INTERNATIONAL STANDARD

ISO/IEC 18032

Second edition
2020-12

Information security — Prime number generation

Sécurité de l'information — Génération de nombres premiers

Reference number
ISO/IEC 18032:2020(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: 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.

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 Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

This second edition cancels and replaces the first edition (ISO/IEC 18032:2005), which has been technically revised.

The main changes compared to the previous edition are as follows:

— the Frobenius-Grantham primality test in 6.2, the Lehmann primality test in 6.3 and Maurer's algorithm in 8.3.1, have been removed;

– the Elliptic curve primality proving algorithm, The Shawe-Taylor algorithm and the algorithm to generate primes with side conditions, have been added or substantially revised.

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.

Information security — Prime number generation

1 Scope

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

Firstly, this document specifies methods for testing whether a given number is prime. The testing methods included in this document are divided into two groups:

— probabilistic primality tests, which have a small error probability. All probabilistic tests described here can declare a composite to be a prime;

— deterministic methods, which are guaranteed to give the right verdict. These methods use so-called primality certificates.

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

NOTE It is possible that readers with a background in algorithm theory have already had previous encounters with probabilistic and deterministic algorithms. The deterministic methods in this document internally still make use of random bits (to be generated via methods described in ISO/IEC 18031), and "deterministic" only refers to the fact that the output is correct with probability one.

Annex A provides error probabilities that are utilized by the Miller-Rabin primality test.

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

Annex C defines primitives utilized by the prime generation and verification methods.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 18031, *Information technology — Security techniques — Random bit generation*

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
composite number
composite
integer for which divisors exist that are not *trivial divisors* (3.8)

3.2
deterministic random bit generator
DRBG
random bit generator that produces a random-appearing sequence of bits by applying a deterministic algorithm to a suitably random initial value called a seed and, possibly, some secondary inputs on which the security of the random bit generator does not depend

3.3
entropy
measure of the disorder, randomness or variability in a closed system

[SOURCE: ISO/IEC 18031:2011, 3.11]

3.4
Jacobi symbol
Jacobi symbol of a positive integer *a* with respect to an odd integer *n*
product of the Legendre symbols of a with respect to the prime factors of n, including *multiplicity* (3.5)

Note 1 to entry: If the prime factor p occurs with multiplicity $m \geq 1$ in the factorization of n, then the Legendre symbol of a with respect to p occurs with multiplicity m in the product that yields the Jacobi symbol of a with respect to n.

Note 2 to entry: The Legendre symbol of a positive integer a with respect to a prime number p is the value

$a^{(p-1)/2} \bmod p$

3.5
multiplicity
multiplicity of a prime divisor *p* of *n*
largest positive integer e with p^e dividing n

3.6
primality certificate
mathematical proof that a given integer is indeed a prime

Note 1 to entry: For a small integer, primality is most efficiently proven by trial division. In this case, the primality certificate can therefore be empty.

3.7
prime number
prime
positive integer for which there exist only *trivial divisors* (3.8)

3.8
trivial divisors
trivial divisors of a nonzero integer *N*
1, -1, N and $-N$

Note 1 to entry: Any nonzero integer N is divisible by (at least) 1, -1, N and $-N$.

4 Symbols and abbreviated terms

a div n	for integers a and n, with $n \neq 0$, a div n is the unique integer s satisfying $a = s \cdot n + r$ where $0 \leq r < n$.
a mod n	for integers a and n, with $n \neq 0$, a mod n is the unique non-negative integer r satisfying $a = (a \text{ div } n) \cdot n + r$.
C	primality certificate

