INTERNATIONAL STANDARD

ISO/IEC 18033-6

First edition
2019-05

IT Security techniques — Encryption algorithms —

Part 6: Homomorphic encryption

Techniques de sécurité IT — Algorithmes de chiffrement —

Partie 6: Chiffrement homomorphe

Reference number
ISO/IEC 18033-6:2019(E)

Contents

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.

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).

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) or the IEC list of patent declarations received (see http://patents.iec.ch).

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

For an explanation of 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 www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee 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.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Homomorphic Encryption is a type of symmetric or asymmetric encryption that allows third parties (i.e. parties that are neither the encryptor nor the decryptor) to perform operations on plaintext data while keeping the data in encrypted form. The primary purpose of homomorphic encryption is to allow third parties to perform such computations on data while simultaneously ensuring that the confidentiality of the plaintext data is preserved. It is typically the case that homomorphic encryption schemes require the plaintext to be represented in the form of elements of a group, rather than strings of bits or bytes as is the case with most conventional methods of encryption.

Homomorphic encryption mechanisms can be categorized by the nature of the operation(s) on the plaintext that they can support. This document considers homomorphic encryption mechanisms where the plaintext operation is typically addition and/or multiplication in a prescribed group.

IT Security techniques — Encryption algorithms —

Part 6:
Homomorphic encryption

1 Scope

This document specifies the following mechanisms for homomorphic encryption.

— Exponential ElGamal encryption;

— Paillier encryption.

For each mechanism, this document specifies the process for:

— generating parameters and the keys of the involved entities;

— encrypting data;

— decrypting encrypted data; and

— homomorphically operating on encrypted data.

Annex A defines the object identifiers assigned to the mechanisms specified in this document. Annex B provides numerical examples.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
ciphertext
data which has been transformed to hide its information content

[SOURCE: ISO/IEC 18033-1:2015, 2.11]

3.2
decryption
reversal of a corresponding *encryption* (3.6)

[SOURCE: ISO/IEC 10116:2017, 3.5]

3.3
decryption algorithm
process which transforms *ciphertext* (3.1) into *plaintext* (3.14)

[SOURCE: ISO/IEC 18033-1:2015, 2.17]

3.4
decryptor
entity which decrypts *ciphertexts* (3.1)

[SOURCE: ISO/IEC 18033-5:2015, 3.1]

3.5
deterministic
<algorithm> characteristic of an algorithm that states that given the same input, the same output is always produced

[SOURCE: ISO/IEC 18031:2011, 3.9, modified — "algorithm" has been removed from the term and added as the domain.]

3.6
encryption
(reversible) transformation of data by a cryptographic algorithm to produce *ciphertext* (3.1), i.e. to hide the information content of the data

[SOURCE: ISO/IEC 18033-1:2015, 2.21]

3.7
encryption algorithm
process which transforms *plaintext* (3.14) into *ciphertext* (3.1)

[SOURCE: ISO/IEC 18033-1:2015, 2.22]

3.8
encryptor
entity which encrypts *plaintexts* (3.14)

[SOURCE: ISO/IEC 18033-5:2015, 3.2]

3.9
group
set of elements S and an operation * defined on the set of elements such that (i) $a*(b*c) = (a*b)*c$ for every a, b and c in S, (ii) there exists an identity element e in S such that $a*e = e*a = a$ for every a in S, and (iii) for every a in S there exists an inverse element a^{-1} in S such that $a*a^{-1} = a^{-1}*a = e$

[SOURCE: ISO/IEC 15946-1:2016, 3.6]

3.10
homomorphic map
map from one *group* (3.9) to another that preserves their respective group operations

Note 1 to entry: A definition of homomorphic map is provided by Cohen et al. in [13].

3.11
key
sequence of symbols that controls the operation of a cryptographic transformation

Note 1 to entry: Examples are *encryption* (3.6), *decryption* (3.2), cryptographic check function computation, signature generation, or signature verification.

