INTERNATIONAL STANDARD

ISO/IEC 18033-2

First edition
2006-05-01

AMENDMENT 1
2017-11

Information technology — Security techniques — Encryption algorithms —

Part 2: Asymmetric ciphers

AMENDMENT 1: FACE

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

Partie 2: Chiffres asymétriques

AMENDEMENT 1: FACE

Reference number
ISO/IEC 18033-2:2006/Amd.1:2017(E)

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

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 ISO documents 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).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO 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).

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 voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

A list of all parts in the ISO/IEC 18033 series can be found on the ISO website.

Information technology — Security techniques — Encryption algorithms —

Part 2: Asymmetric ciphers

AMENDMENT 1: FACE

Introduction

Replace the Introduction with the following:

Introduction

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

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights. The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from

— IBM Corporation. Address: North Castle Drive, Armonk, NY 10504 USA,

— NTT Corporation. Address: 9-11 Midori-Cho 3-chome, Musashino-shi, Tokyo 180-8585, Japan, and

— Hitachi, Ltd. Address: 6-1, Marunouchi 1-chome, Chiyoda-ku, Tokyo, 100-8220, Japan.

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/or IEC shall not be held responsible for identifying any or all such patent rights.

ISO (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.

Scope, NOTE

Replace:

— ECIES-HC; PSEC-HC; ACE-HC: generic hybrid ciphers based on ElGamal encryption;

with the following:

— ECIES-HC; PSEC-HC; ACE-HC; FACE-HC: generic hybrid ciphers based on ElGamal encryption;

8.1.2

Replace:

— *ECIES-KEM* (described in Clause 10.2),

— *PSEC-KEM* (described in Clause 10.3),

— *ACE-KEM* (described in Clause 10.4), and

— *RSA-KEM* (described in Clause 11.5).

with the following:

— *ECIES-KEM* (described in 10.2),

— *PSEC-KEM* (described in 10.3),

— *ACE-KEM* (described in 10.4),

— *FACE-KEM* (described in 10.5), and

— *RSA-KEM* (described in 11.5).

8.1.2

Replace NOTE 1 with the following:

NOTE 1 As a matter of convention, the corresponding generic hybrid ciphers built from these key encapsulation mechanisms via the generic hybrid construction in 8.3 should be called (respectively) *ECIES-HC*, *PSEC-HC*, *ACE-HC*, *RSA-HC*, and *FACE-HC*.

Clause 10

Add the following after "*ACE-KEM* is described in Clause 10.4":

— *FACE-KEM* is described in 10.5.

Clause 10

Add the following after 10.4.4:

10.5 *FACE-KEM*

The key encapsulation mechanism *FACE-KEM* is described in 10.5.

NOTE *FACE-KEM* is based on a series of research papers (see References [43] to [46]).

10.5.1 System parameters

FACE-KEM is a family of key encapsulation mechanisms, parameterized by the following system parameters:

— Γ: a concrete group

$$\Gamma = \left(\mathcal{H}, \mathcal{G}, \mathbf{g}, \mu, \nu, \mathcal{E}, \mathcal{D}, \mathcal{E}', \mathcal{D}'\right);$$

— *KDF*: a key derivation function, as described in 6.2;

— *Hash*: a cryptographic hash function, as described in 6.1;

— *CofactorMode*: one of two values: 0 or 1.

— *KeyLen*: a positive integer.

— *TagLen*: a positive integer.

Any combination of allowable system parameters (in 6.1.1, 6.2.1, 10.1.1) is allowed, except for the following restrictions:

— *Hash.len* shall be less than $\log_{256}\mu$.

— If $v=1$, then *CofactorMode* shall be 0.

— If $v>1$, then *CofactorMode* may be 1 provided $\gcd(\mu,v)=1$.

NOTE The value of *CofactorMode* is used only by the decryption algorithm.

10.5.2 Key generation

The key generation algorithm *FACE-KEM.KeyGen* takes no input, and runs as follows.

a) Generate numbers a_1, a_2 uniformly at random from the range $[0..\mu)$.

b) Compute the group elements

$$\mathbf{g}_1 = a_1 \cdot \mathbf{g},\ \mathbf{g}_2 = a_2 \cdot \mathbf{g}.$$

c) Generate numbers x_1, x_2, y_1, y_2 uniformly at random from the range $[0..\mu)$.

d) Compute the group elements

$$\mathbf{c} = x_1 \cdot \mathbf{g}_1 + x_2 \cdot \mathbf{g}_2,\ \mathbf{d} = y_1 \cdot \mathbf{g}_1 + y_2 \cdot \mathbf{g}_2.$$

e) Output the public key $\mathbf{g}_1, \mathbf{g}_2, \mathbf{c}, \mathbf{d}$.

f) Output the private key $x_1, x_2, y_1, y_2 \in [0..\mu)$.