$C(N)$	primality certificate for the number N
$C_0(N)$	empty primality certificate, indicating that trial division should be used to verify that N is a prime
exp(c)	natural exponential function evaluated at c, i.e. e^c where $e \approx 2{,}718\ 28$
gcd	greatest common divisor
Jacobi(a,N)	Jacobi symbol for an integer a with respect to a nonzero odd integer N
k	number of bits in N
L	limit below which primality is verified by trial division
Lucas(D, K, N)	K^{th} element of the Lucas sequence modulo N with discriminant D
ln(a)	natural logarithm of a with respect to the base $e \approx 2{,}718\ 28$
$\log_b(a)$	logarithm of a with respect to base b
min{a,b}	minimum of the numbers a and b
N	candidate number to be tested for primality, where N is always a positive, odd number
sgn(D)	sign of a number, i.e. sgn(D) = 1 if $D \geq 0$ and -1 otherwise.
T	(probabilistic) test for primality
Z_N	the set of the integers 0, 1, 2, ..., N-1, representing the ring of integers modulo N
Z_N^*	subset of Z_N containing the numbers that have a multiplicative inverse modulo N (e.g., if N is prime, Z_N^* consists of the integers 1, 2, ..., N-1)
β	parameter that determines the lower bound of the entropy of the output of a prime generation algorithm
μ	maximal number of steps in an incremental search for a prime
$\lfloor x \rfloor$	largest integer smaller than or equal to x
$\lceil x \rceil$	smallest integer greater than or equal to x
$\sqrt{n}$	principal (i.e. non-negative) square root of a non-negative number n

5 Trial division

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

a) For all primes $p \leq \sqrt{N}$:

   1) if N mod $p = 0$ then return "N composite" and stop;

b) return "N prime" and stop.

For small integers N, trial division is less computationally expensive than other primality tests. Implementations of any primality test described in this document may define a trial division bound

L, below which trial division is used in order to prove the primality of integers. This document sets no value for L except for a lower bound, i.e. $L > 6$.

NOTE 1 It is assumed that the set of prime numbers below a certain size are already known. One practical way to implement the test is 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 2 The size of integers for which trial division is less computationally expensive than another primality test depends on the test and its implementation. A possible value for L can be $L = 10^{10}$.

NOTE 3 A binned gcd method, as described in ANSI X9.80-2010,[1] can be more efficient than trial division for certain implementations.

6 Probabilistic primality test

6.1 General

A probabilistic primality test takes a positive, odd integer N as input and returns "N accepted" or "N composite". The Miller-Rabin primality test described in 6.3 will always output "N accepted" when N is a prime number. However, if N is a composite number, then an instance of the test can erroneously return "N accepted". In order to reduce the probability of such errors, one usually performs several iterations of testing on N, using different choices for the random values employed.

The probabilistic tests in this clause shall only be applied to odd integers that are greater or equal to the trial division bound L. If $N < L$, trial division shall be applied to determine the primality of N.

6.2 Requirements

In order for a number to be accepted as a (probable) prime, this document requires the error probability, i.e. the probability the number is composite, to be at most 2^{-100}. This provides significant confidence that any candidate prime being tested for primality that meets this threshold is indeed prime. The 2^{-100} probability bound is achieved by requiring a sufficient number of Miller-Rabin tests, depending on how the number was generated (see Annex A).

6.3 Miller-Rabin primality test

The Miller-Rabin primality test is based on the following observation. Suppose that N actually is an odd prime number and that r and s are the unique positive integers such that $N - 1 = 2^r s$, with s odd. For each positive integer $b < N$, exactly one of the following three conditions will be satisfied:

— $b^s \bmod N = 1$;

— $b^s \bmod N = N - 1$; or

— $\left(b^s\right)^{2^i} \bmod N = N - 1$, for some i with $0 < i < r$.

Equivalently, if $N \geq 3$ is an odd integer and there exists a positive integer $b < N$ that does not satisfy any of the conditions above, then N is a composite number. A probabilistic primality test based on this observation shall be applied to any odd integer $N \geq L$, as follows.

Initialization

a) Determine positive integers r and s such that $N - 1 = 2^r s$, where s is odd.

b) Set rounds = 0.

Perform t iterations (rounds) of Miller-Rabin testing (for integer $t \geq 1$).