[SOURCE: ISO/IEC 9798-1:2010, 3.16]

3.12
key generation
process of generating a *key* (3.11)

[SOURCE: ISO/IEC 11770-1:2010, 2.24]

3.13
key generation algorithm
method for generating asymmetric *key* (3.11) pairs

[SOURCE: ISO/IEC 18033-2:2006, 3.27]

3.14
plaintext
unencrypted information

[SOURCE: ISO/IEC 18033-1:2015, 2.30]

3.15
probabilistic
<algorithm> characteristic of an algorithm that states that given the same input, the output could take different values

3.16
security parameter
variables that determine the security strength of a mechanism

[SOURCE: ISO/IEC 20008-2:2013, 3.5]

4 Symbols and abbreviations

$a \in S$	Element a of the set S
sec.key	Private key (secret key)
pub.key	Public key
F_p	Finite field with p elements for a prime p
g	Element in F_p
k	Security parameter
p	Prime number
parameters	Public parameters necessary for encryption, decryption or the group operation on ciphertexts
q	Prime order of g
Z_q^* or Z_n^*	Unit group of Z_q or Z_n, respectively
Z_q or Z_n	Residue ring modulo q or n, respectively
(mod p)	Modulo p
$\bullet$	Operation on the plaintext group
$\odot$	Operation on the ciphertext group
$<g>$	Group generated by g

5 General model for homomorphic encryption

5.1 Entities

There are three entities as follows.

— encryptor: an entity that performs homomorphic encryption using a public key;

— decryptor: an entity that performs homomorphic decryption using a private key;

— operator: an entity that performs homomorphic operations on ciphertexts.

5.2 Key roles

The private key *sec.key* shall be kept secret by the decryptor.

The public key *pub.key* shall be public to the encryptor or operator.

The parameters *parameters* are public.

5.3 Algorithms

A homomorphic encryption mechanism is composed of the following three algorithms.

— KeyGen(k). Given a security parameter k, produce a tuple (*pub.key*, *sec.key*, *parameters*) where *pub.key* denotes the public key, *sec.key* denotes the private key and *parameters* denotes the parameters.

— Encrypt(*m*, *pub.key*, *parameters*). Given a public key *pub.key*, parameters *parameters* and a plaintext m in the plaintext group, perform encryption and produce a ciphertext c.

— Decrypt(*c*, *sec.key*, *parameters*). Given a private key *sec.key*, parameters *parameters* and a ciphertext c in the ciphertext group, perform decryption and produce a plaintext m.

5.4 Functional requirements

Given any tuple (*pub.key*, *sec.key*, *parameters*) produced by KeyGen(k), the following two properties are required.

Correctness. For any plaintext m,

Decrypt(Encrypt(*m*, *pub.key*, *parameters*), *sec.key*, *parameters*) = m .

Homomorphic property. The encryption is a homomorphic map from the plaintext group to the ciphertext group. More specifically, for any two plaintexts m_1 and m_2 in the plaintext group, and letting

$$c_1 = \mathrm{Encrypt}(m_1, \mathit{pub.key}, \mathit{parameters})$$

$$c_2 = \mathrm{Encrypt}(m_2, \mathit{pub.key}, \mathit{parameters}),$$

it is required that

$$\mathrm{Decrypt}(c_1 \odot c_2, \mathit{sec.key}, \mathit{parameters}) = m_1 \bullet m_2.$$

In all the mechanisms specified in this document, the key generation and encryption algorithms are probabilistic, while the decryption is a deterministic algorithm.

6 Homomorphic encryption mechanisms

6.1 General

In Clause 6, two homomorphic encryption mechanisms are specified.

Annex A defines the object identifiers which shall be used to identify the mechanisms specified in this document.

6.2 Exponential ElGamal encryption

6.2.1 General

The detailed algorithm is found in [14].

6.2.2 Key generation algorithm

Key generation: KeyGen(k) →($pub.key$, $sec.key$, $parameters$)

Input: a security parameter k.

Output: a public key $pub.key = y$, a private key $sec.key = x$, and parameters $parameters = (p, q, g)$.

Operations:

a) Parameters' key generation

  1) Choose prime q with security parameter k uniformly at random and independently.

  2) Choose prime p uniformly at random with security parameter k subject to the condition that q divides p-1.

  3) Choose $g \in F_p^*$ with prime order q.

b) User key generation

  1) Choose $x \in \{1,..., q\text{-}1\}$ uniformly at random.

  2) Compute $y = g^x (\text{mod } p)$.

3) Output (y, x, (p, q, g)).

NOTE 1 For the common security levels and corresponding sizes for p and q, see [11].

NOTE 2 For generating a random integer from the specified range, see ISO/IEC 18031.

NOTE 3 For prime number generation, see ISO/IEC 18032.

6.2.3 Encryption

Encryption: Encrypt(m, $pub.key$, $parameters$) → c

Input: a message $m = g^M \in \langle g \rangle$ for $M \in Z_q$, a public key $pub.key = y$, and parameters $parameters = (p, q, g)$.