10.5.3 Encryption

The encryption algorithm *FACE-KEM.Encrypt* takes as input a public key, consisting of

$$\mathbf{g}_1, \mathbf{g}_2, \mathbf{c}, \mathbf{d} \in \mathcal{G},$$

together with an encryption option *fmt* that specifies the format to be used for encoding group elements. It runs as follows.

a) Generate a number r uniformly at random from the range $[0..\mu)$.

b) Compute group elements

$$\boldsymbol{u}_1 = r \cdot \mathbf{g}_1,\ \boldsymbol{u}_2 = r \cdot \mathbf{g}_2.$$

c) Compute the octet strings

$$EU_1 = \mathcal{E}(\boldsymbol{u}_1, fmt),\ EU_2 = \mathcal{E}(\boldsymbol{u}_2, fmt).$$

d) Compute the integer

$$\alpha = OS2IP\left(Hash.eval\left(EU_1 \| EU_2\right)\right).$$

e) Compute the integer

$$r' = \alpha r \bmod \mu.$$

f) Compute the group element

$\mathbf{v} = r \cdot \mathbf{c} + r' \cdot \mathbf{d}$.

g) Set $EV = \mathcal{E}(\mathbf{v}, fmt)$.

h) Set $Len = KeyLen + TagLen$.

i) Set $W = KDF(EV, Len)$.

j) Parse W as $W = \langle W_1, \ldots, W_{Len} \rangle$ of Len octets.

k) Set $K = \langle W_1, \ldots, W_{KeyLen} \rangle$ of $KeyLen$ octets.

l) Set $T = \langle W_{KeyLen+1}, \ldots, W_{Len} \rangle$ of $TagLen$ octets.

m) Set $C_0 = EU_1 \| EU_2 \| T$.

n) Output the ciphertext C_0 and the secret key K.

10.5.4 Decryption

The decryption algorithm *FACE-KEM.Decrypt* takes as input a private key, consisting of

$x_1, x_2, y_1, y_2 \in [0..\mu)$,

and ciphertext C_0. It runs as follows.

a) Parse C_0 as $C_0 = EU_1 \| EU_2 \| T$, where EU_1, EU_2 are octet strings such that for some (uniquely determined) group elements $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathcal{H}$, $\boldsymbol{u}_1 = \mathcal{D}(EU_1)$, $\boldsymbol{u}_2 = \mathcal{D}(EU_2)$. This step **fails** if C_0 cannot be so parsed. Check that $\{EU_1, EU_2\}$ is a consistent set of valid encodings; if not, then **fail**.

b) If CofactorMode $= 0$ and $\nu > 1$: test if $\boldsymbol{u}_1 \in \mathcal{G}$ and $\boldsymbol{u}_2 \in \mathcal{G}$; if either $\boldsymbol{u}_1 \notin \mathcal{G}$ or $\boldsymbol{u}_2 \notin \mathcal{G}$, then **fail**.

c) If CofactorMode $= 0$, set

$\hat{\boldsymbol{u}}_1 = \boldsymbol{u}_1$, $\hat{\boldsymbol{u}}_2 = \boldsymbol{u}_2$;

$\hat{x}_1 = x_1$, $\hat{x}_2 = x_2$, $\hat{y}_1 = y_1$, $\hat{y}_2 = y_2$.

d) If *CofactorMode* $= 1$, set:

$\hat{\boldsymbol{u}}_1 = \nu \cdot \boldsymbol{u}_1$, $\hat{\boldsymbol{u}}_2 = \nu \cdot \boldsymbol{u}_2$;

$\hat{x}_1 = \nu^{-1} x_1 \bmod \mu$, $\hat{x}_2 = \nu^{-1} x_2 \bmod \mu$, $\hat{y}_1 = \nu^{-1} y_1 \bmod \mu$, $\hat{y}_2 = \nu^{-1} y_2 \bmod \mu$.