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

d) Set $y = b^s \bmod N$.

e) If $y = 1$ or $y = N - 1$,

1) set rounds = rounds + 1;

2) if rounds <t;

go to step c)

else

return "N probably prime" and stop testing.

f) For $i = 1$ to $r - 1$, do:

1) set $y = y^2 \bmod N$;

2) if $y = N - 1$;

i) set rounds = rounds + 1;

ii) if rounds < t, go to step c);

otherwise return "N probably prime" and stop testing.

g) Return "N composite" and stop testing.

The candidate N is accepted as being (probably) prime at the conclusion of the iterated testing process if and only if N passes all t rounds of Miller-Rabin primality testing (with each choice of b satisfying one of the conditions associated with primality). The testing process returns "N composite" and is immediately halted if, for some choice of b, none of the tested conditions are satisfied (see A.2 and A.3 to determine the number of iterations required by this document).

The integer b generated in step c) shall be generated using a random bit generator that meets the specifications of ISO/IEC 18031 and converted to a number using the conversion methods in Annex C or ISO/IEC 18031. For each iteration, the process of selecting a value for b in step c) is performed anew.

NOTE 1 The rationale for the base b being randomly generated is two-fold. Firstly, the average case error estimates adopted from Reference [9] and used in A.3 assume b is random. Secondly, for a known base b, it is not difficult to construct composite integers that will pass a round of Miller-Rabin with respect to that base.

NOTE 2 Step f) is only applicable when $r > 1$, i.e. if $N \bmod 4 = 1$.

7 Deterministic primality verification methods

7.1 General

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 The Shawe-Taylor algorithm (see 8.4.2).

A primality certificate contains information that enables efficient verification that a given number is a prime. For both types of certificates described in this document, small numbers (i.e. numbers smaller than the trial division bound L) shall be 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 methods for computing this certificate may be used to verify primality. The primality certificate obtained in The

Shawe-Taylor algorithm is generated as part of the process of generating a prime, and cannot be efficiently computed for an arbitrary prime (after the prime has been generated).

7.2 Elliptic curve primality proving algorithm

7.2.1 General

Information regarding elliptic curves can be obtained from ISO/IEC 15946-1.

7.2.2 Elliptic curve primality certificate generation

To generate a certificate of primality for an odd integer $N \geq L$, the method described below is used recursively. If the method succeeds, the total collection of data generated by the method is organized in a primality certificate, $C(N)$. Verifying $C(N)$, and thereby proving that N is prime, is considerably faster than generating the certificate.

On input of an integer $N \geq L$, with $\gcd(N,6) = 1$, the elliptic curve primality proving algorithm shall start with the following initializations.

a) Set $S = \{ N \}$ (the set of integers that require a primality certificate), and set *Certs* = { } (a contentless certificate).

In the following steps, various primality tests are applied to an integer r selected from set S. If a test (provisionally) accepts the primality of r, it returns a certificate $C(r)$, and, possibly, a set of probable primes $\{ q_i \}$, in which case the certificate shall not be used to prove that r is prime, unless each q_i has been proven prime. If necessary, the algorithm shall proceed recursively, attempting to generate a certificate of primality for each of those probable primes. In the course of generating those certificates, additional probable primes may be added to the set of integers that require certificates. The algorithm shall continue until either there are no more probable primes to process or the processing is aborted because some integer requiring a certificate is determined to be a composite number.

b) Select a value for r from S and set $S = S - \{ r \}$ (i.e. remove the element r from S).

1) Apply either:

— Pocklington's primality test to r (see D.2) and, if that test is inconclusive, also apply the Deterministic Lucas primality test to r (see D.4.2); or

— the Brillhart-Lehmer-Selfridge test to r (see D.5).

In either case, when the testing is performed, allow probable primes $\geq L$ to occur (with multiplicity) in the partial factorizations of $r - 1$ and/or $r + 1$.

2) If the testing indicates r is composite, return "r composite" (along with the value of r) and stop.