Output: a ciphertext $c = (u, v)$.

Operations:

a) Choose r uniformly at random from Z_q^*.

b) Compute $u = g^r (\text{mod } p)$.

c) Compute $v = my^r(= g^M y^r)(\text{mod } p)$.

d) Output c as a ciphertext $c = (u, v)$ of m.

NOTE When a message is used after a conversion function, see ISO/IEC 18033-2.

6.2.4 Decryption

Decryption: Decrypt(c, *sec.key*, *parameters*) $\rightarrow m = g^M$

Input: a ciphertext $c = (u, v)$, a private key *sec.key* $= x$, and parameters *parameters* $= (p, q, g)$.

Output: exponential message $m = g^M$.

Operations:

a) Compute $z = u^x(\text{mod } p)$.

b) Decrypt the ciphertext as $m = vz^{-1}(\text{mod } p)$, where $m = g^M \in <g>$.

The scheme has the homomorphic property with respect to the following two group operations:

— The operation • on plaintexts is defined by a multiplication on $<g>$.

— The operation ⊙ on ciphertext is defined by coordinate-wise multiplication modulo p.

NOTE 1 A homomorphic property is satisfied as follows:

Encrypt(m_1, *pub.key*, *parameters*) ⊙ Encrypt(m_2, *pub.key*, *parameters*)

$= (u_1, v_1) \odot (u_2, v_2)$

$= (u_1u_2(\text{mod } p), v_1v_2(\text{mod } p))$

= Encrypt($m_1 \bullet m_2$, *pub.key*, *parameters*).

NOTE 2 The size of p and q is determined by the security parameter k.

NOTE 3 If m is represented by $m = g^M$, then an additive homomorphic property for M is satisfied as follows:

Encrypt(g^{M_1} , *pub.key*, *parameters*) ⊙ Encrypt(g^{M_2} , *pub.key*, *parameters*)

$= (u_1u_2(\text{mod } p), v_1v_2(\text{mod } p))$

= Encrypt($g^{M1+M2(\text{mod } q)}$, *pub.key*, *parameters*),

where $M_1+M_2(\text{mod } q)$ is an addition over F_q.

Although M cannot be recovered, it is possible that it is not necessary to get M but just to check whether two ciphertexts relate to the same M or not while keeping an additive homomophic property on M. Because it is computationally inexpensive, the Exponential ElGamal encryption scheme is particularly well suited to address that common case.

NOTE 4 In practical applications, such as electronic elections (see [12]), the value of M is small, so M can be recovered from $m = g^M$ with precomputed tables.

6.3 Paillier encryption

6.3.1 General

The detailed algorithm is found in [15].

6.3.2 Key generation algorithm

Key generation: KeyGen(k) → ($pub.key$, $sec.key$, $parameters$)

Input: a security parameter k.

Output: public key $pub.key = n$, private key $sec.key = \lambda$, and parameters $parameters = n$.

Operations:

a) Choose prime numbers p and q independently and appropriately at random from the appropriate range, which depends on a security parameter k. Both p and q shall be secret.

b) Compute $n = pq$.

c) Compute $\lambda = \mathrm{lcm}(p\text{-}1, q\text{-}1)$, which is the least common multiple of p-1and q-1.

d) Output n and λ.

e) Let $d = \lambda^{-1}(\mathrm{mod}\ n)$.

NOTE 1 n+1 is invertible with order n modulo n^2. The value n is both the public key and the only parameter. The value λ is the private key.

NOTE 2 A security parameter k means that 2^k operations are the best known cryptanalysis.

6.3.3 Encryption

Encryption: Encrypt(m, $pub.key$, $parameters$) →c

Input: a message $m \in Z_n$, a public key $pub.key = n$, and parameters $parameters = n$.

Output: a ciphertext c.

Operations:

a) Choose r uniformly at random from Z_n^*.

b) Compute $c = (nm + 1)r^n(\mathrm{mod}\ n^2)$.

c) Output c.

NOTE When a message is used after a conversion function, see ISO/IEC 18033-2.

6.3.4 Decryption

Decryption: Decrypt(c, $sec.key$, $parameters$) →m

Input: a ciphertext c, a private key $sec.key = \lambda$ and parameters $parameters = n$.

Output: plaintext m.

Operations:

a) Compute $L = (c^\lambda(\mathrm{mod}\ n^2)-1)/n$.

b) Decrypt $m = L\lambda^{-1}(\mathrm{mod}\ n)$.

c) Output m.