e) Compute the integer:

$\alpha = OS2IP(Hash.eval(EU_1 \| EU_2))$.

f) Compute the integers:

$t_1 = \hat{x}_1 + \alpha \hat{y}_1 \bmod \mu$, $t_2 = \hat{x}_2 + \alpha \hat{y}_2 \bmod \mu$.

g) Compute the group element:

$$\mathbf{v} = t_1 \cdot \hat{\boldsymbol{u}}_1 + t_2 \cdot \hat{\boldsymbol{u}}_2 .$$

h) Set $EV = \mathcal{E}(\mathbf{v}, fmt)$.

i) Set $Len = KeyLen + TagLen$.

j) Set $W = KDF(EV, Len)$.

k) Parse W as $L = \langle W_1, \ldots, W_{Len} \rangle$ of Len octets.

l) Set $K = \langle W_1, \ldots, W_{KeyLen} \rangle$ of $KeyLen$ octets.

m) Set $T_{dec} = \langle W_{KeyLen+1}, \ldots, W_{Len} \rangle$ of $TagLen$ octets.

n) Test if $T_{dec} = T$; if not then **fail**.

o) Output the secret key K.

Annex A

Replace the title:

ASN.1 syntax for object identifiers

with the following:

Object identifiers

Annex A

Replace the first paragraph:

This annex gives ASN.1 syntax for object identifiers, public keys, and parameter structures to be associated with the algorithms specified in this part of ISO/IEC 18033.

with the following:

Annex A gives object identifiers, public keys, and parameter structures to be associated with the algorithms specified in this document.

Annex A

Replace:

```
-- Key encapsulation mechanisms --
id-kem-ecies OID::= { id-kem ecies(1) }
id-kem-psec OID::= { id-kem psec(2) }
id-kem-ace OID::= { id-kem ace(3) }
id-kem-rsa OID::= { id-kem rsa(4) }
```

with the following:

```
-- Key encapsulation mechanisms --
id-kem-ecies OID::= { id-kem ecies(1) }
```

```
id-kem-psec OID::= { id-kem psec(2) }
id-kem-ace OID::= { id-kem ace(3) }
id-kem-rsa OID::= { id-kem rsa(4) }
id-kem-face OID::= { id-kem face(5) }
```

Annex A

Replace:

```
-- KEM information objects
KeyEncapsulationMechanism::= AlgorithmIdentifier {{ KEMAlgorithms }}
KEMAlgorithms ALGORITHM::= {
{ OID id-kem-ecies PARMS EciesKemParameters } |
{ OID id-kem-psec  PARMS PsecKemParameters }  |
{ OID id-kem-ace   PARMS AceKemParameters }   |
{ OID id-kem-rsa   PARMS RsaKemParameters },
... -- Expect additional algorithms --
}
```

with the following:

```
-- KEM information objects
KeyEncapsulationMechanism::= AlgorithmIdentifier {{ KEMAlgorithms }}
KEMAlgorithms ALGORITHM::= {
{ OID id-kem-ecies PARMS EciesKemParameters } |
{ OID id-kem-psec  PARMS PsecKemParameters }  |
{ OID id-kem-ace   PARMS AceKemParameters }   |
{ OID id-kem-rsa   PARMS RsaKemParameters }   |
{ OID id-kem-face  PARMS FaceKemParameters },
... -- Expect additional algorithms --
}
```

Annex A

Add the following before ~"-- DEM specifications":

```
-- FACE-KEM
-- all components of public key are elements of the group given in
-- FaceKemParameters
FaceKemPublicKey::= SEQUENCE {
g1 FieldElement,
g2 FieldElement,
c  FieldElement,
d  FieldElement
}
FaceKemParameters::= SEQUENCE {
group Group OPTIONAL,
keyDerivationFunction KeyDerivationFunction,
hashFunction HashFunction,
keyLength KeyLength,
tagLength TagLength
}
--################################################################
```

Annex B

Add the following after B.15:

B.16 Security of *FACE-KEM*

The key encapsulation mechanism *FACE-KEM*, defined in 10.5, has the following security properties.

Recall that *FACE-KEM* is parameterized by a group Γ (see 10.1), a hash function *Hash* (see 6.1), and a key derivation *KDF* (see 6.2).