3) If the testing is inconclusive, proceed to step c).

4) If the testing indicates that r is prime, then adjoin $C(r)$ to *Certs*, where $C(r)$ is the certificate for r returned by the successful test, and set $S = S \cup \{ q_i \}$, where $\{ q_i \}$ is the set of probable primes (if any) appearing in the factorizations of $r - 1$ and/or $r + 1$ (whichever were used in the testing); if S is not empty, repeat step b). Otherwise, return "N prime" along with $C(N)$ = *Certs*, and stop.

c) Generate an elliptic curve E given by $y^2 = x^3+ax+b$ for some integers a, b.

1) Apply the elliptic curve primality test to r (see D.6). When the test is performed, allow probable primes $\geq L$ to occur (with multiplicity) in the partial factorization of the order of E_r (the curve E reduced modulo r).

2) If the test indicates that r is composite, return "r composite" along with the value of r and stop.

3) If the test is inconclusive, repeat step c) with a new choice of elliptic curve.

4) If the test indicates that r is prime, then adjoin $C(r)$ to *Certs*, where $C(r)$ is the certificate for r returned by the successful test, and set $S = S \cup \{ q_i \}$, where $\{ q_i \}$ is the set of probable primes (if any) appearing in the factorization of the order of E_r; if S is not empty, go to step b). Otherwise, return "N is prime" along with $C(N) = Certs$, and stop.

The probable prime factors (if any) appearing in steps b) and c) are required to be greater than or equal to the trial division bound. The primality of smaller factors shall be ascertained using trial division.

During the recursion, the tests in step b) can be skipped. It is included in the algorithm's description for efficiency purposes.

If the primality of N is inconclusive, the test may be executed again. The test may not be rerun in full if partial results of the previous execution are retained (e.g., the value of *Certs* when testing process was stopped and the composite r value that caused the early termination). It can be sufficient to back the recursion step one level prior to the point of failure. Varying choices of curves in step c) and/or optionally skipping step b) is likely to result in different sets of probable primes to test.

In case r is composite, the elliptic curve test in step c) is likely to be inconclusive and so the algorithm can fail to terminate. As a precaution, a limit on the number of iterations of step c) may be enforced.

NOTE The elliptic curve E in step c) can be generated using the CM method described in D.8.

7.2.3 Elliptic curve primality certificate verification

An elliptic curve primality certificate for an integer $N \geq L$, with $\gcd(N, 6) = 1$, is a set of (interdependent) certificates, $C(N) = \{ C_i \}$, where each C_i asserts the primality of some integer $r_i \geq L$, and exactly one of the r_i is equal to N. Each C_i is a certificate resulting from the execution of either Pocklington's test (D.2.2), the Deterministic Lucas test (D.4.2), the Brillhart-Selfridge-Lehmer test (D.5) or the elliptic curve test (D.6).

If C_i asserts the primality of r_i based (in part) on the assumed primality of one or more other integers $q_j \geq L$, then the certificate C_i (and hence the primality of r_i) shall only be successfully verified after the primality of each of the q_j is accepted via the verification of their certificates, which shall also be included in $C(N)$. If C_i asserts the primality of r_i based (in part) on the assumed primality of any integers less than L, then trial division shall be used to verify the primality of those integers as part of the verification of C_i.

The elliptic curve primality certificate $C(N) = \{ C_i \}$ is accepted only if every C_i is accepted (and exactly one of the r_i is equal to N). If the verification of any C_i fails, then the elliptic curve primality certificate for N shall be rejected.

Verifying each of the individual certificates C_i shall be done as specified in Annex D.