The scheme has the homomorphic property with respect to the following two group operations.

— The operation + on plaintexts Z_n is defined by an addition over Z_n.

— The operation ⊙ on ciphertexts Z_{n^2} is defined by a multiplication over Z_{n^2} .

NOTE A homomorphic property is satisfied as follows:

Encrypt(m_1, *pub.key*, *parameters*) ⊙ Encrypt(m_2, *pub.key*, *parameters*)

$= ((nm_1+1)r_1{}^n)\ (\ (nm_2+1)r_2{}^n)\ (\mathrm{mod}\ n^2)$

$= (n(m_1+ m_2)+1)\ (r_1\ r_2)^n(\mathrm{mod}\ n^2)$

= Encrypt($m_1 + m_2$, *pub.key*, *parameters*).

Annex A
(normative)

Object identifiers

This annex lists the object identifiers assigned to the mechanisms specified in this document.

```
HomomorphicEncryption { iso(1) standard(0) encryption-
algorithms(18033) part6(6) asn1-module(0) homomorphic-encryption-
mechanisms(0) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN
-- EXPORTS All; ---- IMPORTS None; --
id-homenc-mechanisms OBJECT IDENTIFIER ::= { iso(1) standard(0)
encryption-algorithms(18033) part6(6) mechanisms(1) }
id-homenc-expElGamal OBJECT IDENTIFIER ::= { id-homenc-mechanisms 1 }
id-homenc-pailler    OBJECT IDENTIFIER ::= { id-homenc-mechanisms 2 }
END
```

Annex B
(informative)

Numerical examples

B.1 Exponential ElGamal encryption

B.1.1 General

The parameters p, q, g, and k satisfy the requirements specified in [14].

B.1.2 1 024-bit finite field, 160-bit security parameter, 2-party

B.1.2.1 Key generation

p	e5256a78 8f875183 ec56a332 d38db31d e883cded 25ae635a 656823b5 c801b44a 104f4e1d 604153ad aaa5d6d1 07feb3a8 e721a32f 3e678064 5c85de2d 4f4f8556 8767efc9 b8363193 497c052a 5b832464 b81a209d 393eb6d3 a464cba0 b7607dc7 9b3611dc d1544e4e d329cc91 3f68234b 1d5f209a e7081c0d 44662ee1 f86c458f
q	e6fa5be8 dfd1a200 fd699a9f f4b02761 f05fca69
g	5a0c1ebd e9c0787f 3d426e20 36455fcd 25bc32b1 e666b2ba 90dad169 af7043c1 8b266d53 0d0f607e a46c182d d7c88d91 91583434 41e001b1 0e36c8ff a03cb80d adcf7e84 393561d2 f4f2d067 222d5a33 157b81f4 f4a46c95 26375920 cac73c23 e100e8b4 3eb8a4bc 83047ae4 5b079bca 6dbf69b4 b0c1e6bf fdfd232b 99c5d61a
x	13d5955 a5e91b8f ed1b56b6 bdcd4679 39de9bfc
y	b7866990 d044b1bc cbbcf84c 29f145ee 17d4f460 8c79a55e 249e9e10 8b91e363 81944fa3 c0c3f518 76f63bce 7bb30ffd e9ca0226 5e916dd3 fb2e060b 0dfeaaa6 7d5a3591 59b948c3 df1141f0 e0a22380 a3633c1f fbcb1c22 8ffe4ef0 bab52293 bfdff4b6 4e3f362d 63b11a4d 2507f6e9 e98de71a ff09fdb6 4e3737c0 46044138

B.1.2.2 Encryption and decryption

M_1	41424344 45464748 494a3132 33343536 37383930
g^{M_1}	de318d95 40168f7c bd4e8735 8c2f5b2a 9c541328 ad51cb31 57f8b9ad e4b61e08 0184d277 9ebfb79d 70ec0a07 5aa70b0c eb34e418 ba063c53 b6724cfc 5675c2ce e91cddd7 9ad3bc13 60f78a38 94c3f92c 0523e3fa 694e2bd7 49344676 a118be58 37e676ca 882b5b14 e274b44f 925f4160 e119e9d6 774261c3 676bb36e e47a547f
r_1	d8d0f2d6 a3e2d745 ab4410e2 042a740e 7a81a280