FACE-KEM can be proven secure against adaptive chosen ciphertext attack, assuming that the DDH problem (defined in B.8.2) is hard for group Γ. The security proof does not make use of the random oracle model. Instead, in the security proof, *Hash* is assumed to be secure against second preimage collision attacks, and *KDF* is assumed to transform the encoding of a random group element to a random octet string. For the security proof of *FACE-KEM*, see Reference [44] (indirect security reduction to the DDH problem), or Reference [46] (direct security reduction to the DDH problem).

Annex C

Replace all the following words in Annex C:

Test vector(s)

with the following:

Numerical example(s)

Annex C

Add the following after C.8.6:

C.9 Numerical examples for *FACE-KEM*

C.9.1 Numerical examples over elliptic curve P224

```
-------------
FACE-KEM
-------------
Kdf = Kdf2(Hash = Sha256(outlen = 20 octets))
Hash = Sha256(outlen = 20 octets)
CofactorMode = 0
KeyLen = TagLen = 16 (octets)
-----
Group = ECModp-Group:
p = 0xffffffffffffffffffffffffffffffff000000000000000000000001
a = 0xfffffffffffffffffffffffffffffffefffffffffffffffffffffffe
b = 0xb4050a850c04b3abf54132565044b0b7d7bfd8ba270b39432355ffb4
mu = 0xffffffffffffffffffffffffffff16a2e0b8f03e13dd29455c5c2a3d
nu = 0x01
g(x) = 0xb70e0cbd6bb4bf7f321390b94a03c1d356c21122343280d6115c1d21
g(y) = 0xbd376388b5f723fb4c22dfe6cd4375a05a07476444d5819985007e34
-----
Numbers to generate g1 and g2 in key generation
a1 = 0xdc32d5babd0d3753ca5f7ff8be59f4d6c49168b8f4c6e3b59317ba44
a2 = 0x81314bb3154af84a7489c95bb9fcca0edea88a8a0779497092677d46
-----
Public Key
g1(x) = 0x73e451f448f7c473e436f394de7ddf7a562af2f6cb0a1aa7d2d38f51
g1(y) = 0x892910a01bbd1a6c7995d79c29e72d21ed37112143ab55375bb9a29c
g2(x) = 0x57fc06cf338f547227f1275fe8c11055d73feba2a3eae9245c95d7c5
```

```
g2(y) = 0x7af7c4319f8cf998bf3fc2d48be46d8b527f08e02c8dec7589b23350
c(x) = 0xb4baaf147b33bc50c8f2d20555fb7315cd139332f89a1ada6e7ce53f
c(y) = 0xff46b1bf4d2c2fbe8ae0a34b4f696cfb6b3bc90a4a236a8be0673827
d(x) = 0x9eb266244b51847f0a2eb665e130f5d695a99f8ddd040819d39e7b92
d(y) = 0x14032c241b51f7d40b7612406694fa583d3bfb35eba5b326455a090f
-----
Private Key
x1 = 0xc718604eb67048c28d2d26a7400144b8eb64a4f31be041a4970e00ec
x2 = 0x6c613adf8b9b6fdc083d4ac64f5bec376eae02edae2e9b53bbda98f6
y1 = 0xf179878e0f7ef84d47753bf4ba7a497acae0833c3ed25aa3d15aebae
y2 = 0xc099fb374680995897ecb2c933c47a79f853c4397a2c2e66d09c0da4
----------------------------
Trace for FACE-KEM.Encrypt
----------------------------
Encoding format = uncompressed_fmt
r = 0x453109403b913bd9e1ca9498948f942c8b5e97394e74ffa2b196e8a0
u1(x) = 0x686c7d062e31a49433dec25470228a5f3e101f7b48ae967426e76966
u1(y) = 0x0385cd57a1fc4faf04e2ee791f5fa9fa33d6046f40fb0ea01511e02b
u2(x) = 0x13a4d81283775e9cda6381f42a930cddda3b9e2ed054e0949378c74f
u2(y) = 0x1f1e78ba5a8988a3e37ca923e7fc6b3002cf3d0505353d20d62327b6
EU1 =
0x04686c7d062e31a49433dec25470228a5f3e101f7b48ae967426e769660385cd57a1fc4fa
f04e2ee791f5fa9fa33d6046f40fb0ea01511e02b
EU2 =
0x0413a4d81283775e9cda6381f42a930cddda3b9e2ed054e0949378c74f1f1e78ba5a8988a
3e37ca923e7fc6b3002cf3d0505353d20d62327b6
alpha = 0xe90304e9eab627d21bd80996bbd70e8d9a70861b
r_dash = 0x116bd69f5aeae58c3798ce9e9a99f223c57a28d64c648808943e6810
v(x) = 0xc0d9e1c8c97e694de6f856b5853732038c10de30e79a92028da3b978
v(y) = 0x822e6aa51e1ebf5a29622988798e01332995ea3a2f6870412ddbadc9
EV =
0x04c0d9e1c8c97e694de6f856b5853732038c10de30e79a92028da3b978822e6aa51e1ebf5
a29622988798e01332995ea3a2f6870412ddbadc9
W = 0xc43cf57936c5b1fc6d957a5106d8f61376792f5bb1cefbb315f79d214712a15f
K = 0xc43cf57936c5b1fc6d957a5106d8f613
T = 0x76792f5bb1cefbb315f79d214712a15f
C0  =
0x04686c7d062e31a49433dec25470228a5f3e101f7b48ae967426e769660385cd57a1fc4fa
f04e2ee791f5fa9fa33d6046f40fb0ea01511e02b0413a4d81283775e9cda6381f42a930cdd
da3b9e2ed054e0949378c74f1f1e78ba5a8988a3e37ca923e7fc6b3002cf3d0505353d20d62
327b676792f5bb1cefbb315f79d214712a15f
```

