, and outputs an octet string of length  $l$ .  $KDF2-a(x, b)$  outputs the first  $b$  bits from  $KDF2(x, \lceil b/8 \rceil)$ .  $IHF2$  take three items as input and outputs an integer in a specified range.

Input:

- A bit string  $str \in \{0,1\}^*$
- A security parameter  $\kappa \in \{128\}$
- A non-negative integer  $n$  with bit-length  $b_n$

Output:

- An integer  $x$ ,  $0 < x < n$ .



## ISO/IEC 18033-5:2015+A1:2021(E)

Operation: Perform the following steps.

- a) If  $\kappa = 128$ ,  $KDF2$  uses SM3 as the hash function.
- b) Let  $hlen = 8 \lceil (5 b_n)/32 \rceil$ .
- c) Compute  $Ha = KDF2\text{-}a(str, hlen)$ .
- d) Output  $(BS2IP(Ha) \bmod (n-1)) + 1$ . A1

## 6 General model for identity-based encryption

### 6.1 Composition of algorithms

An identity-based encryption scheme consists of the following four algorithms.

*IBE.Setup*( $\kappa$ ). Given a security parameter  $\kappa$ , generate a tuple  $\langle parms, mpk, msk \rangle$ , where *parms* denotes system parameters, *msk* denotes a master-secret key and *mpk* is the corresponding master-public key.

*IBE.Extract*(*parms*, *mpk*, *msk*, *ID*). Given a master-secret key *msk*, the corresponding master-public key *mpk* and an octet string *ID* with *parms*, generate a private key  $sk_{ID}$  for *ID*.

*IBE.Enc*(*parms*, *mpk*, *ID*, *L*, *Msg*). Given a plaintext *Msg*, a label *L* and an octet string *ID* with *parms* and *mpk*, do the encryption and output the ciphertext of *Msg*, *CT*, for *ID*. Note that *Msg*, *L* and *CT* are octet strings.

*IBE.Dec*(*parms*, *mpk*, *ID*,  $sk_{ID}$ , *L*, *CT*). Given a private key  $sk_{ID}$  with *parms*, *mpk*, *ID* and *L*, decrypt a ciphertext *CT* and output the underlying plaintext.

In general, the setup, key extraction and encryption algorithms are probabilistic algorithms, while the decryption algorithm is deterministic. It is recommended that applications establish a methodology for authenticating access to private keys by using the *ID* string as an identity in a trusted authentication system. The details of authenticating the key request are beyond the scope of this part of ISO/IEC 18033, but are critical for the security of an implemented application.

**NOTE 1** Semantic security against an adaptive chosen ciphertext attack<sup>[4]</sup> is regarded by the cryptographic research community as the appropriate security level that a general purpose IBE mechanism should satisfy. Each IBE mechanism described in this part of ISO/IEC 18033 satisfies this security level. The formal definition of this security notion is described in Annex B.

**NOTE 2** A basic requirement of any IBE mechanism is correctness. For any *ID*/ $sk_{ID}$  pair and for any plaintext of defined length, the ciphertext of *ID* under a master-public key and system parameters *ID* should be able to be decrypted with the private key  $sk_{ID}$  under the master-public key and the system parameters *ID* to the original plaintext. This requirement may be relaxed, so that it holds only for all but a negligible fraction of *ID*/ $sk_{ID}$  pairs.

### 6.2 Plaintext length

Three types of plaintext length of IBE are defined as follows.

- An arbitrary-plaintext-length IBE encrypts plaintexts of an arbitrary length.
- A fixed-plaintext-length IBE only encrypts plaintexts whose length (in octets) is equal to a fixed value *IBE.MsgLen*.

- A bounded-plaintext-length IBE only encrypts plaintexts whose length (in octets) is less than or equal to a fixed value  $IBE.MaxMsgLen(mpk)$ . Here, the maximum plaintext length may depend on the system parameter  $mpk$ .

### 6.3 Use of labels

A label is an octet string whose value is used by the encryption and decryption algorithms. It may contain public data that is implicit from context and need not be encrypted, but that should nevertheless be bound to the ciphertext. A label is an octet string that is meaningful to the application using the IBE scheme, and that is independent of the implementation of the IBE scheme. Three types of label length of IBE are defined as follows.

- An arbitrary-label-length IBE is one in which the encryption and decryption algorithms accept labels of arbitrary length.
- A fixed-label-length IBE is one in which the encryption and decryption algorithms only accept labels whose length (in octets) is equal to a fixed value  $IBE.LabelLen$ .
- A bounded-label-length IBE is one in which the encryption and decryption algorithms only accept labels whose length (in octets) is less than or equal to a fixed value  $IBE.MaxLabelLen$ .

**NOTE** The traditional notion of security against an adaptive chosen ciphertext attack has been extended in Annex B, so that intuitively, for a secure IBE mechanism, the encryption algorithm should bind the label to the ciphertext in an appropriate “non-malleable” fashion.

### 6.4 Ciphertext format

This part of ISO/IEC 18033 describes ciphertext formats in an octet string; however, an implementation is free to store and/or transmit ciphertexts in alternative formats, if it is required. Moreover, ciphertexts in the format prescribed here may not necessarily appear in the encryption or decryption process, since converting the format into an alternative format may be collapsed into a single, functionally equivalent process.

**NOTE** Besides promoting interoperability, prescribing the format of a ciphertext is necessary in order to make meaningful claims and to reason about the security of an IBE against adaptive chosen ciphertext attacks.

### 6.5 IBE operation

In an IBE setup, a trusted third party generates a master-secret key, the corresponding master-public key and system parameters by invoking the  $IBE.Setup$  algorithm and makes the master-public key and the system parameters public, keeping the master-secret key private. When a decryptor requests the private key, a private key issuer invokes the  $IBE.Extract$  algorithm and issues its output to the decryptor as the private key via a secure channel. An encryptor computes a ciphertext by running the  $IBE.Enc$  algorithm with the decryptor’s identity string. The decryptor derives the underlying plaintext from the received ciphertext by running  $IBE.Dec$  with the private key.

The following mechanisms and protocols are outside of the scope of this part of ISO/IEC 18033. One can refer to ISO/IEC 11770 for designing key management operation.

- protocol for making a master-public key and system parameters available.
- protocol for distributing private keys to decryptors.
- protocol for making an identity string available to encryptors.
- mechanism for storing a master-secret key or a private key securely.
- mechanism for confirming a link between a master-public key or system parameters and its issuer.

Each of the IBE mechanisms specified in this part of ISO/IEC 18033 are actually members of families of IBE mechanisms, where a particular IBE is selected from the family by choosing particular values for the system parameters defining the family of IBE mechanisms. For an IBE mechanism selected from a family of mechanisms, prior to master-secret key/master-public key pair generation, specific values of the system parameters for the family should be chosen. Depending on the conventions used for encoding master-public keys, some of the choices of the system parameters may be embedded in the encoding of the master-public key as well. These system parameters shall remain fixed throughout the lifetime of the master-public key.

**NOTE** For example, if an IBE mechanism is parameterized in terms of a cryptographic hash function, the choice of hash function should be fixed once and for all at some point prior to the generation of a master-secret key/master-public key pair, and the encryption and decryption algorithms should use the chosen hash function throughout the lifetime of the master-public key. Failure to abide by this rule not only makes an implementation non-conforming, but also invalidates the security analysis for the mechanism, and can in some cases expose the implementation to severe security risks.

## 7 General model for identity-based hybrid encryption

### 7.1 General

An identity-based hybrid encryption scheme is used for encrypting a large plaintext making use of the advantages of the identity-based encryption scheme, where an IBE is utilized to encrypt a decryption key for encrypting the actual message using symmetric cryptographic techniques. An identity-based hybrid encryption scheme is built from two lower-level “building blocks”: an identity-based key encapsulation scheme and a data encapsulation scheme.

### 7.2 Identity-based key encapsulation

#### 7.2.1 Composition of algorithms

An identity-based key encapsulation scheme consists of the following four algorithms.

*KEM.Setup*( $\kappa$ ). Given a security parameter  $\kappa$ , generate a tuple  $\langle \text{parms}, \text{mpk}, \text{msk} \rangle$ , where *parms* denotes system parameters, *msk* denotes a master-secret key and *mpk* is the corresponding master-public key.

*KEM.Extract*(*parms*, *mpk*, *msk*, *ID*). Given a master-secret key *msk*, the corresponding master-public key *mpk*, and an octet string *ID* with *parms*, generate a private key  $sk_{ID}$  for *ID*.

*KEM.Enc*(*parms*, *mpk*, *ID*). Given an octet string *ID* with *parms* and *mpk*, output an octet string *K* and the ciphertext of *K*,  $CT_{KEM}$ , for *ID*. Note that  $CT_{KEM}$  is an octet string.

*KEM.Dec*(*parms*, *mpk*, *ID*,  $sk_{ID}$ ,  $CT_{KEM}$ ). Given a private key  $sk_{ID}$  with *parms*, *mpk* and *ID*, decrypt a ciphertext  $CT_{KEM}$  and output the underlying octet string.

In the general setup, key extraction and encapsulation algorithms are probabilistic algorithms, while the de-encapsulation algorithm is deterministic. The de-encapsulation algorithm may fail under some circumstances with a negligible probability. A key encapsulation scheme also specifies a positive integer *KEM.KeyLen* — the length of the secret key output by *KEM.Enc* and *KEM.Dec*.

**NOTE 1** The required security level that each key encapsulation mechanism should satisfy is described in Annex B.

NOTE 2 Any key encapsulation mechanism should satisfy a correctness property analogous to the correctness property of an IBE mechanism: for any  $ID / sk_{ID}$  pair and for any octet string  $K$  whose length is within its implementation-defined limits, any encapsulation of  $K$  with predetermined system parameters under an  $ID$  decrypts with the system parameters under the  $ID$  to the original  $K$ . This requirement may be relaxed so that it holds only for all but a negligible fraction of  $ID / sk_{ID}$  pairs.

### 7.2.2 Prefix-freeness

Additionally, a key encapsulation mechanism shall satisfy the following property. The set of all possible ciphertext outputs of the encryption algorithm should be a subset of a candidate set of octet strings (that may depend on the public key), such that the candidate set is prefix free and elements of the candidate set are easy to recognize (given either the public key or the private key).

## 7.3 Data encapsulation

### 7.3.1 Composition of algorithms

A data encapsulation scheme consists of the following two algorithms.

- a)  $DEM.Enc(K, L, Msg)$ . Given octet strings  $K$  and  $L$ , compute the encryption of  $Msg$ ,  $CT_{DEM}$ . Note that  $Msg$  and  $CT_{DEM}$  are octet strings, and  $L$  and  $Msg$  may have length as described in 6.3 and 6.2 respectively. The bit length of  $K$  is  $DEM.KeyLen$ .
- b)  $DEM.Dec(K, L, CT_{DEM})$ . Given octet strings  $K$  and  $L$ , decrypt a ciphertext  $CT_{DEM}$  and output the underlying plaintext.

The encryption algorithm may fail if the lengths  $L$  or  $Msg$  exceed some (very large) implementation-defined limits. The decryption algorithm may fail under some circumstances with a negligible probability.

**A1** The allowable data encapsulation mechanisms are those described in ISO/IEC 18033-2. **A1**

NOTE The encryption and decryption algorithms should be deterministic, and should satisfy the following correctness requirement: for all session keys  $K$ , all labels  $L$ , and all plaintexts  $Msg$ , such that the lengths of  $L$  and  $Msg$  do not exceed the implementation-defined limits, and

$$c) \quad DEM.Dec(K, L, DEM.Enc(K, L, Msg)) = Msg.$$

## 7.4 Identity-based hybrid encryption operation

### 7.4.1 System parameters

An identity-based hybrid encryption scheme (for short, IBhE) is a family of IBE schemes parameterized by the following system parameters:

- KEM: an identity-based key encapsulation scheme, as described in 7.2.
- DEM: a data encapsulation scheme, as described in 7.3.

Any combination of KEM and DEM may be used, provided  $KEM.KeyLen = DEM.KeyLen$ .

NOTE 1 If DEM is instantiated by a fixed-label-length data encapsulation mechanism, with labels restricted to length  $DEM.LabelLen$ , then IBhE is a fixed-label-length identity-based encryption mechanism with  $IBhE.LabelLen = DEM.LabelLen$ .

NOTE 2 If DEM is instantiated by a fixed-plaintext-length data encapsulation mechanism, with plaintexts restricted to length  $DEM.MsgLen$ , then IBhE is a fixed-plaintext-length identity-based encryption mechanism with  $IBhE.MsgLen = DEM.MsgLen$ .

## ISO/IEC 18033-5:2015+A1:2021(E)

NOTE 3 For all the allowable choices of KEM, the value of  $KEM.KeyLen$  is a system parameter that may be chosen so as to equal  $DEM.KeyLen$ . Thus, all possible combinations of allowable KEM and DEM may be realized by appropriate choices of system parameters.

**A)** NOTE 4 The third mechanism defined in 9.4 will work to encrypt messages with either DEM2 or DEM3, which are specified in ISO/IEC 18033-2. In these DEMs, the required hash function is SM3, specified in ISO/IEC 10118-3, and the required block cipher is described in ISO/IEC 18033-3. The required message authentication code is generated by the evaluation function  $MA.eval(K'', MS) = SM3(MS || K'')$ , where  $K''$  is a secret key which is part of the session key  $K$ , and  $MS$  is the octet string to be authenticated as specified in DEM2 and DEM3. The label input to both DEMs is empty. **A1**

### 7.4.2 Set up

The set up algorithm for IBhE is the same as that of the underlying KEM. Let  $parms$  denote system parameters and  $\langle msk, mpk \rangle$  denote a master-secret key/master public-key pair.

### 7.4.3 Private key extraction

The private key extraction algorithm for IBhE is the same as that of the underlying KEM. Let  $sk_{ID}$  denote a private key of  $ID$ .

### 7.4.4 Encryption

The encryption algorithm  $IBhE.Enc$  takes as input a master-public key  $mpk$ , a label  $L$  and a plaintext  $Msg$  with system parameters  $parms$ . It runs as follows.

- Compute  $\langle K, CT_{KEM} \rangle = KEM.Enc(parms, mpk, ID)$ .
- Compute  $CT_{DEM} = DEM.Enc(K, L, Msg)$ .
- Set  $CT = CT_{KEM} || CT_{DEM}$ .
- Output  $CT$ .

### 7.4.5 Decryption

The decryption algorithm  $IBhE.Dec$  takes as input a private key  $sk_{ID}$ , an octet string  $ID$ , a label  $L$ , and a ciphertext  $CT$  with system parameters  $parms$ . It runs as follows.

- Using the prefix-freeness property of the ciphertexts associated with KEM, parse  $CT$  as  $CT = CT_{KEM} || CT_{DEM}$ , where  $CT_{KEM}$  and  $CT_{DEM}$  are octet strings such that  $CT_{KEM}$  is an element of the candidate set of possible ciphertexts associated with KEM. This step fails if  $CT$  cannot be parsed.
- Compute  $K = KEM.Dec(parms, mpk, ID, sk_{ID}, CT_{KEM})$ .
- Compute  $Msg = DEM.Dec(K, L, CT_{DEM})$ .
- Output  $Msg$ .

NOTE The security of IBhE is discussed in Annex B. It is only remarked here that as long as KEM and DEM satisfy the appropriate security properties, then IBhE will be secure against adaptive chosen ciphertext attack.