u_1

```
a0fffc52 73b8c1fa 9a9a9395 204c4eb2 1a8f1e11 8bb2c0d1
4c869dc6 c7de5f29 131c056d aeb4a01b bb3e2178 504032bc
8a0154c8 2686eef8 973c28d2 273fe1f6 3fce8e95 161ceb85
b6041dd3 c86ce6bd 38cb96ec e40e352d da486582 8b069227
f676273c 6b0e98f9 a7320c35 9c6621ce 0a7afd3e caf85fdb
5475698b cea30009
```

v_1

```
351c6cde 7a2e2b3e 1041a8f9 36a382ba 54968a02 db32ac28
6e46e839 baac8854 d7be712d 0277de9c 3686028f 4180502a
475d3ce8 3bdd8717 4ed4f97f 52416914 f6bd6de4 a1dedd55
3b7985d9 034c4df4 cf413d19 74d7f359 66e75247 202edba9
0cd467cd d01f3601 c3a479a9 fc0247e4 390d59bc 1eb32865
a6372c17 197f1de6
```

z_1

```
8825ad4b 48512e4e 51cd4f12 d46eabdf f53fec37 27cb587a
52e6a653 887d9ebd 7eb74efe 79950731 f640e1a4 cb623e0f
a43ebf09 f61789d0 7f24a133 72f04ea6 4bf17c7e 0213358a
5076d19a 4ebe119e 3651ba73 0d95b23b 7dd3a441 2c1074f9
50e64efe a33759bb 5ff215a6 92e26d2a 49b47e57 bf00cce5
50ce4068 169307ce
```

M_2

```
61626364 65666768 696a6b6c 6d6e6f70 71727374
```

g^{M_2}

```
63c27e18 2de87a93 4e2d549d bbd20abd f9e2b9ad 9eadb78d
ba7970e5 3defc949 7465200a 6a4d0580 af7afd52 642d2442
7f28a54a 54be2fdc e6d6db37 aed00813 1fa9670e 5e38581e
7b5841a0 924349ea c936e7ea 5211c5f8 752021d9 8e0c19c1
a440c5d2 e830a7a5 a18de021 1acdda2d d2fead6d 90beaf72
a87571bf 65f510be
```

r_2

```
8706c408 ce1c9cc8 bfb1cc80 fc9f1b9c 4658f6eb
```

u_2

```
50886d4a 535fe180 3f2d4bae 5396719c be11f162 cbf7b1d6
3d59b53d 5a8572d7 51e64c8f bd619859 8e693f8b 8a5dc9b1
2ac27977 f6f2b375 47d46ac4 5c04ae44 c6c0dcf1 051e0f73
d412e801 02313d82 84711dd3 b6061ea4 d42d44b9 1b8df05e
54e29872 925a6236 50e57011 376490e0 1d2f3290 c2b29ba2
04a31931 ca1adffc
```

v_2

```
91d7ea1 75faacf2 909ce456 f8c76ed3 7b7b6848 6ab1d9eb
cc9cf90a 0bfcc778 99c9eb1f 7e2012dd 0ce9c1ee 1189cfc4
cc6782f2 8e751564 209b9209 4bf0e682 835d7a0f 8042ffb1
27078b10 baa31fea dd648101 a07c6d49 95254fde 834be18a
15dc520b 02457145 8c90032b c688566a 75f2787e db5a3688
b95fbe1e bcc75353
```

z_2

```
3fe39e4f 6a5505b9 a9d9e42d 4ae820db ef1a8ba2 a1f518f0
c7420b61 eaa08205 975e3632 27c28cbe 0a15d961 fd9efb30
21743c28 1d9f811a e025858a ae4e15bd 7c0c78f3 579f785b
d7aa997f 65de67df 2c791aa7 2c73510a 444f1921 3defb7cb
0b3ec72f 5feb1348 5deae59f 503d8d55 f27d5e98 50c1eaa7
cb7ce945 3ff1c87d
```

B.1.2.3 Homomorphic map

$g^{M_1} g^{M_2} = g^{(M_1+M_2)}$

```
484d5d48 744e5241 961fe498 89b5a0fe cf262c6f 6514b8b8
7d3dc757 4608d422 b13a05f6 d6ff8012 846d7ca4 ee8de47b
e5446fb2 84d4973a 22a3c924 ba79edf9 124fc15e e5fc283d
1665fa54 28994bc5 09a16bf1 61bb5892 53cc1754 57313cf5
8f5a12aa 7ece8b20 1955f52f 293d723f a2129ece 45e44b7c
b777f7f4 f3888d80
```