C.9.2 Numerical examples over elliptic curve B163

```
-------------
FACE-KEM
-------------
Kdf = Kdf2(Hash = Sha256(outlen = 20 octets))

Hash = Sha256(outlen = 20 octets)

CofactorMode = 0
KeyLen = TagLen = 16 (octets)
-----

Group=ECGF2-Group:

p = 0x080000000000000000000000000000000000000c9

a = 0x01

b = 0x020a601907b8c953ca1481eb10512f78744a3205fd

mu = 0x040000000000000000000292fe77e70c12a4234c33

nu = 0x01

g(x) = 0x03f0eba16286a2d57ea0991168d4994637e8343e36

g(y) = 0xd51fbc6c71a0094fa2cdd545b11c5c0c797324f1
```

```
-----
Numbers to generate g1 and g2 in key generation
a1 = 0x015897ecb2c932fa1bb876e25442682b342fab391c

a2 = 0x0353cedb56d6129658a9c208427a79756979ffa1f2

-----

Public Key

g1(x) = 0x05cf2e1de9dcf32160bef47df954851b52a226f463

g1(y) = 0x06c65878cff713a57fa53bbfc87497ac73067ed3aa

g2(x) = 0x034115a8459671a752b8be5926ac1f604983cc8e45

g2(y) = 0x06ba7e233b76dc98ab9adad1e320c62a29690e52c1

c(x) = 0x03cd12b6bf02ec9f36885a6d6d45eea5a2c6753c53

c(y) = 0x0464d1b820fb17f9b943c12fca6385d799b891d8b8

d(x) = 0x0682142c7a07e7e445ca2c48aca4e9d46bab195821

d(y) = 0x05067a81d6cb789c8bd443fe8e416c706eea7bb435

-----

Private Key

x1 = 0x028d2d26a73f713d3f9d0d5b8ce30d76f4d151c902

x2 = 0xa9836a84a1583f601a2f9b2b2432a0aff42c84e8

y1 = 0x02140a3d998770496c5cbec836b6e8d38e47cc0575

y2 = 0x02f179878e0f7ef84d45966f119bc634d0f246beec


----------------------------
Trace for FACE-KEM.Encrypt
----------------------------
Encoding format = uncompressed_fmt

r = 0x010c6028d090fa88fdd82d281f640a5a3353387048

u1(x) = 0x02f0e6e40244de3232377911ea47cc95d73b4512c6

u1(y) = 0x9fa93f1fb1d81ba29db4d29071506eaaa0fa2def

u2(x) = 0x51d260249605e811007536a7ec3520d9e3a1566f

u2(y) = 0xdb2f64fee47dac599f3744e739fc3a45b21db7d2

EU1 = 0x0402f0e6e40244de3232377911ea47cc95d73b4512c6009fa93f1fb1d81ba29db4d290715
06eaaa0fa2def

EU2 = 0x040051d260249605e811007536a7ec3520d9e3a1566f00db2f64fee47dac599f3744e739f
c3a45b21db7d2

alpha = 0x36d1facc466a738aaa09fe33e4cdf4983eb0c8d1

r_dash = 0x036f3c8b855b8d2d3b30f8f6748ea0229ec5a25c03

v(x) = 0x050dea8e376f2e31c714e59a07ec02d5d17df9a5e7

v(y) = 0x0140402b886f4969289ce1c31ccd82f9f492a41e9a

EV = 0x04050dea8e376f2e31c714e59a07ec02d5d17df9a5e70140402b886f4969289ce1c31ccd8
2f9f492a41e9a
```

```
W = 0x3ee707aec1ab5f0435d8e0e0c0d4d107f4343213cde66426b98a3ce7e91cf302

K = 0x3ee707aec1ab5f0435d8e0e0c0d4d107

T = 0xf4343213cde66426b98a3ce7e91cf302

C0  =  0x0402f0e6e40244de3232377911ea47cc95d73b4512c6009fa93f1fb1d81ba29db4d290715
06eaaa0fa2def040051d260249605e811007536a7ec3520d9e3a1566f00db2f64fee47dac59
9f3744e739fc3a45b21db7d2f4343213cde66426b98a3ce7e91cf302
```