7.3 Primality certificate based on The Shawe-Taylor algorithm

This type of primality certificate C is a collection of certificates $\{ C_i \}$ which shall be computed (only) during the generation process of the prime number using the process described in 8.4.2 (The Shawe-Taylor algorithm). The certificates C_i are the result of the Pocklington test (see D.2.2) whose proofs of primality have the following structure:

$$\text{Proof}(r_i) = (r_i, q_i, a_i)$$

where r_i, q_i, and a_i are integers. The certificate C_i, which asserts that r_i is prime, is verified once it passes the checks in C.2 and q_i is proven to be (an odd) prime (e.g., by trial division or by verifying its own certificate, which has also been included in C). The last certificate to be verified, say $C_1 = (N, q_1, a_1)$, contains the value N. If all of the C_i are verified (and are confirmed to chain up to N), then C itself is verified and N is accepted as being prime.

8 Prime number generation

8.1 General

This clause specifies two approaches to prime number generation. The first approach is to employ the probabilistic Miller-Rabin primality test on randomly chosen candidates (8.3). [Optionally, a probable prime, N, generated by such methods may subsequently be proven prime by generating an elliptic curve primality certificate (7.2) for N.] The second approach to prime number generation is to employ the deterministic Shawe-Taylor method (8.4) which yield integers known to be prime. This method can also produce primality certificates as part of the generation process.

The methods in this clause generate primes in the interval $(2^{k-1}, 2^k)$ for some k. To generate primes with additional constraints, such as generating primes in a more restrictive interval and/or primes that satisfy certain congruence conditions, the methods in B.2 shall be used. Examples of primes generated with additional constraints are given in Annex E.

These techniques shall only be applied to generate primes greater or equal to the trial division bound L. Generating primes less than L can be simply done by selecting integers less than L and testing for primality using trial division.

8.2 Requirements

In order for a prime number generation algorithm to conform to this document, the algorithm shall:

— generate random bits using a deterministic random bit generator that meets the specifications of ISO/IEC 18031;

— generate random numbers from sequences of random bits using the conversion methods specified in Annex C or ISO/IEC 18031;

— ensure, in the case of non-provable primes (8.3), that the error probability that a composite passes as prime is at most 2^{-100}.

Annex A contains more information on how the algorithms in 8.3 shall satisfy the 2^{-100} error properties.

For applications where the primes generated need to be secret, the following also applies:

— the (secret) entropy used in the random bit generator to produce the output number shall satisfy the requirements of ISO/IEC 18031 and shall be at least B bits where B is the security level of that application;

— if additional constraints are placed on the prime being generated (B.2), the constraints shall be limited so as not to affect the security of the system and the requirements in Annex B shall apply.

In the case of generating primes for use in RSA, B should be at least 112 for 1 024-bit primes, 128 for 1 536-bit primes, 192 for 3 840-bit primes, and 256 for 7 680-bit primes. ISO/IEC 18031 requires a minimum entropy of 120 bits to avoid collisions. In the case of RSA, collisions can lead to primes repeating for different RSA moduli. Such primes can be recovered by computing a gcd among the RSA moduli.

NOTE Limiting the total number of constraints is necessary to avoid Coppersmith-style attacks,[7][14] e.g. these attacks can factor RSA moduli if roughly half the bits of a prime factor are known. If the constraints (such as the number of known bits) are limited, then these attacks do not apply. For example, requiring an integer $n < 2^k$ to lie in the interval $(2^{k-2} + 2^{k-1}, 2^k)$ with $n \bmod 4 = 3$, and $\gcd(n-1, e) = 1$ for some odd encryption exponent gives less than 5 bits of constraint. If the interval is replaced with $(2^{k-1}\sqrt{2}, 2^k)$, the total constraint is less than 4,5 bits. Generating a k–bit number n with $n \bmod q = 1$ for some randomly generated (and secret) integer q of size m bits places roughly two bits of constraint on n (since n can be expressed as $n = 1 + u\,q$ where u, q are both k-m, m-bit numbers respectively). Further requiring q to be prime adds a few additional bits of constraint, approximately $\log_2(m) - 0{,}528$ bits by the prime number theorem.

When regenerating (secret) primes from a fixed seed, care should be taken so as to avoid side-channel attacks.