## 8 Identity-based encryption mechanism

### 8.1 General

In this clause an identity-based encryption mechanism is specified. This mechanism is specified to use the following primitives. The hash function  $H_1$  is used for supersingular curves but some other (possibly similar) scheme of hashing a string to a point is required for other curves.

Take a security parameter  $\kappa$ , and a key size parameter  $\delta$ , the mechanism requires the following algorithms:

a) A random bit generation algorithm [1]:

—  $R_1$ : a source of random values in the space  $\{0,1\}^\delta$ .

b) Four hash functions:

—  $H_1: \{0,1\}^* \rightarrow G_1$  where  $H_1(s) = PHF1(s, \kappa, j, q, c, p)$  with  $j=0$  representing the elliptic curve  $E / GF(q): y^2 = x^3 + c$  and  $j=1$  representing the elliptic curve  $E / GF(q): y^2 = x^3 + cx$  and  $p$  is a prime with  $p \nmid \#E(GF(q))$  and  $p^2 \nmid \#E(GF(q))$  for elliptic curve  $E$  defined by the flag  $j$

—  $H_2: G_3 \rightarrow \{0,1\}^\delta$  where  $H_2(x) = SHF1(OS2BSP(z), \delta, \kappa)$  and  $z = FE2OSP(x)$

—  $H_3: \{0,1\}^\delta \times \{0,1\}^\delta \rightarrow Z_p^*$  where  $H_3(s_1, s_2) = IHF1(s_1 || s_2, p-1, \kappa) + 1$

—  $H_4: \{0,1\}^\delta \rightarrow \{0,1\}^\delta$  where  $H_4(s) = SHF1(s, \delta, \kappa)$

## 8.2 The BF mechanism

### 8.2.1 Set up

The setup operation creates public system parameters and a master-secret key. The master-secret key shall be secured by the private key issuer, as all private keys calculated within the system depend on it. It is recommended that applications establish a methodology for changing the master-secret key, the corresponding master-public key and system parameters on a regular basis, and have a methodology for handling the disclosure of a master-secret key. However this is out of scope of this part of ISO/IEC 18033.

The steps to setup the private key issuer and system parameters are:

- Establish the set of base groups  $G_1, G_2, G_3$  and a pairing  $e: G_1 \times G_2 \rightarrow G_3$ .  $G_1$  shall be of the form  $E(GF(q))[p]$ , where  $p$  and  $q$  are primes.
- Select a random generator  $Q$  in  $G_2$ .
- Generate a random master secret  $s$  in  $Z_p^*$ . Calculate the corresponding  $R$  as  $sQ$ .
- Make the system parameters and the master-public key set  $parms = \langle Q, G_1, G_2, G_3, e \rangle$  and  $mpk = R$  available. Secure the master-secret key  $s$ .

ISO/IEC 15946-1 contains recommendations for the generation of the groups, group generators and pairing.

### 8.2.2 Private key extraction

The extract operation takes an arbitrary identity string  $ID_b$  in  $\{0,1\}^*$  and calculates the corresponding private key  $sk_{ID}$  in  $G_1$ . The algorithm to compute the private key  $sk_{ID}$  corresponding to an identity string  $ID_b$  is as follows:

Input:

- The system parameters  $parms = \langle Q, G_1, G_2, G_3, e \rangle$
- The master-public key  $mpk = R$
- The master-secret key  $msk = s$



**ISO/IEC 18033-5:2015+A1:2021(E)**

— An identity string  $ID_b$

Output:

— The derived private key  $sk_{ID}$  that is an element of  $G_1$ .

Operation: Use the following steps to compute  $sk_{ID}$ .

- Compute the identity element  $M = H_1(ID_b)$ .
- Set  $sk_{ID} = sM$ .
- Output  $sk_{ID}$ .

The correctness of the value  $sk_{ID}$  can be verified by using the following algorithm:

Input:

- The system parameters  $parms = \langle Q, G_1, G_2, G_3, e \rangle$
- The master-public key  $mpk = R$
- An identity string  $ID_b$
- The corresponding private key  $sk_{ID} = sM$

Output:

- The value “valid” if  $sk_{ID}$  is consistent with  $parms$ ,  $mpk$  and  $ID_b$ , and “invalid” otherwise

Operation: Use the following steps.

- Compute the identity element  $M = H_1(ID_b)$ .
- Compute  $T_0 = e(sk_{ID}, Q)$ .
- Compute  $T_1 = e(M, R)$ .

If  $T_0 = T_1$  then output the value “valid”, otherwise output the value “invalid.”

**8.2.3 Encryption**

The encrypt operation takes an arbitrary identity string  $ID_b$  in  $\{0,1\}^*$ , a message  $Msg_b$  of length  $\delta$  and the master-public key  $mpk = R$  with the system parameters  $parms$ , and outputs the ciphertext  $CT$  of the message.

The steps to encrypt the message are:

- Using  $R_1$ , compute a  $\delta$ -bit randomizer  $o$ .
- Compute the identity element  $M = H_1(ID_b)$ .
- Compute  $r = H_3(o, Msg_b)$ .
- Compute  $C_1 = rQ$ .
- Compute  $B = e(rM, R)$ .
- Compute  $C_2 = o \oplus H_2(B)$ .
- Compute  $C_3 = Msg_b \oplus H_4(o)$ .

h) Output  $CT = \langle C_1, C_2, C_3 \rangle$ .

### 8.2.4 Decryption

The decrypt operation takes a ciphertext  $CT$  computed for identity  $ID$ , and the private key  $sk_{ID}$  that corresponds to  $ID$ , and outputs the message  $Msg_b$  or “error”.

The steps to recover the message are:

- a) Parse the ciphertext  $CT$  as a tuple  $\langle C_1, C_2, C_3 \rangle$ .
- b) Compute  $B = e(sk_{ID}, C_1)$ .
- c) Compute  $o = C_2 \oplus H_2(B)$ .
- d) Compute  $Msg_b = C_3 \oplus H_4(o)$ .
- e) Compute  $r = H_3(o, Msg_b)$ .
- f) Test if  $C_1 = rQ$ . If not, output “error” and stop.
- g) Output  $Msg_b$ .

## 9 Identity-based hybrid encryption mechanisms

### 9.1 General

**A1** In this clause, three identity-based key encapsulation mechanisms are specified. These mechanisms use the following primitives. **A1** Each mechanism is specified to use the following primitives.

Take a security parameter  $\kappa$ , and a key size parameter  $\delta$ , the mechanism requires the following algorithms:

- a) A random bit generation algorithm:
  - $R_1$ : a source of random values in the space  $\{0, 1\}^\delta$ .
- b) **A1** Four **A1** hash functions:
  - $H_1: \{0, 1\}^* \rightarrow Z_p$  where  $H_1(s) = IHF1(s, p, \kappa)$
  - $H_2: G_3 \rightarrow \{0, 1\}^\delta$  where  $H_2(x) = SHF1(OS2BSP(z), \delta, \kappa)$  and  $z = FE2OSP(x)$
  - $H_3: \{0, 1\}^* \rightarrow \{0, 1\}^\delta$  where  $H_3(s) = SHF1(s, \delta, \kappa)$
  - A1** —  $H_4: \{0, 1\}^* \rightarrow Z_p^*$  where  $H_4(s) = IHF2(0x01 || s || 0x03, p, \kappa)$  **A1**

### 9.2 The SK key encapsulation mechanism

#### 9.2.1 Set up

The setup operation creates public system parameters and a master-secret key. The master-secret key shall be secured by the private key issuer, as all private keys calculated within the system depend on it. It is recommended that applications establish a methodology for changing the master-secret key, the corresponding master-public key and system parameters on a regular basis, and have a methodology for handling the disclosure of a master-secret key. However this is out of scope of this part of ISO/IEC 18033.



**ISO/IEC 18033-5:2015+A1:2021(E)**

The steps to setup the private key issuer and system parameters are:

- a) Establish the set of base groups  $G_1, G_2, G_3$  and a pairing  $e: G_1 \times G_2 \rightarrow G_3$ . The order of each group is  $p$ .
- b) Select a random generator  $Q_1$  in  $G_1$  and a random generator  $Q_2$  in  $G_2$ .
- c) Generate a random master secret  $s$  in  $Z_p^*$ . Calculate the corresponding  $R$  as  $sQ_1$ .
- d) Pre-calculate the pairing value  $J = e(Q_1, Q_2)$ .
- e) Make the system parameters and the master-public key set  $parms = \langle J, Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$  and  $mpk = R$  available. Secure the master-secret key  $s$ .

ISO/IEC 15946-1 contains recommendations for the generation of the groups, group generators and pairing.

**9.2.2 Private key extraction**

The extract operation takes an arbitrary identity string  $ID_b$  in  $\{0, 1\}^*$  and calculates the corresponding private key  $sk_{ID}$  in  $G_2$ . The algorithm to compute the private key  $sk_{ID}$  corresponding to an identity string  $ID_b$  is as follows:

Input:

- The system parameters  $parms = \langle J, Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$
- The master-public key  $mpk = R$
- The master-secret key  $msk = s$
- An identity string  $ID_b$

Output:

- The derived private key  $sk_{ID}$  that is an element of  $G_2$ .

Operation: Use the following steps to compute  $sk_{ID}$ .

- a) Compute the identity element  $M = H_1(ID_b)$ .
- b) If  $M + s \equiv 0 \pmod{p}$ , output “error” and stop.
- c) Compute  $t = (M + s)^{-1} \pmod{p}$ .
- d) Compute  $sk_{ID} = tQ_2$ .
- e) Output  $sk_{ID}$ .

The correctness of the value  $sk_{ID}$  can be verified by using the following algorithm:

Input:

- The system parameters  $parms = \langle J, Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$
- The master-public key  $mpk = R$
- An identity string  $ID_b$
- The corresponding private key  $sk_{ID}$

Output:

— The value “valid” if  $sk_{ID}$  is consistent with  $parms$ ,  $mpk$  and  $ID_b$ , and “invalid” otherwise

Operation: Use the following steps.

- a) Compute the identity element  $M = H_1(ID_b)$ .
- b) Compute  $T = e(MQ_1 + R, sk_{ID})$ .
- c) If  $T = J$  then output the value “valid”, otherwise output the value “invalid.”

### 9.2.3 Session key encapsulation

The encapsulate operation takes an arbitrary identity string  $ID_b$  in  $\{0,1\}^*$  and the master-public key  $mpk = R$  with the system parameters  $parms$ , and outputs the pair  $\langle K, CT_{KEM} \rangle$ , where  $K$  is a session key to be used to encrypt a message, and  $CT_{KEM}$  is the encapsulation of  $K$  to be transmitted to the receiver.

The steps to compute the encapsulation values are:

- a) Using the random number generator  $R_1$ , generate a random  $\delta$ -bit message  $m$ .
- b) Compute  $r = H_1(m)$ .
- c) Compute  $E = r(MQ_1 + R)$ .
- d) Compute  $B = H_2(J^r)$ .
- e) Set  $CT_{KEM} = \langle E, B \oplus m \rangle$ .
- f) Compute  $K = H_3(m)$ .
- g) Output  $\langle K, CT_{KEM} \rangle$ .

### 9.2.4 Session key de-encapsulation

The de-encapsulate operation takes an encapsulated value  $CT_{KEM}$  computed for identity  $ID$ , and the private key  $sk_{ID}$  that corresponds to  $ID$ , and computes the key value  $K$  that can be used to decrypt the message that was encrypted by the sender.

The steps to compute the de-encapsulated key are:

- a) Parse the encapsulated value  $CT_{KEM}$  as the pair  $\langle U, V \rangle$ .
- b) Compute  $B = H_2(e(U, sk_{ID}))$ .
- c) Compute  $m = B \oplus V$ .
- d) Compute  $r = H_1(m)$ .
- e) Compute  $Q = R + (H_1(ID_b) \cdot Q_1)$ .
- f) Verify that  $U = rQ$ . If not, output “error” and stop.
- g) Output  $K = H_3(m)$ .

### 9.3 The BB1 key encapsulation mechanism

#### 9.3.1 Set up

The steps to setup the private key issuer and system parameters are:

- Establish the set of base groups  $G_1, G_2, G_3$  and a pairing  $e: G_1 \times G_2 \rightarrow G_3$ . The order of each group is  $p$ .
- Select a random generator  $Q_1$  in  $G_1$  and a random generator  $Q_2$  in  $G_2$ .
- Generate a random master secret  $s_1, s_2$  and  $s_3$  in  $Z_p^*$ . Calculate the corresponding  $R$  as  $s_1Q_1$  and  $T$  as  $s_3Q_1$ .
- Pre-calculate the pairing value  $J = e(s_1Q_1, s_2Q_2)$ .
- Make the public parameters and the master-public key set  $parms = \langle Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$  and  $mpk = \langle J, R, T \rangle$  available. Secure the master-secret key  $\langle s_1, s_2, s_3 \rangle$ .

ISO/IEC 15946-1 contains recommendations for the generation of the groups, group generators and pairing.

#### 9.3.2 Private key extraction

The extract operation takes an arbitrary identity string  $ID_b$  in  $\{0, 1\}^*$  and calculates the corresponding private key elements  $d_{ID,0}$  and  $d_{ID,1}$  in  $G_2$ . The algorithm to compute the private key  $sk_{ID} = \langle d_{ID,0}, d_{ID,1} \rangle$  corresponding to an identity string  $ID_b$  is as follows:

Input:

- The system parameters  $parms = \langle Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$
- The master-public key  $mpk = \langle J, R, T \rangle$
- The master-secret key  $msk = \langle s_1, s_2, s_3 \rangle$
- An identity string  $ID_b$

Output:

- The derived private key  $sk_{ID}$  that is a pair of elements of  $G_2$ .

Operation: Use the following steps to compute  $sk_{ID}$ .

- Compute the identity element  $M = H_1(ID_b)$ .
- Select random integer  $r$  in  $Z_p^*$ .
- Compute  $t = s_1s_2 + r(s_1M + s_3)$  in  $Z_p$ . Return to (b) if  $t = 0$ , else move to (d).
- Compute  $d_{ID,0} = tQ_2$ .
- Compute  $d_{ID,1} = rQ_2$ .
- Output  $sk_{ID} = \langle d_{ID,0}, d_{ID,1} \rangle$ .

The correctness of  $sk_{ID}$  can be verified by using the following algorithm:

Input:

- The system parameters  $parms = \langle Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$
- The master-public key  $mpk = \langle J, R, T \rangle$
- An identity string  $ID_b$
- The corresponding private key  $sk_{ID} = \langle d_{ID,0}, d_{ID,1} \rangle$

Output:

- The value “valid” if  $sk_{ID}$  is consistent with  $parms$ ,  $mpk$  and  $ID_b$ , and “invalid” otherwise

Operation: Use the following steps.

- a) Compute the identity element  $M = H_1(ID_b)$ .
- b) Compute  $T_0 = e(Q_1, d_{ID,0})$ .
- c) Compute  $T_1 = e(MR + T, d_{ID,1})$ .
- d) If  $T_0 = T_1 J$  then output the value “valid”, otherwise output the value “invalid.”