$u_1 u_2$

```
8b4a47ea 5d4183b1 799202b7 6ae15abd 8a35a5f5 13876412
4e5ceeb1 f653928a d65c1a3f ff458dec 74237006 85181453
b691b569 98a6ad1c 2b5bf057 8606aff7 d3c656d9 b50833e9
8b71fd20 c5bb097b 7cb22d77 af655f54 cc1b97eb 5b61d8e6
ea810dba 0d068ac9 219be853 2f0bc699 3321f2f2 0b016d43
82cbbb6b 42ca9178
```

$v_1 v_2$

```
fc1d51 9b9be789 5f32bf1c bb27e161 e845a322   cbdbc8c7
74b79332 2f5d15d1 fdbefc11 4207e0ca f37f0e65 fbe1b4a6
6a53a29f f8a79ef7 7fa0661b b9459bd0 145fa884 dccbceeb
6174b714 ae446e92 2ae2f2f3 7662a690 ce5bb535 2ebfe5b3
bdeb2e49 07bd11b8 da86ef71 37ee95b4 decc5fb2 88d80033
aaa420d6 562acb7f
```

B.2 Paillier encryption

B.2.1 General

The parameters *p*, *q*, and *k* satisfy the requirements specified in [15].

B.2.2 1 024-bit finite field, 160-bit security parameter

B.2.2.1 Key generation

p

```
ff03b1a7 4827c746 db83d2ea ff000676 22f545b6 25843212
56e62b01 509f1096 2f9c5c8f d0b7f518 4a9ce8e8 1f439df4
7dda1456 3dd55a22 1799d2aa 57ed2713 271678a5 a0b8b40a
84ad13d5 b6e6599e 6467c670 109cf1f4 5ccfed8f 75ea3b81
4548ab29 4626fe4d 14ff764d d8b091f1 1a0943a2 dd2b983b
0df02f4c 4d00b413
```

q

```
dacaabc1 dc57faa9 fd6a4274 c4d58876 5a1d3311 c22e57d8
101431b0 7eb3ddcb 05d77d9a 742ac232 2fe6a063 bd1e05ac
b13b0fe9 1c70115c 2b1eee11 55e07252 7011a5f8 49de7072
a1ce8e6b 71db525f bcda7a89 aaed46d2 7aca5eae af35a262
70a4a833 c5cda681 ffd49baa 0f610bad 100cdf47 cc86e503
4e2a0b21 79e04ec7
```

n

```
d9f3094b 36634c05 a02ae1a5 56903510 7a48029e 39b3c6a1
853817f0 63e18e76 1c0c538e 55ff2c7e 53d603bb 35cabb3b
8d07f82a a0afdeaf 7441fcf6 746c5bca aa2cde39 8ad73edb
9c340c3f fca55913 2581eaf8 f65c13d0 2f3445a9 32a3e1fa
db5912f7 553edec5 047e4d0e d06ee87e ffc549e1 94d38e06
b73a971c 961688ba 2d4aa4f4 50d25233 72f317d4 1d06f9f0
360e962c e953a69f 36c53c37 0799fcfb a195e8f6 91ebe862
f84ae4bb d7747bc1 4499bd0e fffcdc71 54325908 355c2ffc
5b3948b8 102b33aa 24203814 70e4ee85 8380ff0e ea582885
16c263f6 d51dbbd0 e477d139 3a0a3ee6 0e1fde43 30856665
bf522006 608a6104 c138c0f3 9e09c4c5
```

λ

```
24532c37 33bb3756 455c7af0 e3c2b382 bf0c006f b448a11a
eb895952 bb504269 04acb897 b8ffdcbf b8a3ab49 de4c7489
ecd6a95c 701d4fc7 e8b5aa29 136764a1 c7077a5e ec793524
9a08acb5 54c63983 30eafc7e d3ba034d 5d3360f1 8870a5a9
cf398329 38dfcfcb 80bfb782 78127c15 2aa0e1a5 98cded01
1e89c3da 1903c174 0dea0bec 8763180b 19ab8068 b9883c80
9ed4af90 d59ada1e 22a1cfeb 89612d19 bcb057cc b781882e
bfa139e7 ff2dceb0 03961977 f0f3e828 82e99962 1697c3c3
cb5831af 064357dc d4f0c3a3 36b08b16 905fca58 dd22529f
b5315949 72aa4fa7 87c1bfa4 b258ef03 d4374cb7 36bdf6cb
ee34aa2f 4923fb4b bb85166b a3dc2052
```