INTERNATIONAL STANDARD

ISO/IEC 18032

First edition
2005-01-15

Information technology — Security techniques — Prime number generation

Technologies de l'information — Techniques de sécurité — Génération de nombres premiers

Reference number
ISO/IEC 18032:2005(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 18032 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

Information technology — Security techniques — Prime number generation

1 Scope

This International Standard specifies methods for generating and testing prime numbers as required in cryptographic protocols and algorithms.

Firstly, this International Standard specifies methods for testing whether a given number is prime. The testing methods included in this International Standard can be divided into two groups:

- Probabilistic primality tests, which have a small error probability. All probabilistic tests described here may declare a composite to be a prime. One test described here may declare a prime to be composite.
- Deterministic methods, which are guaranteed to give the right verdict. These methods use so-called primality certificates.

Secondly, this International Standard specifies methods to generate prime numbers. Again, both probabilistic and deterministic methods are presented.

NOTE Readers with a background in algorithm theory may have had previous encounters with probabilistic and deterministic algorithms. We stress that the deterministic methods in this International Standard internally still make use of random bits, and "deterministic" only refers to the fact that the output is correct with probability one.

Annex B describes variants of the methods for generating primes so that particular cryptographic requirements can be met.

The methods for generating, proving and verifying primality defined by this International Standard are applicable to cryptographic systems based on the properties of the primes.

NOTE The specifications of the tests given in this International Standard define the properties to be tested in the simplest possible form. Following these specifications directly will not necessarily produce the most efficient implementations. This is especially the case for the Frobenius-Grantham test.

2 Normative references

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

ISO/IEC 9796-2:2002, *Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms*

ISO/IEC 15946-1:2002, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 1: General*

3 Terms and definitions

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

3.1
composite number
composite
Integer $N > 1$ is composite if it is not prime, i.e. there exist divisors of N that are not trivial divisors.

3.2
entropy
Total amount of information yielded by a set of bits. It is representative of the work effort required for an adversary to be able to reproduce the same set of bits.

3.3
Jacobi symbol
Jacobi symbol of a with respect to an odd number n is the product of the Legendre symbols of a with respect to the prime factors of n (repeating the Legendre symbols for repeated prime factors).

NOTE Defined in Annex A of ISO/IEC 9796-2.

3.4
Legendre symbol
Let p be an odd prime, and let a be a positive integer. The Legendre symbol of a with respect to p is defined as $a^{(p-1)/2} \bmod p$.

NOTE Defined in Annex A of ISO/IEC 9796-2.

3.5
primality certificate
Mathematical proof that a given number is indeed a prime. For small numbers primality is most efficiently proven by trial division. In these cases, the primality certificate may therefore be empty.

3.6
prime number
prime
Integer $N > 1$ is prime if the only divisors of N are trivial divisors.

3.7
pseudo-random bit generator
Deterministic algorithm which when given some form of a bit sequence of length k outputs a sequence of bits of length $l > k$, computationally infeasible to distinguish from true random bits.

3.8
trial division
Trial division of a number N means checking all prime numbers smaller than or equal to $\sqrt{N}$ to see if they divide N.

3.9
trivial divisor
Any integer N is always divisible by 1, -1, N and $-N$. These numbers are the trivial divisors of N.

4 Symbols

a mod n – for integers a and n, (a mod n) denotes the (non-negative) remainder obtained when a is divided by n.

C – a primality certificate

$C(N)$ – primality certificate of the number N.

$C_0(N)$ – the empty primality certificate. It indicates that trial division should be used to verify that N is a prime.

gcd – greatest common divisor

$g(x) \bmod (N,f(x))$ – the remainder when the polynomial $g(x)$ is divided by the polynomial $f(x)$, calculating coefficients modulo the integer N

k – number of bits in N

L – limit below which primality is verified by trial division

$\log_b(a)$ – the logarithm of a with respect to base b

$\ln()$ – the natural logarithm (i.e., with respect to the number e)

M – a lower bound on the size of an interval in which a prime number has to be found

$\min\{a,b\}$ – the minimum of the numbers a and b

N – candidate number to be tested for primality, where N is always a positive, odd number.

n_0 – a starting point in an incremental search for a prime

n_{max} – an end point in an incremental search for a prime

T – a (probabilistic) test for primality

Z_N – the set of the integer numbers 0, 1, 2, ..., N-1, representing the ring of integers modulo N.

Z_N^* – the subset of Z_N containing the numbers that have a multiplicative inverse modulo N (i.e., if N is prime: the integer numbers 1, 2, ..., N-1)

$Z_N[x]/f(x)$ – the ring of polynomials modulo $f(x)$ with coefficients in Z_N

$\lfloor x \rfloor$ – the largest integer smaller than or equal to x.