### 9.3.3 Session key encapsulation

The encapsulate operation takes an arbitrary identity string  $ID_b$  in  $\{0,1\}^*$  and the master-public key  $mpk = \langle J, R, T \rangle$  with the system parameters  $parms$ , and outputs the pair  $\langle K, CT_{KEM} \rangle$ , where  $K$  is a session key to be used to encrypt a message, and  $CT_{KEM}$  is the encapsulation of  $K$  to be transmitted to the receiver.

The steps to compute the encapsulation values are:

- a) Compute the identity element  $M = H_1(ID_b)$ .
- b) Select random integer  $r'$  in  $Z_p^*$ .
- c) Compute  $E_0 = r'Q_1$ .
- d) Compute  $E_1 = (r'M)R + r'T$ .
- e) Compute  $B = J^{r'}$ .
- f) Set  $CT_{KEM} = \langle E_0, E_1 \rangle$ .
- g) Compute  $K = H_2(B)$ .
- h) Output  $\langle K, CT_{KEM} \rangle$ .

### 9.3.4 Session key de-encapsulation

The de-encapsulate operation takes an encapsulated value  $CT_{KEM}$  computed for identity  $ID$ , and the private key  $sk_{ID}$  that corresponds to  $ID$ , and computes the key value  $K$  that can be used to decrypt the message that was encrypted by the sender.

The steps to compute the de-encapsulated key are:

- a) Parse the encapsulated value  $CT_{KEM}$  as the pair  $\langle U, V \rangle$ .

- b) Compute  $B = e(U, d_{ID,0}) / e(V, d_{ID,1})$ .
- c) Compute  $K = H_2(B)$ .
- d) Output  $K$ .

## **A1** 9.4 The SM9 key encapsulation mechanism

### 9.4.1 Set up

The setup operation creates public system parameters and a master-secret key. This operation shall be completed by the private key issuer, an entity which shall be trusted by its subscribers.

The steps to create public system parameters and a master-secret key are:

- a) Establish the set of base groups  $G_1, G_2, G_3$ , and a pairing  $e: G_1 \times G_2 \rightarrow G_3$ . The order of each group is  $p$ .
- b) Select a random generator  $Q_1$  in  $G_1$  and a random generator  $Q_2$  in  $G_2$ .
- c) Generate a random master secret  $s$  in  $Z_p^*$ . Calculate the corresponding  $R$  as  $sQ_1$ .
- d) Pre-calculate the pairing value  $J = e(R, Q_2)$ .
- e) Make the system parameters and the master-public key set  $params = \langle J, Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$  and  $mpk = R$  available. Secure the master-secret key  $msk = s$ .

### 9.4.2 Private key extraction

The extract operation takes an arbitrary identity string  $ID_b$  in  $\{0,1\}^*$  and calculates the corresponding private key  $sk_{ID}$  in  $G_2$ . The algorithm to compute the private key  $sk_{ID}$  corresponding to an identity string  $ID_b$  is as follows:

Input:

- The system parameters  $params = \langle J, Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$
- The master-public key  $mpk = R$
- The master-secret key  $msk = s$
- An identity string  $ID_b$

Output:

- The derived private key  $sk_{ID}$ , an element of  $G_2$ .

Operation: Use the following steps to compute  $sk_{ID}$ .

- a) Compute  $M = H_4(ID_b)$ .
- b) If  $M + s = 0 \bmod p$ , output "error" and stop.
- c) Compute  $t = (M + s)^{-1} s \bmod p$ .
- d) Compute  $sk_{ID} = tQ_2$ .
- e) Output  $sk_{ID}$ .

The correctness of the value  $sk_{ID}$  can be verified by using the following algorithm:

Input:

- The system parameters  $params = \langle J, Q_1, Q_2, G_1, G_2, G_3, e, p \rangle$
- The master-public key  $mpk = R$
- An identity string  $ID_b$
- The corresponding private key  $sk_{ID}$

Output:

- The value "valid" if  $sk_{ID}$  is consistent with  $params$ ,  $msk$  and  $ID_b$ , and "invalid" otherwise.

Operation: Use the following steps.

- a) Compute  $M = H_4(ID_b)$ .
- b) Compute  $T = e(MQ_1 + R, sk_{ID})$ .
- c) If  $T = J$ , then output the value "valid", otherwise output the value "invalid".

#### 9.4.3 Session key encapsulation

The encapsulate operation ( $KEM.Enc$ ) takes an arbitrary identity string  $ID_b$  in  $\{0,1\}^*$  and the master-public key  $mpk = R$  with the system parameters  $params$ , and outputs the pair  $\langle K, CT_{KEM} \rangle$  where  $K$  is a session key to be used to encrypt a message, and  $CT_{KEM}$  is the encapsulation of  $K$  to be transmitted to the receiver.


The steps to compute the encapsulation values are:

- a) Select a random integer  $r$  in  $Z_p^*$ .
- b) Compute  $M = H_4(ID_b)$ .
- c) Compute  $E = r(MQ_1 + R)$ .
- d) Compute  $B = J^r$ .
- e) Compute  $K = KDF2-a(EC2OSP(E) || FE2OSP(B) || ID_b, klen)$ , where  $klen$  is the bit-length of the required session key.
- f) Set  $CT_{KEM} = EC2OSP(E)$ .
- g) Output  $\langle K, CT_{KEM} \rangle$ .

#### 9.4.4 Session key de-encapsulation

The de-encapsulate operation ( $KEM.Dec$ ) takes an encapsulated value  $CT_{KEM}$  computed for identity  $ID_b$  and the private  $sk_{ID}$  that corresponds to  $ID_b$ , and computes the key value  $K$  that can be used to decrypt the message that was encrypted by the sender.

The steps to compute the de-encapsulation key are:

- a) Parse  $CT_{KEM}$  as an element  $E = OS2ECP(CT_{KEM})$ .
- b) Check whether  $E$  is in  $G_1$ ; if not, output "error".
- c) Compute  $B = e(E, sk_{ID})$ .
- d) Compute  $K = KDF2-a(EC2OSP(E) || FE2OSP(B) || ID_b, klen)$ , where  $klen$  is the bit-length of the required session key.
- e) Output  $K$ . 

## Annex A (normative)

### Object identifiers

This annex lists the object identifiers assigned to mechanisms specified in this part of ISO/IEC 18033 and defines algorithm parameter structures.

```

EncryptionAlgorithms-5 {
    iso(1) standard(0) encryption-algorithms(18033) part5(5)
        asn1-module(0) algorithm-object-identifiers(0)
}

DEFINITIONS EXPLICIT TAGS ::= BEGIN

IMPORTS
    HashFunction
        FROM EncryptionAlgorithms-2 {
            iso(1) standard(0) encryption-algorithms(18033) part2(2)
                asn1-module(0) algorithm-object-identifiers(0) };

-- ID-based cipher mechanism identifiers --

OID ::= OBJECT IDENTIFIER -- Alias --

is18033-5 OID ::= { iso(1) standard(0) is18033(18033) part5(5) }

ib-enc OID ::= { is18033-5 ib-encryption-mechanism(1) }
ib-kem OID ::= { is18033-5 ib-key-encapsulation-mechanism(2) }

ib-enc-mechanism-bf OID ::= { ib-enc bf(1) }

[A1] ib-enc-mechanism-sm9a OID ::= { ib-enc sm9a(2) } -- sm9 kem with DEM2 as in 7.4.1
ib-enc-mechanism-sm9b OID ::= { ib-enc sm9b(3) } -- sm9 kem with DEM3 as in 7.4.1 [A1]

ib-kem-mechanism-sk OID ::= { ib-kem sk(1) }
ib-kem-mechanism-bb1 OID ::= { ib-kem bb1(2) }

[A1] ib-kem-mechanism-sm9 OID ::= { ib-kem sm9(3) }
sm9-dem-one-time-mac OID ::= { ib-kem-mechanism-sm9 one-time-mac(1) } [A1]

-- Identity-based encryption mechanisms --
IbEncMechanismIdentifier ::=
    AlgorithmIdentifier {{ IbEncMechanism }}

IbEncMechanism ALGORITHM ::= {
    { OID ib-enc-mechanism-bf PARMS HashFunction },
    [A1] { OID ib-enc-mechanism-sm9a PARMS HashFunction }
    { OID ib-enc-mechanism-sm9b PARMS HashFunction } [A1]
    ... -- Expect additional IB-ENC mechanisms --
}

-- Identity-based key-encapsulation mechanisms --

IbKemMechanismIdentifier ::=
    AlgorithmIdentifier {{ IbKemMechanism }}

```

```

IbKemMechanism ALGORITHM ::= {
    { OID ib-kem-mechanism-sk PARMS HashFunction } |
    { OID ib-kem-mechanism-bb1 PARMS HashFunction },
    A1 | { OID ib-kem-mechanism-sm9 PARMS HashFunction } A1
    ... -- Expect additional IB-KEM mechanisms --
}
AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
    algorithm ALGORITHM.&id({IOSet}),
    parameters ALGORITHM.&Type({IOSet}){@algorithm} OPTIONAL
}

ALGORITHM ::= CLASS {
    &id OBJECT IDENTIFIER UNIQUE,
    &Type OPTIONAL
}
    WITH SYNTAX { OID &id [PARMS &Type] }

END -- EncryptionAlgorithms-5 --

```



**Annex B**  
**(informative)**

**Security considerations**

All the mechanisms specified in this part of ISO/IEC 18033 satisfy the property known as *chosen ciphertext security*. This security notion is widely regarded as one of the strongest security notions for a public key encryption mechanism. <sup>[A1]</sup> Security analyses of the BF, SK, BB1 and SM9 mechanisms can be found in References [\[4\]](#), [\[5\]](#), [\[3\]](#) and [\[9\]](#), respectively. <sup>[A1]</sup>

## Annex C (informative)

### Numerical examples

In this annex, unless otherwise explicitly stated, all the values are expressed in hexadecimal notation.

#### C.1 BF ID-based encryption mechanism

##### C.1.1 Example 1

###### C.1.1.1 Setup

This example makes use of the same elliptic curve  $y^2 = x^3 + 1$  used in ISO/IEC 14888-3:2006, F.7.1. An element in  $GF(q^2)$  is represented as  $a + b\sigma$  where  $a$  and  $b$  are elements in  $GF(q)$  and  $\sigma$  is an element in  $GF(q^2)$  satisfying  $\sigma^2 + 1 \equiv 0 \pmod{q}$ . The distortion map used to convert a point in one torsion group to an element of other torsion group to achieve non-degeneracy of pairings works as follows:  $\varphi(Q) = \varphi(Q_x, Q_y) = (\beta \cdot Q_x, Q_y)$  where  $\beta \neq 1$  is an element in  $GF(q^2)$  satisfying  $\beta^3 - 1 \equiv 0 \pmod{q}$ . Bilinear pairing  $e(P, Q)$  is implemented as the Weil pairing on two input points  $P$  and  $\varphi(Q)$ .

$H_1$  is implemented as follows:

Let  $q$  be a prime satisfying  $q \equiv 2 \pmod{3}$  and  $q = cp - 1$  for some integer  $c$  and prime  $p > 3$ . Let  $E(GF(q))$  be the elliptic curve  $y^2 = x^3 + 1$  and let  $G_1$  be  $E(GF(q)) [p]$ . Let bit string  $str$  be the input to  $H_1$ . Then, compute the output  $P_{str} \in G_1$  in the following steps:

- 1) Let  $y_0 = IHF1(str, q, \kappa)$
- 2) Let  $x_0 = (y_0^2 - 1)^{(2q-1)/3} \pmod{q}$
- 3) Let  $J = (x_0, y_0) \in G_1$
- 4) Let  $P_{str} = cJ$

```

q  = b35fa5fd e47falab bble57e9 3balff96 38b89b99 5c49be81 a38e3194 a0983816
    4ee51fb9 1d285832 f9a05d63 9c8d9680 10c93a35 27e561f2 fd6a45cc 70aba1fb

p  = 80000000 000fffff ffffffff ffffffff ffffffff

Qx = 7ee6f118 9329adb4 1e8cd405 2295f1a7 6096631d f065dd38 85fff26b 8ed52022 7bf-
    c3f3d 07fe9cfe 093424ba 9dbd4c0d 73fff367 3ce2c922 f0b73c50 2992093b

Qy = 67c855c9 e6b617be b24b792a e9c3e21e 95f37006 25b91058 3bc0b293 c36a762c
    feba4266 038989cc 59797235 c6116d99 8a97b805 ff82c664 53720e5c 1de95e3e

s  = 56e84bdc e3d76405 33612345 d6bf0725 fb7d0391

Rx = 2141437a ca95b5d1 15a11812 e779233f 2b6470d2 4678738e 95960498 10e5147a
    b75daceb 299b1b4c cacfe4df 597b3d0d cb488876 36477353 562cfb8c 56bf6a6d

Ry = 1c101269 0b92fd0f 2a89a3ea 10037022 ac696e28 50123794 e0b218d3 53214505
    1f669aa4 4d363612 1b54bc4f 0166e667 9a3797e8 25aca54c fbc62c7e 3e881b86

k  = 256 (decimal)

```

**ISO/IEC 18033-5:2015+A1:2021(E)****C.1.1.2 Private key extraction**

```

ID =    sc27wg2-secretary@ipa.go.jp

IDb =    73633237 7767322d 73656372 65746172 79406970 612e676f 2e6a70

Mx =    22223f9c 4aeaf934 723d739b 6e6ea3ea ea6d1aaa b3581de8 bff77695 5a6d4327
        175b44d5 0ea32c59 26fa3a1c 028ebad1 fc1b6988 701be579 9b3d6a01 b3e85c67

My =    073113c6 6fb31cb0 1c12b0f0 18ca7993 428a309c 7f6f54d5 d51651af 4ad2c829
        6e5ac712 9854a077 8d8f3576 f5b56505 47fbd7bc a8c9aaa4 85a9a8a8 58175395

skIDx= 775fb86b a773f24a a32d569b aac7d4f8 2c1d99ae 9432eed5 df746842 588d053d
        6b195f49 55dbcd64 84424dc8 fa6f9d6e c44ea1e4 84ed9190 fdd38fea 959eb706

skIDy= 8fba3ae1 140e919f b19b40ed c0dfe6de b47f8e9b b87e86d3 e1b12ee3 9c73489a
        95e17dc3 9bb26224 8373a5e9 6dabb9f8 ba4d1ca7 884320ad 2f35c6bb a30dc698

T0.a= 4c1e2713 096f15b9 030f00f9 2bcded161 dce0fd0c 66ff94d7 c57e2c85 87037e3b
        8c1ed769 2f96d79d 174ca761 552ed0f7 fb5af4a1 5d987bb7 53d169aa a36edecb

T0.b= aede1726 13721ee2 4b54800e f8339e71 075d6fb0 4e68294e f066937c 71f5d69d
        cb05d4f6 eb2e5d2a bcfcf174 251acfe3 a85aac0a 460b8836 252d1cc1 f81d8a53

T1.a= 4c1e2713 096f15b9 030f00f9 2bcded161 dce0fd0c 66ff94d7 c57e2c85 87037e3b
        8c1ed769 2f96d79d 174ca761 552ed0f7 fb5af4a1 5d987bb7 53d169aa a36edecb

T1.b= aede1726 13721ee2 4b54800e f8339e71 075d6fb0 4e68294e f066937c 71f5d69d
        cb05d4f6 eb2e5d2a bcfcf174 251acfe3 a85aac0a 460b8836 252d1cc1 f81d8a53

```

Private key skID is valid.