C.9.3 Numerical examples over Zp group

```
-------------
FACE-KEM
-------------
Kdf = Kdf2(Hash = Sha256(outlen = 20 octets))

Hash = Sha256(outlen = 20 octets)

CofactorMode = 0
KeyLen = TagLen = 16 (octets)
-----

Group=Modp-Group:

p =
0xa35178c0f9b33e25e6d473a41bcfc1c9bb38182821d16a25de75b75b81b71c4cb9f245590
976fc2c4d62dd2dfc2973dd6131cc84ccb0cd75d80b7e802bb7537b591ef4234989471e0dd
f6577e35b28140fc13f97c925a97d1bad5b9946a0bf80b097f5f10cf134a395b4af80b949e
7c1f02c3df831fec9fbf4c18b617b8513

g =
0x06bb7bb2b9c9218947602a11c58a91a39b28eaa73057765cfb1dc14e563f21d5209583a6f
68b4ac7d2d7a42e4cd6796437841c4746f27a098ef6a26455410003e54f25b3d252b8f35569
b0017496d60829609382a951fb19bf471a9674d8d4418df1563244ad709de93386e9d29ca66
e9bb6ea173009b8a1d4502d7c7f461dd5

mu =
0x51a8bc607cd99f12f36a39d20de7e0e4dd9c0c1410e8b512ef3adbadc0db8e265cf922ac8
4bb7e1626b16e96fe14b9eeb098e642665866baec05bf4015dba9bdae48f7a11a4c4a38f06e
fb2bbf1ad940a07e09fcbe492d4be8dd6adcca3505fc0584bfaf8836789a51cada57c05ca4f
3e0f8161efc18ff64fdfa60c5b0bdc289

nu = 0x02

-----
Numbers to generate g1 and g2 in key generation

a1 =
0x306e82cd2471c56a70a3522d2f51021d862ce87b1b55da895a4ed2dfa2bb7751e4472e320
2052413a250289387216fca1e5b7a140fdcb00ee01e20acc79fc0af0f33b87b025061db163b
f4c1b9f973bf73540281d9761f807644c35b35f0adbe9873ab9600f62fc37f0c7dff99cef3f
d6695ed5be70c0bc6e712710f520b310a

a2 =
0x06c4410333e2b6531346ed3e1e093aa15d8305e169655c0f6c42131c070f44eb98d46fde8
2ef858c8af1074cef1c22b612b9b27ea206ea60ebed82e57e67e80b562fb4a8f15cbcd1b7b5
70eb9b4ed9ecbcce59271204e03411c64b7f849240ae83158a72274957d997e7e5489f352c0
b58b586d2cf278ae4abf2866902b75733

-----

Public Key

g1 =
0x84e36daa032b03469efcacce74f5efe02aca71e5ea4499e0e9bfca4d4acdf94d52daa36ca
1b3cfde68f2fa22a0b4fe633e83a60f73bd4eac21e806dbc61c47fb71533890a24b83e3d5e7
a82a54e07fc06bf83fa90f08147ae587e12ad276bfcccc1f7d7c194fe45edc02806a84a46c1
efda06394ed4a49813a8d3b5541d42441
```