β – a parameter which determines the lower bound of the entropy of the output of a prime generation algorithm

μ – the maximal number of steps in an incremental search for a prime

5 Trial division

The primality of an integer N can be proven by means of trial division. This is done in the following way:

1. For all primes $p \leq \sqrt{N}$
    a. If $N \bmod p = 0$ then return “reject”
2. Return “accept”

For small integers N, trial division is less computationally expensive than other primality tests. Implementations of any primality test described in this standard may define a trial division bound L, below which trial division is used in order to prove the primality of integers. This International Standard sets no value for L.

NOTE It is assumed that the set of prime numbers below a certain size are already known. One practical way to implement the test may be to have a pre-computed table of the first few primes, do trial division by these, and then simply trial divide by all odd integers up to the square root.

NOTE The size of integers for which trial division is less computationally expensive than another primality test, depends on the test and its implementation. A typical value for L might be $L = 10^3$.

6 Probabilistic primality tests

A probabilistic primality test takes a positive, odd number N as input and returns "accept" or "reject". If N is a composite number, the tests described in this clause output "reject", except with some error probability depending on the test being used. The probability that a composite number is accepted is not negligible. If N is a prime number, the Miller-Rabin test and the Frobenius-Grantham test always output "accept". The Lehmann test rejects a prime number with a probability that is not negligible.

In order to reduce the probability of errors, one usually makes several iterations with the same input (and different choices for the random values).

6.1 Miller-Rabin primality test

On input of a candidate number, N, the Miller-Rabin test starts with the following initialisation step:

1. Determine positive integers t and s such that $N\text{-}1 = 2^t s$, where s is odd.

Subsequently, the Miller-Rabin test proceeds with one or more iterations of the following algorithm, which takes inputs t, s, N and outputs "accept" or "reject". For each iteration, a different base b must be selected, with $2 \le b \le N\text{-}2$.

NOTE One iteration of the algorithm is called testing N for primality with respect to the base b.

1. Choose a random integer b such that $2 \le b \le N\text{-}2$.
2. Let $y = b^s \bmod N$

3. if $y \neq 1$ and $y \neq N\text{-}1$ then do
    a. $i = 1$
    b. while $i < t$ and $y \neq N\text{-}1$ do
        i. $y = y^2 \bmod N$
        ii. if $y = 1$ then return "reject"
        iii. $i = i + 1$
    c. if $y \neq N\text{-}1$ then return "reject"
4. return "accept"

A test for primality does not need to use exactly the same algorithm to be compliant with this International Standard. A test for primality is compliant with this International Standard if one iteration outputs "accept" if and only if one of the following requirements is satisfied for the chosen base:

- $b^s \bmod N$ equals 1
- $b^s \bmod N$ equals N-1
- For some i $(0<i<t)$: $(b^s)^{2^i} \bmod N$ equals N–1

The test only accepts the candidate if all iterations accept. Therefore the test may be stopped as soon as a base not leading to acceptance has been found.

NOTE Informative Annex A contains estimates of the error probability of this test depending on the number of iterations.

6.2 Frobenius-Grantham primality test

This test uses arithmetic in the ring $Z_N[x]/(f(x))$, where $f(x)$ is some polynomial of degree 2. The test is as follows.

On input of an odd number N, the Frobenius-Grantham primality test starts with the following initialisation steps:

1. Test N for divisibility by primes less than or equal to min{50000, $\sqrt{N}$} (trial division).
2. Test if N is a square. If yes, reject N and stop.
3. Determine positive integers r and s such that $2^r s = N^2-1$ and s is odd.

Subsequently, the test proceeds with one or more iterations of the following algorithm, which takes inputs r, s, N and outputs "accept" or "reject". In each iteration, different values for b and c must be selected.

1. Choose random $b,c \in Z_N$ until one of the following is true:
   - Either $\gcd(b^2+4c,N)$ or $\gcd(c,N)$ is a non-trivial divisor of N.
   - The Jacobi symbol of b^2+4c with respect to N is -1 mod N and the Jacobi symbol of $-c$ with respect to N is 1 mod N.

   In the first case, reject N and stop. In the second case continue with the next step.
2. Test if the polynomial $x^{(N+1)/2}$ mod (N, x^2-bx-c) has degree 0, or equivalently, is in Z_N. If not, reject N and stop.
3. Test if x^{N+1} mod $(N, x^2-bx-c) = -c$. If not, reject N and stop.
4. Test if either x^s mod $(N, x^2-bx-c) = 1$ or j $(0 \le j \le r-2)$ exists such that $x^{2^j s}$ mod $(N, x^2-bx-c) = -1$. If not, reject N and stop, otherwise accept N.

NOTE The Jacobi symbol with respect to N may be efficiently computed without knowledge of the prime factors of N [16].