**C.1.1.3 Encryption**

Message length: 896 bits

```

Msgb= 01234567 89abcdef fedcba98 76543210 00112233 44556677 8899aabb ccddeeff
        ffeeddccb bbbaa9988 77665544 33221100 00001111 22223333 44445555 66667777
        88889999 aaaabbbb ccccdddd eeeeefff ffffefeee ddddcccc bbbbbaaaa 99998888
        77776666 55554444 33332222 11110000

o =    32733aa5 cc085739 74f52a96 ec181a31 97f8fe02 95525b0b 4a69970f 7b631732
        0e96b2e1 044c43ad e3b8c68f 3c59d046 f852e0b9 f35e892a cb837a85 ce4ae9e1
        26ae3344 a1552cf3 8e523462 05f7e4ab e2aa88a5 41d5f886 9bce6c6f 3d8874bb
        bf845027 ef68509c fd7f64bd 7801b627

r =    647ed607 923a6589 a90c3b48 40a1e4e7 599d86a7

Clx = 96e2f727 7af63756 cb34e4e5 13b6fee9 9be157e9 0bc69ab8 cf08902a 5f5e3fdb
        a0d57c4e 09ab41b0 ece23b70 86e51ea0 0f10c7c5 d96b7a63 e6676c0a faalb5ea

Cly = a8ab6927 c7c9a15f 568d8b4c 2e30b431 c46629a6 7ea67551 a5809192 ac970935
        d5882644 0f870256 bc1ee3a3 8559ad9f f60ba485 80c22a8e 892bf3de b738cf14

rMx = 98013221 69e44256 bec7edd8 347dbf84 e245ba0e 4a745eb2 85834b66 45eab3b5
        d9bddb0f a7ace4b5 8d244f14 d61b6e2c c4f41b79 2a1c5911 a31107e2 ab598c00

rMy = 69f22e4f b551efa8 26521a1e aabaa1e9 f152c847 7caba380 fe3fcbd2 943fe0ae
        cd6d3c29 ea2a9073 11a5ce17 f6aa86a0 e75ff57b 129bc971 e70a5c02 35bd1529

B.a = 5888c21f f767d71a 0cbad876 f96afab8 2a7b6fbe c00fbb9b celbaaec 5aaedcc2
        928f8ad0 0773746d 063d4b94 d40deee1 de33ea8f 03db24a6 972ee8bc laeaae28

B.b = 2558dcb1 707636c4 5006c3ca 75d57740 62f5ecd4 327cb97a e79e9566 89f1e6f7
        0f71fd1c 68614416 b277d756 f3f07109 6415d6c0 88cb221b c1757f8c 3c74e59f

```

Within H2:  $z = \text{FE2OSP}(B) = \text{I2OSP}(B.a + (B.b)q, 128)$

```

z   = 1a2b1681 47e1a03b 6192811f d8fe8e6f f04e0035 f2f3057d 2dcedb32 68e180d8
      4a82957c 056d4767 b917551a 91d7f071 bdba0b18 588125c7 130866b2 0fe00ab0
      f8e2c79b 4901d3c2 827a30e0 c3c7c9df eace492f b30f77dc 4488f2bd d53feb3b
      5e4f2cc3 c0bafa58 0327c7ef 5c122321 cfffd9e1 1f36fa80 9d9a857f d725d00d

C2  = ea671342 3816ba62 e4d08601 a9b292b9 7f4e49ac f65ca5ff f7fa6f41 de227ed9
      9f1bd16d 85c29f3f 15d2904e 2af8f5e7 74ac6996 61bb138b 74850241 9fab2cf5
      d589270e 5eca6437 c3970aca be44b202 ae70211d 6058e745 21c33892 520c9671
      eblc795c f77430d0 e23642ec 73d9d00e

C3  = 8368728c 9d4ae8b6 434a961b 494c25c7 36159933 3701e330 e64111a8 0829368e
      7e9c5a00 cd4789dd f4c3722b 76682199 a2cf7e1b 2cf49dae ce51cd74 2dc621f7
      9127bbec 5d44c2bd 9a6b6415 f4172ac8 1927932c 533e5962 dedc558b 32a3dbaa
      b8d1c25b 9c433a02 9492e100 66f4d9c8
  
```

Encryption completed.

### C.1.1.4 Decryption

```

B.a = 5888c21f f767d71a 0cbad876 f96afab8 2a7b6fbe c00fbb9b celbaaec 5aaedcc2
      928f8ad0 0773746d 063d4b94 d40deee1 de33ea8f 03db24a6 972ee8bc laeaae28

B.b = 2558dcb1 707636c4 5006c3ca 75d57740 62f5ecd4 327cb97a e79e9566 89fle6f7
      0f71fd1c 68614416 b277d756 f3f07109 6415d6c0 88cb221b c1757f8c 3c74e59f

o   = 32733aa5 cc085739 74f52a96 ec181a31 97f8fe02 95525b0b 4a69970f 7b631732
      0e96b2e1 044c43ad e3b8c68f 3c59d046 f852e0b9 f35e892a cb837a85 ce4ae9e1
      26ae3344 a1552cf3 8e523462 05f7e4ab e2aa88a5 41d5f886 9bce6c6f 3d8874bb
      bf845027 ef68509c fd7f64bd 7801b627

r   = 647ed607 923a6589 a90c3b48 40a1e4e7 599d86a7

Msgb= 01234567 89abcdef fedcba98 76543210 00112233 44556677 8899aabb ccddeeff
      ffeeddcc bbba9988 77665544 33221100 00001111 22223333 44445555 66667777
      88889999 aaaabbbb ccccdddd eeeeffff fffffeee ddddcccc bbbbaaaa 99998888
      77776666 55554444 33332222 11110000
  
```

Decryption successful.

## C.1.2 Example 2

### C.1.2.1 Setup

This example makes use of the same elliptic curve  $y^2 = x^3 + 1$  and  $H_1$  as in [C.1.1](#). Bilinear pairing  $e(P, Q)$  is implemented as the Reduced Tate pairing on two input points  $P$  and  $\varphi(Q)$ .

```

q   = b35fa5fd e47fa1ab bb1e57e9 3ba1ff96 38b89b99 5c49be81 a38e3194 a0983816
      4ee51fb9 1d285832 f9a05d63 9c8d9680 10c93a35 27e561f2 fd6a45cc 70abafbf

p   = 80000000 000fffff ffffffff ffffffff ffffffff

Qx  = 7ee6f118 9329adb4 1e8cd405 2295f1a7 6096631d f065dd38 85fff26b 8ed52022
      7bfc3f3d 07fe9cfe 093424ba 9dbd4c0d 73fff367 3ce2c922 f0b73c50 2992093b

Qy  = 67c855c9 e6b617be b24b792a e9c3e21e 95f37006 25b91058 3bc0b293 c36a762c
      feba4266 038989cc 59797235 c6116d99 8a97b805 ff82c664 53720e5c 1de95e3e

s   = 56e84bdc e3d76405 33612345 d6bf0725 fb7d0391

Rx  = 2141437a ca95b5d1 15a11812 e779233f 2b6470d2 4678738e 95960498 10e5147a
      b75daceb 299b1b4c cacfe4df 597b3d0d cb488876 36477353 562cfb8c 56bf6a6d
  
```

**ISO/IEC 18033-5:2015+A1:2021(E)**

Ry = 1c101269 0b92fd0f 2a89a3ea 10037022 ac696e28 50123794 e0b218d3 53214505  
1f669aa4 4d363612 1b54bc4f 0166e667 9a3797e8 25aca54c fbc62c7e 3e881b86

k = 256 (decimal)

**C.1.2.2 Private key extraction**

ID = sc27wg2-secretary@ipa.go.jp

IDb = 73633237 7767322d 73656372 65746172 79406970 612e676f 2e6a70

Mx = 22223f9c 4aeaf934 723d739b 6e6ea3ea ea6d1aaa b3581de8 bff77695 5a6d4327  
175b44d5 0ea32c59 26fa3a1c 028ebad1 fc1b6988 701be579 9b3d6a01 b3e85c67

My = 073113c6 6fb31cb0 1c12b0f0 18ca7993 428a309c 7f6f54d5 d51651af 4ad2c829  
6e5ac712 9854a077 8d8f3576 f5b56505 47fbd7bc a8c9aaa4 85a9a8a8 58175395

skIDx= 775fb86b a773f24a a32d569b aac7d4f8 2c1d99ae 9432eed5 df746842 588d053d  
6b195f49 55dbcd64 84424dc8 fa6f9d6e c44ea1e4 84ed9190 fdd38fea 959eb706

skIDy= 8fba3ae1 140e919f b19b40ed c0dfe6de b47f8e9b b87e86d3 e1b12ee3 9c73489a  
95e17dc3 9bb26224 8373a5e9 6dabb9f8 ba4d1ca7 884320ad 2f35c6bb a30dc698

T0.a= 124f0489 e46f7973 0a223780 0ceaf462 943f38b7 00d8ccc0 9cc3bc2d 61283841  
2fc946d4 3a4762b1 0f7e9388 35c7008a d3e5e85e edfd8d74 11b5ca21 67fb1ad0

T0.b= 3374b3b6 f2547afc 0110f8e4 f4e2e852 d6239287 ae277fc5 44f088b8 905fd5b9  
aa65fb79 3c96d719 0b1f7c29 f2801344 6f527d87 222cdcec de6e8755 5c17d719

T1.a= 124f0489 e46f7973 0a223780 0ceaf462 943f38b7 00d8ccc0 9cc3bc2d 61283841  
2fc946d4 3a4762b1 0f7e9388 35c7008a d3e5e85e edfd8d74 11b5ca21 67fb1ad0

T1.b= 3374b3b6 f2547afc 0110f8e4 f4e2e852 d6239287 ae277fc5 44f088b8 905fd5b9  
aa65fb79 3c96d719 0b1f7c29 f2801344 6f527d87 222cdcec de6e8755 5c17d719

Private key skID is valid.

**C.1.2.3 Encryption**

Message length: 896 bits

Msgb= 01234567 89abcdef fedcba98 76543210 00112233 44556677 8899aabb ccddeeff  
ffeeddcc bbbaa998 77665544 33221100 00001111 22223333 44445555 66667777  
88889999 aaaabbbb ccccdddd eeeeffff ffffefee ddddcccc bbbbaaaa 99998888  
77776666 55554444 33332222 11110000

o = 2ef53f26 046cd174 81063fd8 1fcf7ee6 9009b433 b31f0c69 12a57558 11a25d1e  
63ee9a33 8da3f2fd 99a04c76 d53c444f 2ef28fb9 44dfdcbl 0edcb29b 86dc2fa3  
bb0938f0 70109f78 08ae5e34 9cf3e92f 36709fda d10c5049 f9d0d2c4 fd382266  
ac748d2b 4d02151d c7d1e4c3 5763445f

r = 5bbf2159 d859b329 cade07e3 8f9bfe51 3a41b396

Clx = 023debf1 52cbe5cd efadbd31 65765e0c 627ba725 00fcf66f 008c9aa2 655e3c09  
241e2655 c50e677f 31162635 0cc84cce cdd3777f 91905f8f c0126061 785208f3

Cly = 4df18cfe ec316138 57a4aeba d1c8a2e2 6f809f27 02f30dd9 09070bfd 39ba19d6  
b092560d 9bfd967d 30ff6740 3af0520b 10eb138c 0c634c4a 93e377aa cab53fe0

rMx = 5abbbf3f2 5fac96f5 c75d3795 0fa58a2f 8db666e9 1146b2d8 aad3c26c 0f2c408b  
9e485d95 e7308ee8 adf9bd22 a496c60e 16909aeb 48d7e9ca cae7d420 c7091b42

rMy = 0abfbc77 468e16fa 3c6045e4 6d385487 65278b9a 2a6545ad 6665c146 9a69a2c1  
95ac55a4 51cc879a 17ae67ed 8569a72a 048c14fe 6e0b124a 7396c99b 05d9c8fa

B.a = a5ca565a b0b9e928 d1cda8da 25368ea5 1fcc27b3 3d9d429c 830f2b33 f850ba9c  
210d59a2 cc97d5b8 fb5aedbd 1a1255bb 48dcd725 67cd0633 296dcae1 2b28e881

B.b = 27845eb3 990ad135 668e6899 3564ed1a da6d61f4 b5f59c4a 212fbd06 5f85ea78  
9efd69a5 e1a0cf8a 94c72896 1393adcc fa949c5f lee5baa2 826d1c5c 06782ab1

Within H2:  $z = \text{FE2OSP}(B) = \text{I2OSP}(B.a + (B.b)q, 128)$

z = 1bb051f6 3ac94c65 e351f306 bcae1f21 79b5f4af eafed6d0 66227391 6f63a669  
3d7f443c 5b621e7b 3fd8cfd8 b1808de1 6f417b2d d79a637f f71dcf92 52dc503e  
526764c9 f0239a22 306137bb e66e65ff 51e21312 af7b634d fd87af60 6fa2c67e  
60220dae afb18021 5b880b7b 2797caad 183e32e3 c6395632 0319182b 0a0f150c

C2 = 8cc39c03 8facb65e 01e8970e 45de4fca 26486afb 2d2a7148 f09f733e e6db8ac2  
0c7004bf e0a5a6f7 8ecd91a5 a0d7e25e e34be7c0 2c1ec62c 328b9296 56ffb635  
3bf28bea 364085a3 b90b3305 a2acc06b 9ccff4d6 ed3e97e9 141820f6 a8d34c88  
4cb8df34 91079724 083db54e 4ba0de15

C3 = f998848b 2c735001 19985202 ec285a6b 22cdb577 e6959fa2 ac6058c6 0da4c5d1  
38035c1b aaa4e33a 950a467a c660af5a 053c5852 065906ba 39f842dc 6ee90807  
1b55497a 84c3cfb4 25a5b235 8c23712e 3c28c1de 200f9c9b d0bcldd1 e21a0fe0  
431718b7 58151e83 923151d7 9d982d61

Encryption completed.

#### C.1.2.4 Decryption

B.a = a5ca565a b0b9e928 d1cda8da 25368ea5 1fcc27b3 3d9d429c 830f2b33 f850ba9c  
210d59a2 cc97d5b8 fb5aedbd 1a1255bb 48dcd725 67cd0633 296dcae1 2b28e881

B.b = 27845eb3 990ad135 668e6899 3564ed1a da6d61f4 b5f59c4a 212fbd06 5f85ea78  
9efd69a5 e1a0cf8a 94c72896 1393adcc fa949c5f lee5baa2 826d1c5c 06782ab1

o = 2ef53f26 046cd174 81063fd8 1fcf7ee6 9009b433 b31f0c69 12a57558 11a25d1e  
63ee9a33 8da3f2fd 99a04c76 d53c444f 2ef28fb9 44dfdcb1 0edcb29b 86dc2fa3  
bb0938f0 70109f78 08ae5e34 9cf3e92f 36709fda d10c5049 f9d0d2c4 fd382266  
ac748d2b 4d02151d c7d1e4c3 5763445f

r = 5bbf2159 d859b329 cade07e3 8f9bfe51 3a41b396

Msgb= 01234567 89abcdef fedcba98 76543210 00112233 44556677 8899aabb ccddeeff  
ffeeddcc bbba9988 77665544 33221100 00001111 22223333 44445555 66667777  
88889999 aaaabbbb ccccdddd eeeeffff ffffefeee ddddcccc bbbbaaaa 99998888  
77776666 55554444 33332222 11110000

Decryption successful.

## C.2 SK ID-based key encapsulation mechanism

### C.2.1 Example 1

#### C.2.1.1 Setup

This example makes use of the same elliptic curve  $y^2 = x^3 + x$  used in ISO/IEC 14888-3:2006, F.7.2. An element in  $GF(q^2)$  is represented as  $a + b\sigma$  where  $a$  and  $b$  are elements in  $GF(q)$  and  $\sigma$  is an element in  $GF(q^2)$  satisfying  $\sigma^2 + 1 \equiv 0 \pmod{q}$ . The distortion map used to convert a point in one torsion group to an element of other torsion group to achieve non-degeneracy of pairings works as follows:

**ISO/IEC 18033-5:2015+A1:2021(E)**

$\varphi(Q) = \varphi(Q_x, Q_y) = (-Q_x, \sigma Q_y)$ . Bilinear pairing  $e(P, Q)$  is implemented as the Weil pairing on two input points  $P$  and  $\varphi(Q)$ .

```

q = 80000000 00000000 00000000 00000000 00020001 40000000 00000000 00000000
    00000000 00010000 80000002 00000000 00000000 00000000 00000000 00080003

p = 80000000 00000000 00000000 00000000 00020001

Q1x = 0db4e0f7 22dd090d a2b6d8fe adaf21d9 546ab265 1515af9b a87108f3 4e1ae0e3
      eb132c10 81452cc1 e52bb2a7 4287a0cb d8ff8dd9 3a225641 5321f0e4 c8892a50

Q1y = 762c096c 49f1ab04 7d7f37de 537a4e7c 2991c400 22e0c9a9 b3f58b1b 9df4f28a
      4a4330e2 170e14d2 f55a0719 8b667d0b 01e5a482 3f07e921 8516481e 641970ac

Q2x = 6b8f666b cf6b4672 d4634753 1f734e71 41bcd5fd 125f3ef3 714edc28 f6426900
      75ffb5f7 9e745cc0 fb03f940 3bdcefe8 acbe6286 d5d9955c 2a0e5ed7 657748c6

Q2y = 69584e47 f3070fed 9800d6cd e0f314b4 03955126 1c5bfef6 f3595f94 5958f7d9
      34dcdb3d 63125410 ccd363f8 02df1c7e 4a3d7ac7 24cf3865 0fb16ec1 7bb30a85

s = 37e28416 c95170c9 46ebfb30 3e422717 191b4b73

Rx = 76c25ec0 08bc4246 263d5078 c4846a7c a68469cf ed65108b bb5c5ce4 5d3597d6
      fe9a2b60 17a1f7db 2f7561ef fd2f72d9 0b7445b7 4d648674 8da13ac0 c34788bb

Ry = 4b913c48 0cc4d1c7 04e6d7ed 9424d4bc 2a532cd3 fbac55fe 6009d92f 39c2571b
      3bf0d363 29a6b810 8a0fc795 829a3cca 91472f29 5df5bdc6 219648b6 8aba737f

J.a = 56adfd18 df69d7ac c91f60a6 aa9620e2 482d9266 0d35fa24 ee29de58 2fb201eb
      5e4fd96e e646befc 9bd16125 265458cb 0043806e c4b3e2b6 297d5741 a8198d71

J.b = 65a05adc 1528b2bd 44508723 7d386249 cec79493 dc725dcd 54c08455 386ecce2
      dc85e2cb 0a31abc0 56f5eee2 4b603092 7750ff0e c40fff2d cec6357b 5fc9c1ef

k = 192

```

**C.2.1.2 Private key extraction**

```

ID = sc27wg2-secretary@ipa.go.jp

IDb = 73633237 7767322d 73656372 65746172 79406970 612e676f 2e6a70

M = 7de5a0a1 92c9b2b0 b3847381 72c16e9e 2cb63566

skIDx= 4a884819 c9a35eb0 c910573f 00e6c679 e08d2368 15e3fb95 1cbf882e acc10574
      08ad6342 e8922973 7f2493eb 80138666 a9dc5b61 afeb158c 43251271 f101e9cb

skIDy= 5015e072 1560366e 19a01a2c b4eecd98 8c725f55 e22af966 409b9fcf ed97860b
      499fa4ec 9ef4ae8a 482757ba 1a36035d d42fbed7 7b9668cf f318c4b5 22e7e354

T.a = 56adfd18 df69d7ac c91f60a6 aa9620e2 482d9266 0d35fa24 ee29de58 2fb201eb
      5e4fd96e e646befc 9bd16125 265458cb 0043806e c4b3e2b6 297d5741 a8198d71

T.b = 65a05adc 1528b2bd 44508723 7d386249 cec79493 dc725dcd 54c08455 386ecce2
      dc85e2cb 0a31abc0 56f5eee2 4b603092 7750ff0e c40fff2d cec6357b 5fc9c1ef

```

Private key skID is valid.

**C.2.1.3 Session key encapsulation**

```

m = 5973f2f0 3a6c618d ccf2d355 9625b18f e462e3d7 29b9d263 ce10272a f80946f3

r = 7d086f6b a788dc7e 6a90f9d5 b2e0cd17 3197e14e ef1e24b3 a18c10fe 9656375d
      97af8805 96a78069 b627d018 be0b4f20 4779a1e7 73b857f3 1d840f04 175279b2

```



```

Jr.a = 2ecb5c37 cc84ef22 3862fc40 d406c14e d8820fab ad8b299d 91b0c0b0 e33dce53
      242fa52c 4e59d298 c487b1cc 8d8e8f34 9148d7ec 7a271ee7 e2d4e990 143e5483

Jr.b = 310d2619 c98a845c 938d7b8d 645f5014 ef7fa880 640bbfbc 778f00c5 04a80e58
      bf3f07db bfa1f0f8 fe6a122d aaa09eaa 04f484a7 72a71345 930c7f22 0098a6df

Ex = 1542ebf9 39278252 1edd2691 28216053 6939e5fe ea0ec941 9ed5157b d0a1c0df
     3e9a2d1f 02559391 51948288 9127aed1 20b2667e de6690a8 b7ae80eb 8713fd35

Ey = 1a5210b3 15293e58 225e3efa fe3504b6 43d1f8c5 b5a526b1 e384ca64 7143381f
     1cf9eb3b a4665bae 88904cb1 918533c0 e1a119b2 438f5478 7ebb3ded 16e3b8e9

Within H2: z = FE2OSP(Or) = I2OSP(Or.a + (Or.b)q, 128)

z = 1886930c e4c5422e 49c6bdc6 b22fa80a 77c0365a bb89e293 806dccf1 3ldfaa5b
    bd408707 5c318441 ba5d0478 37893b8a bd66fd9f e179615e 44018212 336f7041
    ca01ef9a 5b546bb8 b61df054 c727d444 ca278fcb 08609def dae8da8f 3a7e1953
    cec12083 e24bea87 d2648b18 82c092d6 c561fb1b 6c48f11c 950a6bbb 4d004920

B = beeb76c3 b69c04e9 19870d88 47905d8b bf93f4fa dd49c84f 13ea36e8 16a2a8b4

B+m = e7988433 8cf06564 d575dedd d1b5ec04 5bf1172d f4f01a2c ddfallc2 eeabee47

K = 26551345 9d2de791 d7f4cdcf 578c202f 11542f58 30a0c5fa a6b2dab5 0ac0c7ff

Session key encapsulation completed.

```

### C.2.1.4 Decapsulation

```

B = beeb76c3 b69c04e9 19870d88 47905d8b bf93f4fa dd49c84f 13ea36e8 16a2a8b4

m = 5973f2f0 3a6c618d ccf2d355 9625b18f e462e3d7 29b9d263 ce10272a f80946f3

r = 7d086f6b a788dc7e 6a90f9d5 b2e0cd17 3197e14e ef1e24b3 a18c10fe 9656375d
    97af8805 96a78069 b627d018 be0b4f20 4779a1e7 73b857f3 1d840f04 175279b2

Qx = 1542ebf9 39278252 1edd2691 28216053 6939e5fe ea0ec941 9ed5157b d0a1c0df
     3e9a2d1f 02559391 51948288 9127aed1 20b2667e de6690a8 b7ae80eb 8713fd35

Qy = 1a5210b3 15293e58 225e3efa fe3504b6 43d1f8c5 b5a526b1 e384ca64 7143381f
     1cf9eb3b a4665bae 88904cb1 918533c0 e1a119b2 438f5478 7ebb3ded 16e3b8e9

K = 26551345 9d2de791 d7f4cdcf 578c202f 11542f58 30a0c5fa a6b2dab5 0ac0c7ff

Decapsulation successful.

```

## C.2.2 Example 2

### C.2.2.1 Setup

This example makes use of the same elliptic curve  $y^2 = x^3 + 1$ . The bilinear mapping function used in this example is the Reduced Tate pairing.

```

q = b35fa5fd e47fa1ab bb1e57e9 3ba1ff96 38b89b99 5c49be81 a38e3194 a0983816
    4ee51fb9 1d285832 f9a05d63 9c8d9680 10c93a35 27e561f2 fd6a45cc 70aba1fb

p = 80000000 000fffff ffffffff ffffffff ffffffff

Q1x = 7ee6f118 9329adb4 1e8cd405 2295f1a7 6096631d f065dd38 85fff26b 8ed52022
     7bfc3f3d 07fe9cfe 093424ba 9dbd4c0d 73fff367 3ce2c922 f0b73c50 2992093b

Q1y = 67c855c9 e6b617be b24b792a e9c3e21e 95f37006 25b91058 3bc0b293 c36a762c
     feba4266 038989cc 59797235 c6116d99 8a97b805 ff82c664 53720e5c 1de95e3e

```



**ISO/IEC 18033-5:2015+A1:2021(E)**

```

Q2x = 596aabeb e89571de 383450c5 4290a154 60bde0e2 a3a982cf f46596f9 9121e38d
      857d1362 4d445bf7 8ab3377d ecc122e3 2c3255f0 62369c8a 7d13d2d4 af91e143

Q2y = 4e89178d 16514938 ae638f70 76a4ca1f 6082ec4b 1c444785 6769e791 385e81da
      c8f7b174 8a2ce2fd c6080cf7 d6631e07 da57f5f9 adb85b7d 149f434e fe2dca40

s = 37e28416 c95170c9 46ebfb30 3e422717 191b4b73

Rx = 2b9e01f7 b1156ea3 32e9ed27 11ea9a28 4f4a1da6 e4ac08c0 2e7991f2 8bee9cbf
      c4b9c138 94a71aa8 5a79c016 680f0e16 180faf69 211705b1 da199e04 2c27b0cc

Ry = 0599b140 2272573a 7b8209fa a922a435 b46d4634 f6af4e81 ba810480 5120476b
      dalba5ba f0e475b5 96ce9d90 afc8eab2 46f290da d4e91f66 b98cdfc9 3bd321c5

J.a = 257694ae 4281deb6 58831a07 4ead48a3 6eb03bea 94063a8f d9b87386 a36ca231
      49abaa05 a8d3bd88 ead29fbb cdf7bd30 3aeelfa4 eae7690b a7380c0f 386f280d

J.b = ac6b84b8 1113130c 4316fcdc f1d40a9c 095a7739 2077518e 3a6b94b6 ee38079d
      1a6cf29b f904d909 54e3137c c47eb066 b39129da 57bfe26b 0c4497e2 f4e4a251

k = 192

```

**C.2.2.2 Private key extraction**

```

ID = sc27wg2-secretary@ipa.go.jp

IDb = 73633237 7767322d 73656372 65746172 79406970 612e676f 2e6a70

M = 305c1ac0 9f517694 0c6f4153 02fd22ff dd9214c9

skIDx= 0857e829 7f03bb71 ced24ee9 7b728d8c 9afdc7d6 c8a8b027 064a3208 0ad49829
      962f0f4f 1c5dac75 fd1f351d ebaee531 cbebceeb 79214bca 0041db9d c1175aad

skIDy= 3fa56ae5 67260a13 dc22a3fa c4853085 cab75f6c dd363363 eb5729a9 5e17dbcd
      39a504da 6fd2a27d f68a5570 4f4140ae d25dedb3 274e1bfe 44cbd823 3d981509

T.a = 257694ae 4281deb6 58831a07 4ead48a3 6eb03bea 94063a8f d9b87386 a36ca231
      49abaa05 a8d3bd88 ead29fbb cdf7bd30 3aeelfa4 eae7690b a7380c0f 386f280d

T.b = ac6b84b8 1113130c 4316fcdc f1d40a9c 095a7739 2077518e 3a6b94b6 ee38079d
      1a6cf29b f904d909 54e3137c c47eb066 b39129da 57bfe26b 0c4497e2 f4e4a251

```

Private key skID is valid.

**C.2.2.3 Session key encapsulation**

```

m = cb341a16 04f777b3 c58432b9 c937f1a1 2ce7d67c f05e6937 804d80b4 7b636131

r = 54d92604 6c850c8b f0a95b7d 71e49ff6 a4792df5 acae2905 7b8125d0 55a677de
      1c624f0e 7212f6d9 63d74475 fbb5a69c bf4abc59 694dfb15 98b9d32a 53f9324b

Jr.a= 7e100de0 1f4cfbee 120b3902 49022589 5aebca4f cc8e41eb 9fffb00b 7b49d0d1
      01c5127e 85117211 eb0847e5 e2bfc40f 0245b657 cfca575c c6bbbaee 40c815d5

Jr.b= 361fdbe3 5e81de98 2c2cd0db 0d3d98c9 b37e732c 52cb857e 017ad3cd ad087830
      c0df9179 6f024a2d b29ce2f5 85ff424b ea7b8d37 d535716f 53593bf7 f0686c5e

Ex = 7f3e916f 5e617517 0119fb05 4733b8fe d38a2fad 86387019 38a92fb9 0a2dc631
      2a879d6d 46c0f7d1 a567b614 8f656c46 35f1d405 42419e32 71e1ec62 787889e9

```

```

Ey = 016b9bf2 4e3d7c78 a339d895 817d66b3 2f8e8a8a dcb059fb d8c75acf 93454afe
     c83fa158 1c26cfb5 90d893bd c1e7d092 a468ec3b 38d9920d d90db72c cbeab75f

```

Within H2:  $z = \text{FE2OSP}(\text{Or}) = \text{I2OSP}(\text{Or.a} + (\text{Or.b})q, 128)$

```

z = 25ec7faa cafa1847 0a45e774 f709bb02 7edea353 dc4ba6ab 1f821235 b19962f8
    c895e709 e90d46f8 143ce16b e049dead eda63f29 2b7b596d 41dc2d8d 415ff97f
    ab1341b7 9a32c89e 66ec586c 26f80e58 2036239b 3314fe03 a102abd4 1af8988b
    e8484a92 b36b1446 80167cf4 be6192cd 12e482be 5cbdb457 a3135f0a 261b73ff

```

```

B = 3dc6fe7b 2d8c7f35 4cffd996 0ae5274c 40780dlf deadf5e7 2f3be147 c33c017d

```

```

B+m = f6f2e46d 297b0886 897beb2f c3d2d6ed 6c9fdb63 2ef39cd0 af7661f3 b85f604c

```

```

K = 3ca06eae 57a1895e 279950b7 503ce290 31b38175 4e7747f8 7092514f 91dc8202

```

Session key encapsulation completed.