NOTE A more detailed description of the test can be found in [9]. In the same document, it is proven that an iteration of the test can be implemented in such a way that its running time is approximately equal to 3 times the running time of one iteration of the Miller-Rabin test.

NOTE Information regarding the error probability of this test can be found in Annex A.

6.3 Lehmann primality test

NOTE The Lehmann test is based on the following fact: An odd integer N > 1 is prime if and only if $\forall\ a \in Z_N^*$: $a^{(N-1)/2} = \pm 1$ mod N and $\exists\ a \in Z_N^*$: $a^{(N-1)/2} = -1$ mod N.

The Lehmann primality test takes as input a candidate number N and a parameter t indicating the maximum number of iterations to be executed. The test executes the following algorithm.

1. Set f = "false".
2. Do t times

a. Choose a random integer b such that $2 \leq b \leq N-2$.

b. Let $y = b^{(N-1)/2} \bmod N$.

c. If $y \neq 1$ and $y \neq N-1$ then return “reject”.

d. If $y = N-1$, then set f = “true”.

3. If f = “true” then return “accept”, else return “reject”.

NOTE Informative Annex A contains information on the error probability of this test depending on the number of iterations.

7 Deterministic primality verification methods

Deterministic primality verification methods use primality certificates in order to verify the primality of a given number. This clause specifies the content of two types of primality certificates:

- Primality certificates based on elliptic curves
- Primality certificates for primes generated by Maurer’s algorithm (see Clause 8.3.1)

A primality certificate contains information that enables efficient verification that a given number is a prime. For both types of certificates described in this International Standard, small numbers (i.e. numbers smaller than the trial division bound L) are most efficiently verified to be primes by trial division. Let C_0 denote the empty primality certificate for such numbers.

An elliptic curve primality certificate can be computed given any prime. Hence, the method for computing this certificate can in effect also be used to verify primality. The primality certificate obtained through Maurer’s algorithm is generated as part of generating a prime, and it cannot in general be efficiently computed for an arbitrary prime (after the prime has been generated).

7.1 Elliptic curve primality certificate

Information regarding elliptic curves can be found in ISO/IEC 15946-1.

NOTE The method is based on the following fact: Let gcd(N,6) = 1 and let r be a prime such that $r > (N^{1/4}+1)^2$. If there exists a (non-singular) elliptic curve $y^2 = x^3+ax+b$ modulo N and a point $P \neq 0$ on this curve with order r, then N is prime.

To prove that a number is prime the method is used recursively. The recursion parameter is denoted by r. The total collection of data generated by the method is organised in a primality certificate. Checking the certificate, and thereby proving that N is prime, is considerably faster than generating the certificate.

NOTE If the number N is not prime, then the method will search for an elliptic curve that doesn't exist. Hence, the method will run indefinitely, unless precautions are taken. If the method is stopped before it produces a primality certificate (e.g. after a predetermined amount of time), the number N cannot be judged to be prime or composite.

7.1.1 Elliptic curve primality certificate generation

On input of an integer number N with gcd(N,6) = 1, the elliptic curve primality certificate generation starts with the following initialisations:

1. Let C = {}
2. Let j = 0
3. Let $r = N$

Subsequently the following steps are executed for decreasing values of r, until $r \le L$:

1. $j = j + 1$.
2. $t_{min} = (\sqrt[4]{r}+1)^2$.
3. Determine integers t, a, b and a point P such that the following conditions are satisfied:
   a. t is a probable prime and $t \ge t_{min}$
   b. The order of the elliptic curve E: $y^2 = x^3 + ax + b \bmod r$ is a multiple of t.
   c. P is a point on E with order t.
4. Let $C_j = (r,t,a,b,P)$.
5. Append C_j to C.
6. $r = t$.

When $r \le L$, the primality of r is proven by means of trial division. The output of the algorithm is the sequence C.

NOTE Algorithms to implement step 3.b. efficiently are described e.g. in [2].

7.1.2 Elliptic curve primality certificate verification

For a number N with $\gcd(N,6) = 1$, an elliptic curve primality certificate is a sequence $C = \{C_1, C_2, \ldots, C_k\}$, where:

$$C_i = (p_i, r_i, a_i, b_i, P_i)$$

such that:

- $p_1 = N$.
- r_k is a prime less than L.

and for all $i = 1, 2, \ldots, k$:

1. $\gcd(p_i,6) = 1$.
2. P_i is a point on the elliptic curve E: $y^2 = x^3 + a_i x + b_i \bmod p_i$,
3. P_i has order r_i.
4. $p_{i+1} = r_i$ for $1 \le i < k$.
5. $r_i > (p_i^{1/4}+1)^2$.

The number N is prime if it has such a certificate.

7.2 Primality certificate based on Maurer's algorithm

This type of primality certificate can be computed only during the generation process of the prime number, which is described in Clause 8.3.1.