### C.2.2.4 Decapsulation

```

B = 3dc6fe7b 2d8c7f35 4cffd996 0ae5274c 40780dlf deadf5e7 2f3be147 c33c017d

```

```

m = cb341a16 04f777b3 c58432b9 c937f1a1 2ce7d67c f05e6937 804d80b4 7b636131

```

```

r = 54d92604 6c850c8b f0a95b7d 71e49ff6 a4792df5 acae2905 7b8125d0 55a677de
    1c624f0e 7212f6d9 63d74475 fbb5a69c bf4abc59 694dfb15 98b9d32a 53f9324b

```

```

Qx = 7f3e916f 5e617517 0119fb05 4733b8fe d38a2fad 86387019 38a92fb9 0a2dc631
    2a879d6d 46c0f7d1 a567b614 8f656c46 35f1d405 42419e32 71eleec62 787889e9

```

```

Qy = 016b9bf2 4e3d7c78 a339d895 817d66b3 2f8e8a8a dcb059fb d8c75acf 93454afe
    c83fa158 1c26cfb5 90d893bd c1e7d092 a468ec3b 38d9920d d90db72c cbeab75f

```

```

K = 3ca06eae 57a1895e 279950b7 503ce290 31b38175 4e7747f8 7092514f 91dc8202

```

Decapsulation successful.

## C.3 BB1 ID-based key encapsulation mechanism

### C.3.1 Example 1

#### C.3.1.1 Setup

This example makes use of the same elliptic curve  $y^2 = x^3 + 1$  and the Weil pairing.

```

q = b35fa5fd e47fa1ab bb1e57e9 3ba1ff96 38b89b99 5c49be81 a38e3194 a0983816
    4ee51fb9 1d285832 f9a05d63 9c8d9680 10c93a35 27e561f2 fd6a45cc 70aba1fb

```

```

p = 80000000 000fffff ffffffff ffffffff ffffffff

```

```

Q1x = 7ee6f118 9329adb4 1e8cd405 2295f1a7 6096631d f065dd38 85fff26b 8ed52022
    7bfc3f3d 07fe9cfe 093424ba 9dbd4c0d 73fff367 3ce2c922 f0b73c50 2992093b

```

```

Q1y = 67c855c9 e6b617be b24b792a e9c3e21e 95f37006 25b91058 3bc0b293 c36a762c
    feba4266 038989cc 59797235 c6116d99 8a97b805 ff82c664 53720e5c 1de95e3e

```

```

Q2x = 596aabeb e89571de 383450c5 4290a154 60bde0e2 a3a982cf f46596f9 9121e38d
    857d1362 4d445bf7 8ab3377d ecc122e3 2c3255f0 62369c8a 7d13d2d4 af91e143

```

```

Q2y = 4e89178d 16514938 ae638f70 76a4ca1f 6082ec4b 1c444785 6769e791 385e81da
    c8f7b174 8a2ce2fd c6080cf7 d6631e07 da57f5f9 adb85b7d 149f434e fe2dca40

```

```

s1 = 7623d9ea 1f8beada 83952da1 7d204958 109b585d

```

**ISO/IEC 18033-5:2015+A1:2021(E)**

```

s2 = 7fcbfded 301777ed 3dc86e24 7fcbfded 301777ed
s3 = 10a20b20 f0c0a1e0 e422555b 23abbafe 25b76ea7
Rx = b2888069 db9e121e 559afb38 3fe2b451 a4108bae 94ec259f dc8aa1ba b9fd8ab0
    0a7b90f5 47abf145 7826e9f7 a57f7213 8748f0a9 4519cbcd f5de3cf9 dff95c2c
Ry = 4802e661 5b71faab 5f9816fd bfde04c0 1de7d0a3 c9eb05c7 84441dcd 1def44ea
    d02317f4 ec8e7754 301d2ee4 1176aa94 caa49709 f03de657 05edc9b2 5f5aa2dc
Tx = 607db02e ce4deb26 4a935296 03234fc7 aa83c96d 0ca3e8ee 293c3cf8 730bfff5
    c05c75af ala1e0bf 8eee41de 89f41f03 9330883d aafd5efb ed43239f 19826577
Ty = 3f3dbea9 c4b05a71 7f0fce9b 282ae771 e7eace91 a72c7ca6 713d7a63 acb6fc35
    7c698d9c a4a8dfd4 a03968e5 a7e68a51 a0e7e23e 4aec721c 65580e51 675d5307
J.a = 6b4693b7 e66ad25a b547042c b8f858d3 ad52a8c2 ea980ab1 c0a9326a 4b93e6ec
    d3cb3a8f 29198f1d 1e4f5356 d51c116f ffde1928 58429878 8439c2c1 11a6abb1
J.b = 7c263a78 b70b6902 42d352e1 a89dc9f6 06e22732 8e1ee8dd f2231c30 7ceed6c9
    d6ef65a3 ac7419a0 a53a3146 af9e6dc0 dc93722a e03992a1 53025a8b 0c47f0ed
k = 192

```

**C.3.1.2 Private key extraction**

```

ID = sc27wg2-secretary@ipa.go.jp
IDb = 73633237 7767322d 73656372 65746172 79406970 612e676f 2e6a70
M = 305c1ac0 9f517694 0c6f4153 02fd22ff dd9214c9
r = 307df8f5 11358190 47694eae 8f2dc0c3 f4d81e56
t = 5f2350f9 631d7095 033137d5 2f597415 304b712b
dID0x= 2d92e629 f86c6fa8 b1bddf5e e8872965 2cf04dce 62f08160 e84a644b 432d1a1c
    ccd6272d 0bac8f18 86a6a434 f524c0fa d73d1499 0938225a 715fba67 a7a626a2
dID0y= 6515d933 9124341c 8ab13862 5e433278 afa8b6a7 32daf20b 5a9f3d1f 327d770b
    db8a18b5 3c74c97b b746a500 36e812c0 65c29971 e6095c5e 13968410 793a363f
dID1x= 929cc12c fd74beb6 4f062f55 0f8a252b e3b8abbb 0abeb7b1 7d3ac147 d4ad83c4
    4f1069f0 1616e318 e5087cec 49873cb7 fc11cfd7 e3bfbf3c 69e5337e 7899067f
dID1y= 4e1b894e 87aebd2c 6dc24553 4e7f1e1f 62558914 e1d2a924 e7853b54 b9b741db
    0517870d ac830baf 54fb958d ad4c3530 2a4f7d6f b466d0fb eb441217 220f4ed4
T0.a= 51d0adff dfb544ac 56867378 4ab4419d 4c86c3ad b4f5b851 d8a1dbc8 592b007c
    0e216163 4748a58b c57690d2 0fb77466 3865c31c e1b1a0a1 2726c483 a27fb151
T0.b= 867bbac1 acb9ad9b 91bce537 a3b14e57 8dbe44a1 5e8ea58f c4a698f1 309f84fb
    caf08027 73bc17da be80a6e1 dac0b4cd f8c4bedc e8bdfe0a 7a0ddb01 fc3bcde3
T1.a= a68e9d1f 8b2b1afc 4eb2de87 29d058a7 4f4ac828 c3da3fc6 c6e9b9c3 d86cdeae
    066e2d6f 18c6613e 393b4b39 242afddd bclade14 02d3cc27 9943019b 11b7274b
T1.b= a8cb789e 20f1daf3 9b259428 eb63ca7b 009142f9 a0f2cc82 82a8cad2 b28ebddd
    13bbb836 9b311a56 1758817a c2d1f199 dcb6c790 f3ba5bf6 dd19ba1d 4478918b
OT1.a= 51d0adff dfb544ac 56867378 4ab4419d 4c86c3ad b4f5b851 d8a1dbc8 592b007c
    0e216163 4748a58b c57690d2 0fb77466 3865c31c e1b1a0a1 2726c483 a27fb151
OT1.b= 867bbac1 acb9ad9b 91bce537 a3b14e57 8dbe44a1 5e8ea58f c4a698f1 309f84fb
    caf08027 73bc17da be80a6e1 dac0b4cd f8c4bedc e8bdfe0a 7a0ddb01 fc3bcde3

```

Private key <dID0, dID1> is valid.

**C.3.1.3 Session key encapsulation**

```

r' = 798cdb57 64e9d910 fb353f92 e8d22d4b c6ceff27

E0x = 3ec733e6 f813c81f f3a272cd 015ffefd 4ac02c03 e9d7aa58 7a41c238 02a55298
      093268b1 696a7396 0b75318f 27ef9e36 721a435c ebd48fbb 08c288ae f9f420fb

E0y = 3cde70f8 0157e267 a466061f ffceb029 b65f0c78 b6a671cb d9c388f9 5882abb1
      6089f6a1 32f480a8 69b7210c 9aded83e 8a26bf37 521a03bd 83b3d759 16b2ff13

E1x = 3f259b34 9f128746 8b6f9df1 f929a11f 2b060ccf c74c9c82 befd7f35 f30e8b7b
      fbdfb419 041738c5 22c1651b cd1b5e0a 4fade396 7ddf2a5d 7f7186ae 98b0632c

E1y = 91e827b1 c13331be ab021930 ce980ab7 a2d32415 e155e871 4fe4f7bb c1926196
      c08a0f49 97f36c17 6075608f 9739a921 d0438e3d 76e8f6f3 47a6a3fa 0c44edf3

B.a = 2d763660 63070c7e ecf627b2 7086d821 07473066 3a654b19 842f9e01 0fda239f
      51e72f6f 1f4a24b1 afd999f8 966ccab7 b1b4ba3b 1799e34d 019276f1 b11c928a

B.b = 677206ed 615d812c 1fbcf31e d7ef074e 50ec925b c7605867 c99347f4 610e27ae
      3cb7b7bc 51ee0311 f2956123 06e2a611 3e01f593 e1537d82 4db21660 416eb44e

Within H2: z = FE2OSP(B)= I2OSP(B.a+(B.b)q, 128)

z = 487b613b a3a87c70 3cd80b59 5de3e836 290815da 82cc0519 f1520883 2b74e6c8
    05fa7143 32f5c330 f4039bc9 65dd8935 f2d97365 e4aa01cf 4bc0a5bf 3ba3b348
    b3814906 ad5b230a 6b8faf1d f33f91d5 e4ca174f 7f07726e d7814d0f 9a566d96
    104de63c aa80ee15 231b78ea 52d856d2 f070e0ab a5a1fac5 f749c875 08266904

K = 080d9ed8 c25802ce df4a24da 0ae3f8c4 5fa0f9d2 5e0f875a b39f7e52 d10e8b20

Session key encapsulation completed.

```

**C.3.1.4 Decapsulation**

```

B.a = 2d763660 63070c7e ecf627b2 7086d821 07473066 3a654b19 842f9e01 0fda239f
      51e72f6f 1f4a24b1 afd999f8 966ccab7 b1b4ba3b 1799e34d 019276f1 b11c928a

B.b = 677206ed 615d812c 1fbcf31e d7ef074e 50ec925b c7605867 c99347f4 610e27ae
      3cb7b7bc 51ee0311 f2956123 06e2a611 3e01f593 e1537d82 4db21660 416eb44e

K = 080d9ed8 c25802ce df4a24da 0ae3f8c4 5fa0f9d2 5e0f875a b39f7e52 d10e8b20

Decapsulation successful.

```

**C.3.2 Example 2****C.3.2.1 Setup**

This example makes use of the same elliptic curve  $y^2 = x^3 + x$  and the Reduced Tate pairing.

```

q = 80000000 00000000 00000000 00000000 00020001 40000000 00000000
    0000000000000000 00010000 80000002 00000000 00000000 00000000 00000000
    00080003

p = 80000000 00000000 00000000 00000000 00020001

Q1x = 0db4e0f7 22dd090d a2b6d8fe adaf21d9 546ab265 1515af9b a87108f3 4e1ae0e3
      eb132c10 81452cc1 e52bb2a7 4287a0cb d8ff8dd9 3a225641 5321f0e4 c8892a50

Q1y = 762c096c 49f1ab04 7d7f37de 537a4e7c 2991c400 22e0c9a9 b3f58b1b 9df4f28a
      4a4330e2 170e14d2 f55a0719 8b667d0b 01e5a482 3f07e921 8516481e 641970ac

Q2x = 6b8f666b cf6b4672 d4634753 1f734e71 41bcd5fd 125f3ef3 714edc28 f6426900
      75ffb5f7 9e745cc0 fb03f940 3bdcefe8 acbe6286 d5d9955c 2a0e5ed7 657748c6

```

**ISO/IEC 18033-5:2015+A1:2021(E)**

Q2y = 69584e47 f3070fed 9800d6cd e0f314b4 03955126 1c5bfef6 f3595f94 5958f7d9  
 34dcbd3d 63125410 ccd363f8 02df1c7e 4a3d7ac7 24cf3865 0fb16ec1 7bb30a85  
  
 s1 = 7623d9ea 1f8beada 83952da1 7d204958 109b585d  
  
 s2 = 7fcbfdded 301777ed 3dc86e24 7fcbfdded 301777ed  
  
 s3 = 10a20b20 f0c0a1e0 e422555b 23abbafe 25b76ea7  
  
 Rx = 142d5e5d 22670c96 1fa51e66 1882a317 0541ac63 653ce2d5 0be026a7 891699c8  
 8d413e52 607c91c9 623de680 2268809b b3b2b57f 437713e1 ce7f756a f1d0809a  
  
 Ry = 0b418104 b1b5dd7e 9144a853 5b9b7dd2 5e30ab90 97355d60 75052c0d edd738b8  
 e71e0773 16b772a9 2fc72619 25d7fd83 65dd097b 56010aa9 6ada597d 3b54df7e  
  
 Tx = 6163c179 8bd08560 79b96b62 f8e97961 bfcac250 a8dd433d e02f2180 50cf90c1  
 372e2e30 438cfe0c 1be1640c f7370d29 bb6ad4c4 163ace4a aa7c316e 0e47801e  
  
 Ty = 427f55b2 10f57d60 4271d022 d184ef29 fed0a2c4 7e51a99b b7340c5f 9b084693  
 6142cd8e 0a89172c d503656d 3e358164 3170ad13 c2c4b93e d96a83e6 215a52e6  
  
 J.a = 518c40f1 f0942278 64e49282 8ee77ac4 5d8943e8 abacf3f1 fdc8d465 1ac1cfb5  
 2bb1bf59 e941db84 70f92c3e 55fd0645 0888800d 0046a33a ee4570cb 6f462617  
  
 J.b = 5231d656 4a2b144c 4b1a38c3 804db0a3 b4930a19 1496ad21 71617cf0 0859f16f  
 a4bde018 9b851678 a683205d 04e0768d cf22b1c0 2907525b 97abf0f8 09e4b7a1  
  
 k = 192

**C.3.2.2 Private key extraction**

ID = sc27wg2-secretary@ipa.go.jp  
  
 IDb = 73633237 7767322d 73656372 65746172 79406970 612e676f 2e6a70  
  
 M = 7de5a0a1 92c9b2b0 b3847381 72c16e9e 2cb63566  
  
 r = 05351a22 d10f5339 dfe1eb30 30924229 2302d090  
  
 t = 703a597c 3b7396f8 386e89b9 5af2fb1b 76969bd3  
  
 dID0x= 5ed9195c 81863738 e8cf7a26 3e00db1c e8a72a22 7579309d 3a11282f 964a54c0  
 27446231 58c24ab3 ae4c901f a18e210b adc60fc6 f5d310bc a900351c dcf782b0  
  
 dID0y= 1a490f37 a26b205e 789f5585 a49fffd9d 9491feac 01b5b2bb 1f4fdcab d301f533  
 874034f9 1a8db549 15422e4a 3ca3a83e 4eecab72 f863a746 d640d798 675ba0b5  
  
 dID1x= 5d2a8994 ed44dca6 852a5ebb ba7337f9 824aee7c e904ffd1 20ddc0e7 dbca2b16  
 ccbe101e e4adac32 3cb9a8ed 37c34720 16519890 44417767 e1acdf0b ee0a4459  
  
 dID1y= 6264b10b 9c5b5ad9 7d56ef17 5922caf4 ee8154f4 aaca86ac a72c7b34 8e9cd873  
 a0659aa6 b7e0ee34 850f4def 9b5b50b6 98ff5571 827a1d64 37783bc7 db378def  
  
 T0.a= 44084f72 c31c6b70 55dd4931 140b6a18 77db2fd4 fae3944e a6838cb9 bd27326c  
 cd366fe9 c848610c e0e0e473 97d04196 40d0e03f 0924aa21 8b6dac56 795ea605  
  
 T0.b= 69c27507 78d5272e c8938f17 c868c956 df8d2364 b2b8e1ff dc73a3c9 5a6d5c9a  
 315d3d69 7a6ald3d 68d551c5 68a1feef a4dd73a5 854cfb63 16c1281f eaf09e31  
  
 T1.a= 6ef90912 510ba657 0121df17 2ae97124 9353f2b2 9bcf43aa 533b0e4a 0f68b6e3  
 flaba24c f8c3c018 619f6c0e adf3452c ac91d8c5 69451a9c 91fc7f30 ec31dd7d  
  
 T1.b= 65ba8521 75167ea3 83ed9488 24347da3 b3541c17 dbaad01 482a3772 a1e122a8  
 6f06bcc3 66cfd74d 80dcdab47 32a68280 7233968b 655327be 075b7187 ab8d8a19  
  
 JT1.a= 44084f72 c31c6b70 55dd4931 140b6a18 77db2fd4 fae3944e a6838cb9 bd27326c  
 cd366fe9 c848610c e0e0e473 97d04196 40d0e03f 0924aa21 8b6dac56 795ea605

```

JT1.b= 69c27507 78d5272e c8938f17 c868c956 df8d2364 b2b8e1ff dc73a3c9 5a6d5c9a
      315d3d69 7a6ald3d 68d551c5 68alfeef a4dd73a5 854cfb63 16c1281f eaf09e31

```

Private key <dID0, dID1> is valid.

### C.3.2.3 Session key encapsulation

```

r'   = 3bda4388 5211ae25 fa4ad7d8 fecf0108 786cb1bc

E0x  = 266f9cde 74752bc1 15bb0a01 a122c008 c3d50b40 6429d9e2 7ea6a6c8 f653b932
      b468d332 2ee75be7 a259d6bb 5330da08 85feee12 0fe78295 33759ff9 26bc7f4a

E0y  = 202390c6 276b4df0 afba4684 a88d6deb 7005dfc4 674f866b 2e275e04 bedba792
      2c901ba4 aee167c6 603e35dd e8153c98 c46d99b4 2c3eaf3b dd84aade 9270b9ad

E1x  = 26405024 f8bf9a2d 2884f055 abdae2f5 5823d52c 50498888 35b8a7b9 d1b226e3
      8ae4710b 1473b6cd f5fddc41 a96a9a47 5c065fad ac4e5b7f b549b7eb 25141d6d

E1y  = 644172fb 549983ab 7b8ca683 4966c1fd 14a2667e 292b34c6 5001bded 208e4e74
      e49b6583 ac06c840 bf08146d ff481576 f9e16dce 133c9b34 4cfc9ef5 fd4c7798

B.a  = 7e22bfc1 112d0bd0 d30aed86 2923563d 546157a0 1188f9d0 70c11677 f965ca25
      8dcc03a2 9b5decdb b2db93c0 92b6275e 76dfa49c bca75f80 298d7e43 b05b2b86

B.b  = 4ffda204 0a79ed2f 82fbbae0 0dec9464 78b0588c 37e2ec87 dcb9ada7 8ffee336
      5cdad0f2 df0cc9ab 97fbbb31 6e34030c 5180c4ea 0abebede b2106de1 cd56e42f

```

Within H2:  $z = \text{FE2OSP}(B) = \text{I2OSP}(B.a + (B.b)q, 128)$

```

z    = 27fed102 053cf697 c17ddd70 06f64a32 3c58cc41 c3f695bc d5d44546 a17a370c
      689e1357 b77b9348 b4ec1a32 78b8f830 f47d7dea 3ca8db5b 33ecef0 d4e57c81
      d9dfe04f ee7520a4 5d886e35 234c9bac d1279cba c78e6e9f 3e3d12b2 2a54f5f3
      085af0e6 b4dc5797 6f07ff52 f5b4bc89 92b24950 d3d92c9f aecd32a0 39d7d813

```

```

K    = 9f9f10c1 2616df83 b785a730 0a57b165 92ef8790 f5869922 2a9056e0 cd0ada89

```

Session key encapsulation completed.

### C.3.2.4 Decapsulation

```

B.a  = 7e22bfc1 112d0bd0 d30aed86 2923563d 546157a0 1188f9d0 70c11677 f965ca25
      8dcc03a2 9b5decdb b2db93c0 92b6275e 76dfa49c bca75f80 298d7e43 b05b2b86

B.b  = 4ffda204 0a79ed2f 82fbbae0 0dec9464 78b0588c 37e2ec87 dcb9ada7 8ffee336
      5cdad0f2 df0cc9ab 97fbbb31 6e34030c 5180c4ea 0abebede b2106de1 cd56e42f

```

```

K    = 9f9f10c1 2616df83 b785a730 0a57b165 92ef8790 f5869922 2a9056e0 cd0ada89

```

Decapsulation successful.

## C.4 SM9 ID-based key encapsulation mechanism

### C.4.1 Example 1

#### C.4.1.1 Set up

This example makes use of the same Barreto-Naehrig elliptic curve  $y^2 = x^3 + 5$  used in ISO/IEC 14888-3:2018, F.15.1. An element  $A_0$  in  $Fq^2$  is represented as  $A_{0,1}\sigma + A_{0,0}$ , where  $A_{0,0}$  and  $A_{0,1}$  are elements of  $Fq$  and  $\sigma$  is an element of  $Fq^2$  such that  $\sigma^2 + 2 = 0 \pmod q$ . Let  $v$  be an element of  $Fq^4$  such that  $v^2 - \sigma = 0$  in  $Fq^2$  and  $\omega$  be an element of  $Fq^{12}$  such that  $\omega^3 - v = 0$  in  $Fq^4$ , an element of  $Fq^{12}$  is represented as  $A\omega^2 + B\omega + C$ , where  $A, B, C$  are elements of  $Fq^4$  which are represented as  $A = A_1v + A_0$ ,  $B = B_1v + B_0$ ,  $C = C_1v + C_0$  respectively, and  $A_0, A_1, B_0, B_1, C_0, C_1$  are elements of  $Fq^2$ . In this towered fashion, an element of  $Fq^{12}$  is represented as a vector  $(A_{1,1}, A_{1,0}, A_{0,1}, A_{0,0}, B_{1,1}, B_{1,0}, B_{0,1}, B_{0,0}, C_{1,1}, C_{1,0}, C_{0,1}, C_{0,0})$  with components in  $Fq$ .  $P$  is a point on the curve  $y^2 = x^3 + 5$ ,  $Q$  is a point on the corresponding sextic twist  $y^2 = x^3 + 5\sigma$ . Pairing  $e$  is implemented as the optimal R-ate pairing on two input points  $P$  and  $\phi(Q)$ , i.e.,



## ISO/IEC 18033-5:2015+A1:2021(E)

$e(P, Q) = [d(\phi(Q), P, u) \cdot f(u\phi(Q), \pi_q(\phi(Q)), P) \cdot f(u\phi(Q) + \pi_q(\phi(Q)), -\pi_{q^2}(\phi(Q)), P)] \frac{q^{12}-1}{p}$ , where  
 $q = 36z^4 + 36z^3 + 24z^2 + 6z + 1$ ,  $p = 36z^4 + 36z^3 + 18z^2 + 6z + 1$ ,  $u = 6z + 2$ ,  $z$  is the parameter of the BN curve,  $\phi$  is the group homomorphism such that  $\phi(x, y) = (\sigma^{-1/3}x, \sigma^{-1/2}y)$  and  $\pi$  is the Frobenius endomorphism such that  $\pi_{q^i}(x, y) = (x^{q^i}, y^{q^i})$  for  $i = 1, 2$ .

```

q  = b6400000 02a3a6f1 d603ab4f f58ec745 21f2934b 1a7aeedb e56f9b27 e351457d
p  = b6400000 02a3a6f1 d603ab4f f58ec744 49f2934b 18ea8bee e56ee19c d69ecf25
Q1x = 93de051d 62bf718f f5ed0704 487d01d6 e1e40869 09dc3280 e8c4e481 7c66dddd
Q1y = 21fe8dda 4f21e607 63106512 5c395bbc 1c1c00cb fa602435 0c464cd7 0a3ea616
Q2x = 85aef3d0 78640c98 597b6027 b441a01f f1dd2c19 0f5e93c4 54806c11 d8806141,
    37227552 92130b08 22aab97f d34ec120 ee265948 d19c17ab f9b7213b af82d65b
Q2y = 17509b09 2e845c12 66ba0d26 2cbee6ed 0736a96f a347c8bd 856dc76b 84eb9b96,
    a7cf28d5 19be3da6 5f317015 3d278ff2 47efba98 a71a0811 6215bba5 c999a7c7
s  = 0001edee 3778f441 f8dea3d9 fa0acc4e 07ee36c9 3f9a0861 8af4ad85 cede1c22
Rx = 787ed7b8 a51f3ab8 4e0a6600 3f32da5c 720b17ec a7137d39 abc66e3c 80a892ff
Ry = 769de617 91e5adc4 b9ff85a3 1354900b 20287127 9a8c49dc 3f220f64 4c57a7b1
J  = 9746fc5b 231cedf3 6f835c47 893d63c6 ff652bcb 92375ce3 c2ab256d 1fd56413,
    232a2f80 cfbac061 f196bb99 213d5030 6648ac33 cdc78e8f 8a1563ff bf3bd3eb,
    68e8a16c 0ac905f6 92904abc c004b1ac f12106bd 0a15b6e7 08d76e72 b9288ef2,
    9436a60c 403f4f8b ac4dd3e3 93e25419 e634fc2b 3daf247f 6092a802 f60d5c58,
    a140eaeef 3893d574 cb83c01d 951a53f5 1975760b e57f3bbd 89817498 d2158352,
    95a2bcce 25359d03 3fc654bd 6a9e462e 5bd0686f f6ddd745 5f71ffff 5affd3f0,
    b0432019 0b1e90ce df6ac570 147a23ae 6f0eae45 034e6c62 124dd6e8 978f78ad,
    a504e3b4 3c1dd367 94217fa1 b05ac046 c4131854 c3d3e3a5 b5967a64 a861f0a2,
    897f7b35 d1c0e21d 84d75cff ac08c73e 744a16a4 7ee76e28 a0b03849 888d10ff,
    24443bb4 24b12c41 eaf6d34d 92520590 1f5cba59 cfeba352 24660db3 848b0bf5,
    0825403f b3f681ab 2b036dbb a25483d5 cb98bd56 f3df95f0 a7a705a2 f6fd804b,
    9ce7bc68 062182cf 5d9f4a98 c5a4ed1f 3b4ce4ea 817d19ed 7ef2ce98 e6f5864d

```

## C.4.1.2 Private key extraction

```

ID    = Bob
IDb   = 426f62
M     = 9cb1f628 8ce0e510 43ce7234 4582ffc3 01e0a812 a7f5f200 4b85547a 24b82716
t     = 864e4d83 91948b37 535ecfa4 4c3f8d4e 545ada50 2ff8229c 7c32f529 af406e06
skIDx 94736acd 2c8c8796 cc4785e9 38301a13 9a059d35 37b64141 40b2d31e ecf41683,
    115bae85 f5d8bc6c 3dbd9e53 42979acc cf3c2f4f 28420b1c b4f8c0b5 9a19b158
skIDy = 7aa5e475 70da7600 cd760a0c f7beaf71 c447f384 4753fe74 fa7ba92c a7d3b55f,
    27538a62 e7f7bfb5 1dce0870 4796d94c 9d56734f 119ea447 32b50e31 cdeb75c1
T     = 9746fc5b 231cedf3 6f835c47 893d63c6 ff652bcb 92375ce3 c2ab256d 1fd56413,
    232a2f80 cfbac061 f196bb99 213d5030 6648ac33 cdc78e8f 8a1563ff bf3bd3eb,
    68e8a16c 0ac905f6 92904abc c004b1ac f12106bd 0a15b6e7 08d76e72 b9288ef2,
    9436a60c 403f4f8b ac4dd3e3 93e25419 e634fc2b 3daf247f 6092a802 f60d5c58,
    a140eaeef 3893d574 cb83c01d 951a53f5 1975760b e57f3bbd 89817498 d2158352,
    95a2bcce 25359d03 3fc654bd 6a9e462e 5bd0686f f6ddd745 5f71ffff 5affd3f0,
    b0432019 0b1e90ce df6ac570 147a23ae 6f0eae45 034e6c62 124dd6e8 978f78ad,
    a504e3b4 3c1dd367 94217fa1 b05ac046 c4131854 c3d3e3a5 b5967a64 a861f0a2,
    897f7b35 d1c0e21d 84d75cff ac08c73e 744a16a4 7ee76e28 a0b03849 888d10ff,
    24443bb4 24b12c41 eaf6d34d 92520590 1f5cba59 cfeba352 24660db3 848b0bf5,
    0825403f b3f681ab 2b036dbb a25483d5 cb98bd56 f3df95f0 a7a705a2 f6fd804b,
    9ce7bc68 062182cf 5d9f4a98 c5a4ed1f 3b4ce4ea 817d19ed 7ef2ce98 e6f5864d

```

Private key skID is valid.

## C.4.1.3 Session key encapsulation

```

r     = 0000aac0 541779c8 fc45e3e2 cb25c12b 5d2576b2 129ae8bb 5ee2cbe5 ec9e785c
IDb   = 426f62
M     = 9cb1f628 8ce0e510 43ce7234 4582ffc3 01e0a812 a7f5f200 4b85547a 24b82716
Ex    = 24454711 64490618 e1ee2052 8ff1d545 b0f14c8b caa44544 f03dab5d ac07d8ff
Ey    = 42ffca97 d57cdcd0 5ea405f2 e586feb3 a6930715 532b8000 759f1305 9ed59ac0
B     = 63253798 b7535975 a90f2025 61fc5457 0fee88bf 69e3b7a5 12697069 e59e1f5d,
    42d54b98 4af01d71 0ba0030c 18738f6b 14e4df47 2acaf893 99228d85 af117904,
    b426dff0 40c49f9a 43bcd7fd 7d757b7d 1d8d7311 c08fc3b5 7616c5ee 137785a3,
    28d19396 dbdfac50 eee62b1c 7f994bb6 f9bd9efb 2221a1be 1b6eb3e8 f71485b4,
    a3eef46e 1b99f614 d7bd7f57 574ba7eb b502af0b daba0787 c5c4dbc5 6a344a25,
    a06790b6 05cea0bb af34776d 6b1fc019 8a02d05b baac6f64 a555ab2c a576f0da,
    b405cbbf 22197b94 fd18d27d a0b0e52c 8754ee94 27963469 1fea6e13 ffd0584e,

```

```

aa2a94a7 e2259b67 1896302b 4275ae3e 8cf20100 98d5beaf 19d0a6e6 0354e1c5,
5c97e64f 848b06d3 9ba8828f f59502c0 81d3dae6 8f35f7e6 448db96d 220a0fba,
02be03c5 1bf062b6 f564ae0b fb42dca3 6e71d387 512e3bcc ca3379b7 3ec47176,
52be92fb 9e78ba9e 1d80a156 06580493 5742dbd2 b9675430 11aac533 33909fbf,
5fadec14 a2fbd152 48e77467 442a6969 8246fb03 14c7a824 6d952219 dd2144ed
EC2OSP(E) || FE2OSP(B) || IDb =
24454711 64490618 e1ee2052 8ff1d545 b0f14c8b caa44544 f03dab5d ac07d8ff
42ffca97 d57cddc0 5ea405f2 e586feb3 a6930715 532b8000 759f1305 9ed59ac0
63253798 b7535975 a90f2025 61fc5457 0fee88bf 69e3b7a5 12697069 e59e1f5d,
42d54b98 4af01d71 0ba0030c 18738f6b 14e4df47 2acaf893 99228d85 af117904,
b426dff0 40c49f9a 43bcd7fd 7d757b7d 1d8d7311 c08fc3b5 7616c5ee 137785a3,
28d19396 dbdfac50 eee62b1c 7f994bb6 f9bd9efb 2221a1be 1b6eb3e8 f71485b4,
a3eef46e 1b99f614 d7bd7f57 574ba7eb b502af0b daba0787 c5c4dbc5 6a344a25,
a06790b6 05cea0bb af34776d 6b1fc019 8a02d05b baac6f64 a555ab2c a576f0da,
b405cbbf 22197b94 fd18d27d a0b0e52c 8754ee94 27963469 1fea6e13 ffd0584e,
aa2a94a7 e2259b67 1896302b 4275ae3e 8cf20100 98d5beaf 19d0a6e6 0354e1c5,
5c97e64f 848b06d3 9ba8828f f59502c0 81d3dae6 8f35f7e6 448db96d 220a0fba,
02be03c5 1bf062b6 f564ae0b fb42dca3 6e71d387 512e3bcc ca3379b7 3ec47176,
52be92fb 9e78ba9e 1d80a156 06580493 5742dbd2 b9675430 11aac533 33909fbf,
5fadec14 a2fbd152 48e77467 442a6969 8246fb03 14c7a824 6d952219 dd2144ed,
426f62
CTDEM = EC2OSP(E)
With KDF2-a(EC2OSP(E) || FE2OSP(B) || IDb, 384),
K = 58373260 f067ec48 667c21c1 44f8bc33 cd304978 8651ffd5 f738003e 51df3117
4d0e4e40 2fd87f45 81b612f7 4259db57
With KDF2-a(EC2OSP(E) || FE2OSP(B) || IDb, 416),
K = 58373260 f067ec48 667c21c1 44f8bc33 cd304978 8651ffd5 f738003e 51df3117
4d0e4e40 2fd87f45 81b612f7 4259db57 4f67ece6

```

#### C.4.1.4 Session key de-capsulation

```

B = 63253798 b7535975 a90f2025 61fc5457 0fee88bf 69e3b7a5 12697069 e59e1f5d,
42d54b98 4af01d71 0ba0030c 18738f6b 14e4df47 2acaf893 99228d85 af117904,
b426dff0 40c49f9a 43bcd7fd 7d757b7d 1d8d7311 c08fc3b5 7616c5ee 137785a3,
28d19396 dbdfac50 eee62b1c 7f994bb6 f9bd9efb 2221a1be 1b6eb3e8 f71485b4,
a3eef46e 1b99f614 d7bd7f57 574ba7eb b502af0b daba0787 c5c4dbc5 6a344a25,
a06790b6 05cea0bb af34776d 6b1fc019 8a02d05b baac6f64 a555ab2c a576f0da,
b405cbbf 22197b94 fd18d27d a0b0e52c 8754ee94 27963469 1fea6e13 ffd0584e,
aa2a94a7 e2259b67 1896302b 4275ae3e 8cf20100 98d5beaf 19d0a6e6 0354e1c5,
5c97e64f 848b06d3 9ba8828f f59502c0 81d3dae6 8f35f7e6 448db96d 220a0fba,
02be03c5 1bf062b6 f564ae0b fb42dca3 6e71d387 512e3bcc ca3379b7 3ec47176,
52be92fb 9e78ba9e 1d80a156 06580493 5742dbd2 b9675430 11aac533 33909fbf,
5fadec14 a2fbd152 48e77467 442a6969 8246fb03 14c7a824 6d952219 dd2144ed
EC2OSP(E) || FE2OSP(B) || IDb =
24454711 64490618 e1ee2052 8ff1d545 b0f14c8b caa44544 f03dab5d ac07d8ff
42ffca97 d57cddc0 5ea405f2 e586feb3 a6930715 532b8000 759f1305 9ed59ac0
63253798 b7535975 a90f2025 61fc5457 0fee88bf 69e3b7a5 12697069 e59e1f5d,
42d54b98 4af01d71 0ba0030c 18738f6b 14e4df47 2acaf893 99228d85 af117904,
b426dff0 40c49f9a 43bcd7fd 7d757b7d 1d8d7311 c08fc3b5 7616c5ee 137785a3,
28d19396 dbdfac50 eee62b1c 7f994bb6 f9bd9efb 2221a1be 1b6eb3e8 f71485b4,
a3eef46e 1b99f614 d7bd7f57 574ba7eb b502af0b daba0787 c5c4dbc5 6a344a25,
a06790b6 05cea0bb af34776d 6b1fc019 8a02d05b baac6f64 a555ab2c a576f0da,
b405cbbf 22197b94 fd18d27d a0b0e52c 8754ee94 27963469 1fea6e13 ffd0584e,
aa2a94a7 e2259b67 1896302b 4275ae3e 8cf20100 98d5beaf 19d0a6e6 0354e1c5,
5c97e64f 848b06d3 9ba8828f f59502c0 81d3dae6 8f35f7e6 448db96d 220a0fba,
02be03c5 1bf062b6 f564ae0b fb42dca3 6e71d387 512e3bcc ca3379b7 3ec47176,
52be92fb 9e78ba9e 1d80a156 06580493 5742dbd2 b9675430 11aac533 33909fbf,
5fadec14 a2fbd152 48e77467 442a6969 8246fb03 14c7a824 6d952219 dd2144ed,
426f62
With KDF2-a(EC2OSP(E) || FE2OSP(B) || IDb, 384),
K = 58373260 f067ec48 667c21c1 44f8bc33 cd304978 8651ffd5 f738003e 51df3117
4d0e4e40 2fd87f45 81b612f7 4259db57
With KDF2-a(EC2OSP(E) || FE2OSP(B) || IDb, 416),
K = 58373260 f067ec48 667c21c1 44f8bc33 cd304978 8651ffd5 f738003e 51df3117
4d0e4e40 2fd87f45 81b612f7 4259db57 4f67ece6
Key de-capsulation successful.

```



## ISO/IEC 18033-5:2015+A1:2021(E)

## C.4.1.5 Data encapsulation with DEM2

This example shows how to use DEM2 with the symmetric cipher SC1 specified in ISO/IEC 18033-2 to encrypt arbitrary-length messages with the session key generated by the SM9 key encapsulation mechanism. SM4 is used as the block cipher, which works in the cipher-block-chaining (CBC) mode with the initial value (IV) set as all 0s. SM3 is used as the required hash function. The label input  $L$  to DEM2 is empty.


```
Msg = Chinese IBE standard
Msg with padding:
4368696e 65736520 49424520 7374616e 64617264 0c0c0c0c 0c0c0c0c 0c0c0c0c
Parse K of length 384-bit as K' || K'' with K' of 128-bit and K'' of 256-bit:
K' = 58373260 f067ec48 667c21c1 44f8bc33
K'' = cd304978 8651ffd5 f738003e 51df3117 4d0e4e40 2fd87f45 81b612f7 4259db57
SC1.Encrypt(K', Msg) = SM4-CBC(K', Msg) with IV = 0s:
c = e05b6fac 6f11b965 268c994f 00dba7a8 132c9574 5b2cacb3 82fbfd90 6d9ba86a
MA.eval(K'', c) = SM3(c || K''):
MAC = 12af121d e3795aa5 14d0c6e7 949ce479 807e8b03 140dca09 d18dd075 e47eb03c
CTDEM = c || MAC
```

Data de-capsulation is the decryption process of DEM2. It runs with  $K''$  to check the correctness of MAC first, and then uses  $K'$  to decrypt  $c$ . If MAC does not pass the check, the decryption process outputs "error" and stops.

## C.4.1.6 Data encapsulation with DEM3

This example shows how to use DEM3 specified in ISO/IEC 18033-2 to encrypt fixed-length messages with the session key generated by the SM9 key encapsulation mechanism. DEM3 uses the bit-wise exclusive-or (XOR) operation to encrypt a message with a secret key. SM3 is used as the required hash function. The label input  $L$  to DEM3 is empty.

```
Msg = Chinese IBE standard
Parse K of length 416-bit as K' || K'' with K' of 160-bit and K'' of 256-bit:
K' = 58373260 f067ec48 667c21c1 44f8bc33 cd304978
K'' = 8651ffd5 f738003e 51df3117 4d0e4e40 2fd87f45 81b612f7 4259db57 4f67ece6
Encrypt(K', Msg) = K' ⊕ Msg:
c = 1b5f5b0e 95148968 2f3e64e1 378cdd5d a9513b1c
MA.eval(K'', c) = SM3(c || K''):
MAC = ba672387 bcd6de50 16a158a5 2bb2e7fc 429197bc ab70b25a fee37a2b 9db9f367
CTDEM = c || MAC
```

Data de-capsulation is the decryption process of DEM3. It first checks the length of  $c$  and then checks the correctness of MAC with  $K''$ , and finally uses  $K'$  to decrypt  $c$ . If any check fails, the decryption process outputs "error" and stops. 

## **Annex D** (informative)

### **Mechanisms to prevent access to keys by third parties**

No IBE works without a third party called the Private Key Generator (PKG), which generates private keys for all decryptors using its master-secret key. Thus, an IBE is available only when one allows such a key escrow operation. The compromise of the master-secret key could be disastrous in an IBE system, and usually more severe than the compromise of a CA's signing key in a traditional PKI. For this reason, it seems that the use of IBE may be restricted to small, closed groups or to applications with limited security requirements.

Al-Riyami and Paterson<sup>[2]</sup> introduced a new paradigm for public key cryptography, which they named certificateless public key cryptography (CL-PKC) that do not require the use of certificates and yet do not have the built-in key escrow feature of IBE. Their proposed encryption schemes enjoy both of these properties. Inspired by Al-Riyami and Paterson's work, several researchers proposed various certificateless public key encryption schemes. One can overview their studies in <sup>[6]</sup>

## Bibliography

- [1] ISO/IEC 18031, *Information technology — Security techniques — Random bit generation*
- [2] AL-RIYAMI S.S., & PATERSON K.G. Certificateless public key cryptography. In: *Advances in Cryptology — ASIACRYPT'03.* Springer, **Vol. 2894**, 2003, pp. 452–73
- [3] BONEH D., & BOYEN X. Efficient selective-ID secure identity based encryption without random oracles. In: *Advances in Cryptology — EUROCRYPT'04.* Springer, **Vol. 3027**, 2004, pp. 223–38
- [4] BONEH D., & FRANKLIN M.K. Identity-based encryption from the Weil pairing. In: *Advances in Cryptology — CRYPTO'01.* Springer, **Vol. 2139**, 2001, pp. 213–29
- [5] CHEN L., CHENG Z., MALONE-LEE J., SMART N. Efficient ID-KEM based on the Sakai-Kasahara key construction. *IEEE Proceedings Information Security*, 153(1):19-26, 2006
- [6] DENT A. W. A survey of certificateless encryption schemes and security models, *International Journal of Information Security*, Volume 7 Issue 5, pages 349-377. Springer-Verlag, 2008
- [7] IEEE Std 1363.3-2013, *IEEE Standard for Identity-Based Cryptographic Techniques using Pairings*
- [A<sub>1</sub>] [8] GM/T 0044.4-2016, *SM9 Identity-Based Cryptographic Algorithms using Bilinear Pairings — Part 4: Key Encapsulation Mechanism and Public Key Encryption Algorithm*
- [9] Cheng Z. Security analysis of SM9 key agreement and encryption. In: *Information Security and Cryptology - Inscrypt 2018*. Springer, **Vol. 11449**, 2019, pp. 3-25 [A<sub>1</sub>]

# British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

## About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards-based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

## Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at [bsigroup.com/standards](https://bsigroup.com/standards) or contacting our Customer Services team or Knowledge Centre.

## Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at [bsigroup.com/shop](https://bsigroup.com/shop), where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

## Copyright in BSI publications

All the content in BSI publications, including British Standards, is the property of and copyrighted by BSI or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use.

Save for the provisions below, you may not transfer, share or disseminate any portion of the standard to any other person. You may not adapt, distribute, commercially exploit or publicly display the standard or any portion thereof in any manner whatsoever without BSI's prior written consent.

## Storing and using standards

Standards purchased in soft copy format:

- A British Standard purchased in soft copy format is licensed to a sole named user for personal or internal company use only.
- The standard may be stored on more than one device provided that it is accessible by the sole named user only and that only one copy is accessed at any one time.
- A single paper copy may be printed for personal or internal company use only.

Standards purchased in hard copy format:

- A British Standard purchased in hard copy format is for personal or internal company use only.
- It may not be further reproduced – in any format – to create an additional copy. This includes scanning of the document.

If you need more than one copy of the document, or if you wish to share the document on an internal network, you can save money by choosing a subscription product (see 'Subscriptions').

## Reproducing extracts

For permission to reproduce content from BSI publications contact the BSI Copyright and Licensing team.

## Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to [bsigroup.com/subscriptions](https://bsigroup.com/subscriptions).

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

**PLUS** is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit [bsigroup.com/shop](https://bsigroup.com/shop).

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email [cservices@bsigroup.com](mailto:cservices@bsigroup.com).

## Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

## Useful Contacts

### Customer Services

**Tel:** +44 345 086 9001

**Email:** [cservices@bsigroup.com](mailto:cservices@bsigroup.com)

### Subscriptions

**Tel:** +44 345 086 9001

**Email:** [subscriptions@bsigroup.com](mailto:subscriptions@bsigroup.com)

### Knowledge Centre

**Tel:** +44 20 8996 7004

**Email:** [knowledgecentre@bsigroup.com](mailto:knowledgecentre@bsigroup.com)

### Copyright & Licensing

**Tel:** +44 20 8996 7070

**Email:** [copyright@bsigroup.com](mailto:copyright@bsigroup.com)

## BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK