INTERNATIONAL STANDARD

ISO/IEC 15946-5

Third edition
2022-02

Information security — Cryptographic techniques based on elliptic curves —

Part 5: Elliptic curve generation

Sécurité de l'information — Techniques cryptographiques fondées sur les courbes elliptiques —

Partie 5: Génération de courbes elliptiques

Reference number
ISO/IEC 15946-5:2022(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 or www.iec.ch/members_experts/refdocs).

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 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. In the IEC, see www.iec.ch/understanding-standards.

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

This third edition cancels and replaces the second edition (ISO/IEC 15946-5:2017), which has been technically revised.

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

— BLS curves have been added to Clause 7;

— security background for pairing-friendly curves has been added to Annex B, including the exTNFS attack that affects the security of numerical examples of BN curves;

— except for BN curves, all other curves have been moved to Annex C;

— associated numerical examples (Annex D) and properties (Annex E) have been updated.

A list of all parts in the ISO/IEC 15946 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 and www.iec.ch/national-committees.

Introduction

Some of the most interesting alternatives to the RSA and $F(p)$ based systems are cryptosystems based on elliptic curves defined over finite fields. The concept of an elliptic curve based public-key cryptosystem is rather simple.

— Every elliptic curve over a finite field is endowed with an addition operation "+", under which it forms a finite abelian group.

— The group law on elliptic curves extends in a natural way to a "discrete exponentiation" on the point group of the elliptic curve.

— Based on the discrete exponentiation on an elliptic curve, one can easily derive elliptic curve analogues of the well-known public-key schemes of Diffie-Hellman and ElGamal type.

The security of such a public-key system depends on the difficulty of determining discrete logarithms in the group of points of an elliptic curve. With current knowledge, this problem is much harder than the factorization of integers or the computation of discrete logarithms in a finite field. Indeed, since Miller and Koblitz independently suggested the use of elliptic curves for public-key cryptographic systems in 1985, the elliptic curve discrete logarithm problem has only been shown to be solvable in certain specific and easily recognizable cases. There has been no substantial progress in finding an efficient method for solving the elliptic curve discrete logarithm problem on arbitrary elliptic curves. Thus, it is possible for elliptic curve based public-key systems to use much shorter parameters than the RSA system or the classical discrete logarithm-based systems that make use of the multiplicative group of a finite field. This yields significantly shorter digital signatures and system parameters.

The purpose of this document is to meet the increasing interest in elliptic curve based public-key technology by describing elliptic curve generation methods to support key management, encryption and digital signatures based on an elliptic curve.

Information security — Cryptographic techniques based on elliptic curves —

Part 5: Elliptic curve generation

1 Scope

The ISO/IEC 15946 series specifies public-key cryptographic techniques based on elliptic curves described in ISO/IEC 15946-1.

This document defines elliptic curve generation techniques useful for implementing the elliptic curve based mechanisms defined in ISO/IEC 29192-4, ISO/IEC 9796-3, ISO/IEC 11770-3, ISO/IEC 14888-3, ISO/IEC 18033-2 and ISO/IEC 18033-5.

This document is applicable to cryptographic techniques based on elliptic curves defined over finite fields of prime power order (including the special cases of prime order and characteristic two). This document is not applicable to the representation of elements of the underlying finite field (i.e. which basis is used).

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

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15946-1 and the following apply.

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

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

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

3.1
cryptographic hash function
function that maps octet strings of any length to octet strings of fixed length, such that it is computationally infeasible to find correlations between inputs and outputs, and such that given one part of the output, but not the input, it is computationally infeasible to predict any bit of the remaining output

[SOURCE: ISO/IEC 18033-2:2006, 3.11, modified — Deleted the last phrase, "The precise security requirements depend on the application.]

3.2
definition field of an elliptic curve
field that includes all the coefficients of the formula describing an elliptic curve

3.3
hash-function
function which maps strings of bits of variable (but usually upper bounded) length to fixed-length strings of bits, satisfying the following two properties:

— for a given output, it is computationally infeasible to find an input which maps to this output;

— for a given input, it is computationally infeasible to find a second input which maps to the same output.

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment. Refer to ISO/IEC 10118-1:2016, Annex C.

[SOURCE: ISO/IEC 10118-1:2016, 3.4]

3.4
nearly prime number
positive integer, $n = m \cdot r$, where m is a large prime number and r is a small *smooth integer* (3.6)

Note 1 to entry: The meaning of the terms large and small prime numbers is dependent on the application and is based on bounds determined by the designer.

3.5
order of an elliptic curve *E*(*F*)
number of points on an elliptic curve, E, defined over a finite field, F

3.6
smooth integer
integer, r, whose prime factors are all small, i.e. less than some defined bound

4 Symbols and conversion functions

4.1 Symbols

0_F	zero element in a field F
a, b	elliptic curve parameters
B	embedding degree, the smallest B such that number $\#E(F(q))$ is a divisor of q^B-1
E	elliptic curve given by an equation of the form $Y^2 = X^3 + aX + b$ over the field $F(p^m)$ for $p > 3$, by an equation of the form $Y^2 + XY = X^3 + aX^2 + b$ over the field $F(2^m)$, or by an equation of the form $Y^2 = X^3 + aX^2 + b$ over the field $F(3^m)$, together with an extra point O_E referred to as the point at infinity; the curve is denoted by $E/F(p^m)$, $E/F(2^m)$, or $E/F(3^m)$, respectively
F	finite field
$F(p)^*$	multiplicative group of $F(p)$
L, m	positive integers
mod	modulo, a mod b means a residue of a modulo b for integers a and b (b is positive)
N	number of points on an elliptic curve E over $F(q)$, $\#E(F(q))$
n	prime divisor of $\#E(F(q))$
O_E	elliptic curve point at infinity
p	prime number

q	prime power, p^m for some prime p and some integer $m \geq 1$
r	cofactor, that is $\#E(F(q)) = rn$
u	positive integer
v	positive integer
λ	positive integer
$v(i)$	polynomial of i
$\#E(F(q))$	order of an elliptic curve $E(F(q))$
$\lceil x \rceil$	smallest integer greater than or equal to the real number x
$\lfloor x \rfloor$	largest integer smaller than or equal to the real number x
$\cup$	or
$\|\|$	concatenation
$\in$	element is included in a set

4.2 Conversion functions

BS2IP	bit string to integer conversion primitive
BS2OSP	bit string to octet string conversion primitive
I2BSP	integer to bit string conversion primitive
OS2FEP	octet string to finite field element conversion primitive

NOTE Refer to ISO/IEC 15946-1:2016, Clause 7 for detailed conversion functions.

5 Conventions for elliptic curves

5.1 Definitions of elliptic curves

5.1.1 Elliptic curves over $F(p^m)$

Let $F(p^m)$ be a definition field of an elliptic curve, where $F(p^m)$ is a finite field with a prime $p > 3$ and a positive integer m. In this document, it is assumed that E is described by a "short (affine) Weierstrass equation", that is an equation of type

$$Y^2 = X^3 + aX + b \text{ with } a, b \in F(p^m)$$

such that $4a^3 + 27b^2 \neq 0_F$ holds in $F(p^m)$.

NOTE 1 The above curve with $4a^3 + 27b^2 = 0_F$ is called a singular curve, which is not an elliptic curve.

The set of $F(p^m)$-valued points of E is given by Formula (1):

$$E(F(p^m)) = \{Q = (x_Q, y_Q) \in F(p^m) \times F(p^m) | {y_Q}^2 = {x_Q}^3 + ax_Q + b\} \cup \{O_E\} \quad (1)$$

where O_E is an extra point referred to as the point at infinity of E.

NOTE 2 In applications not based on a pairing, $E/F(p)$ or $E/F(2^m)$ is preferable from an efficiency point of view. In applications that use a pairing, $E/F(p)$ is preferable from an efficiency point of view.

5.1.2 Elliptic curves over $F(2^m)$

Let $F(2^m)$ for $m \geq 1$ be a definition field of an elliptic curve. In this document, it is assumed that E is described by an equation of the type

$$Y^2 + XY = X^3 + aX^2 + b \text{ with } a, b \in F(2^m)$$

such that $b \neq 0_F$ holds in $F(2^m)$.

For cryptographic use, m shall be a prime to prevent certain kinds of attacks on the cryptosystem.

NOTE The above curve with $b = 0_F$ is called a singular curve, which is not an elliptic curve.

The set of $F(2^m)$-valued points of E is given by Formula (2):

$$E(F(2^m)) = \{Q = (x_Q, y_Q) \in F(2^m) \times F(2^m) | {y_Q}^2 + x_Q y_Q = {x_Q}^3 + a{x_Q}^2 + b\} \cup \{O_E\} \quad (2)$$

where O_E is an extra point referred to as the point at infinity of E.

5.1.3 Elliptic curves over $F(3^m)$

Let $F(3^m)$ be a definition field of an elliptic curve, where $F(3^m)$ is a finite field with a positive integer m. In this document, it is assumed that E is described by an equation of the type

$$Y^2 = X^3 + aX^2 + b \text{ with } a, b \in F(3^m)$$

such that $a, b \neq 0_F$ holds in $F(3^m)$.

NOTE The above curve with a or $b = 0_F$ is called a singular curve, which is not an elliptic curve.

The set of $F(3^m)$-valued points of E is given by Formula (3):

$$E(F(3^m)) = \{Q = (x_Q, y_Q) \in F(3^m) \times F(3^m) | {y_Q}^2 = {x_Q}^3 + a{x_Q}^2 + b\} \cup \{O_E\} \quad (3)$$

where O_E is an extra point referred to as the point at infinity of E.

5.2 Group law on elliptic curves

Elliptic curves are endowed with the addition operation +: $E \times E \rightarrow E$, defining for each pair (Q_1, Q_2) of points on E a third point $Q_1 + Q_2$. With respect to this addition, E is an abelian group with identity element O_E. The kth multiple of Q is given as kQ, where $kQ = Q + ... + Q$ (k summands) if $k > 0$, $kQ = (-k)(-Q)$ if $k < 0$, and $kQ = O_E$ if $k = 0$. The smallest positive k with $kQ = O_E$ is called the order of Q.

In order to use an elliptic curve for a cryptosystem, it is necessary to compute group law on an elliptic curve. ISO/IEC 15946-1 shall be referred to for methods of group law on elliptic curves.

6 Framework for elliptic curve generation

6.1 Trust in elliptic curve

If an elliptic curve is poorly or maliciously generated, this can enable an adversary to break a cryptosystem that uses it. There are a number of ways in which a user can obtain trust in the provenance of an elliptic curve, including the following:

— The curve can be obtained from an impartial trusted source (e.g. an international or national standard).

— The curve can be generated and/or verified by a trusted third party.

— The curve can be generated and/or verified by the user.

NOTE 1 Refer to Annex A for background information on elliptic curves.

NOTE 2 Refer to Annex B for background information on elliptic curve cryptography.

6.2 Overview of elliptic curve generation

There are three main ways to generate elliptic curves:

— by applying the order of counting algorithms to a (pseudo-)randomly chosen elliptic curve as specified in Clause 7;

— by applying a complex multiplication method as specified in Clause 8;

— by lifting an elliptic curve over a small finite field over a reasonably large field as specified in Clause 9.

NOTE 1 Refer to Annex A for background information on elliptic curves.

NOTE 2 Refer to Annex B for background information on elliptic curve cryptography.

7 Verifiably pseudo-random elliptic curve generation

7.1 General

The generation of verifiably pseudo-random elliptic curves focuses on curves over prime and binary fields.

7.2 Constructing verifiably pseudo-random elliptic curves (prime case)

7.2.1 Construction algorithm

The following algorithm produces a set of elliptic curve parameters over a field $F(p)$ selected (pseudo-) randomly from the curves of appropriate order, along with sufficient information for others to verify that the curve was indeed chosen pseudo-randomly.

NOTE 1 The algorithm is consistent with IEEE P1363-2000.

NOTE 2 Methods of choosing a prime number p (pseudo) randomly are described in ISO/IEC 18032.

It is assumed that the following quantities have been chosen:

— a lower bound, $n_{\min}$, for the order of the base point;

— a cryptographic hash function, H, with output length L_{Hash} bits;

— the bit length, L, of inputs to H, satisfying $L \geq L_{\text{Hash}}$.

The following notation is adopted below:

— $v = \lceil \log_2 p \rceil$;

— $s = \lfloor (v - 1)/L_{\text{Hash}} \rfloor$;

— $w = v - sL_{\text{Hash}} - 1$.

Input: a prime number p; lower bound $n_{\min}$ for n; a trial division bound $l_{\max}$.

Output: a bit string X; EC parameters a, b, n, and G.

a) Choose an arbitrary bit string X of bit length L.

b) Compute $h = \text{H}(X)$.

c) Let W_0 be the bit string obtained by taking the w rightmost bits of h.

d) Let $Z = \text{BS2IP}(X)$.

e) For i from 1 to s, do the following:

   1) Let $X_i = \text{I2BSP}(Z + i \bmod 2^L)$.

   2) Compute $W_i = \text{H}(X_i)$.

f) Let $W = W_0 \,||\, W_1 \,||\, \ldots \,||\, W_s$.

g) Let $c = \text{OS2FEP}[\text{BS2OSP}(W)]$.

h) If $c = 0_F$ or $4c + 27 = 0_F$, then go to step a).

i) Choose finite field elements a, $b \in F(p)$ such that $b \neq 0_F$ and $cb^2 - a^3 = 0_F$. Choosing $a = b = c$ guarantees the conditions hold, and this choice is recommended.

   NOTE 3 It is possible that choosing $a = b = c$ is not optimal from a performance perspective.

   NOTE 4 If the default values are chosen as suggested, the randomness of the generated curve is explicitly guaranteed.

j) Compute the order $\#E(F(p))$ of the elliptic curve E over $F(p)$ given by $y^2 = x^3 + ax + b$.

k) Test whether $\#E(F(p))$ is a nearly prime number using the algorithm specified in 7.2.2. If so, the output of the algorithm specified in 7.2.2 consists of integers r, n. If not, then go to step a).

   NOTE 5 The necessity of near primality is described in B.2.2

l) Check if $E(F(p))$ satisfies the MOV-condition specified in B.2.3, that is the smallest integer B such that n divides $q^B - 1$ ensures the desirable security level. If not, then go to step a).

m) If $\#E(F(p)) = p$, then go to step a).

   NOTE 6 This check is performed in order to protect against the attack specified in B.2.2.

n) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5. If not, then go to step a).

o) Generate a point G on E of order n using the algorithm specified in 7.2.3.

p) Output X, a, b, n, G.

NOTE 7 Methods to compute the order $\#E(F(p))$ are described in References [12], [31] and [32].

7.2.2 Test for near primality

Given a lower bound n_{min} and a trial division bound l_{max}, the following procedures test $N = \#E(F(p))$ for near primality.

Input: positive integers N, l_{max}, and n_{min}.

Output: if N is nearly prime, output a prime n with $n_{min} \leq n$ and a smooth integer r such that $N = rn$. If N is not nearly prime, output the message "not nearly prime".

a) Set $n = N$, $r = 1$.

b) For l from 2 to l_{max}, do the following:

   1) If l is composite, then go to step 3).

   2) While (l divides n)

      i) Set $n = n/l$ and $r = rl$.

      ii) If $n < n_{min}$, then output "not nearly prime" and stop.

   3) Next l.

c) Test n for primality.

d) If n is prime, then output r and n and stop.

e) Output "not nearly prime".

NOTE Methods to test for primality are described in ISO/IEC 18032 and Reference [11].

7.2.3 Finding a point of large prime order

If the order $\#E(F(q))$ of an elliptic curve E is nearly prime, the following algorithm efficiently produces a random point in $E(F(q))$ whose order is the large prime factor n of $\#E(F(q)) = rn$.

Input: an elliptic curve E over the field $F(q)$, a prime n, and a positive integer r not divisible by n.

Output: if $\#E(F(q)) = rn$, a point G on E of order n; if not, the message "wrong order."

a) Generate a random point P (not O_E) on E.

b) Set $G = rP$.

c) If $G = O_E$, then go to step a).

d) Set $Q = nG$.

e) If $Q \neq O_E$, then output "wrong order" and stop.

f) Output G.

7.2.4 Verification of elliptic curve pseudo-randomness

The following algorithm determines whether or not an elliptic curve over $F(p)$ was generated using the method of 7.2.1. The quantities L_{Hash}, L, v, s, and w, and the hash-function H, are as in 7.2.1.

Input: a bit string X of length L, EC parameters $q = p$, a, b, n, and $G = (x_G, y_G)$, and a positive integer n_{min}.

Output: "True" or "False".

a) Compute $h = H(X)$.

b) Let W_0 be the bit string obtained by taking the w rightmost bits of h.

c) Let Z = BS2IP(X).

d) For i from 1 to s, do the following:

1) Let X_i = I2BSP($Z + i \bmod 2^L$).

2) Compute $W_i = H(X_i)$.

e) Let $W = W_0 \,||\, W_1 \,||\, \ldots \,||\, W_s$.

f) Convert W to a finite field element c = OS2FEP [BS2OSP (W)].

g) Verify the following conditions.

1) $n \geq n_{\min}$.

2) n is a prime.

3) $c \neq 0_F$.

4) $4c + 27 \neq 0_F$.

5) $b \neq 0_F$.

6) $cb^2 - a^3 = 0_F$.

7) $G \neq O_E$.

8) $y^2{}_G = x^3{}_G + ax_G + b$.

9) $nG = O_E$.

h) If all the conditions in step g) hold, then output "True"; otherwise output "False".

7.3 Constructing verifiably pseudo-random elliptic curves (binary case)

7.3.1 Construction algorithm

The following algorithm produces a set of elliptic curve parameters for a pseudo-random curve over a field $F(2^m)$, along with sufficient information for others to verify that the curve was indeed chosen pseudo-randomly. See Annex D for additional information.

NOTE 1 The algorithm is consistent with IEEE P1363-2000.

It is assumed that the following quantities have been chosen:

— a field $F(2^m)$;

— a lower bound $n_{\min}$ for the order of the base point;

— a cryptographic hash function H with output length L_{Hash} bits;

— the bit length L of inputs to H, satisfying $L \geq L_{\text{Hash}}$.

The following notation is adopted below:

— $s = \lfloor (m - 1)/ L_{\text{Hash}} \rfloor$;

— $w = m - sL_{\text{Hash}}$.

Input: a field $F(2^m)$; a lower bound $n_{\min}$ for n; a trial division bound $l_{\max}$.

Output: a bit string X; EC parameters a, b, n, and G.

a) Choose an arbitrary bit string X of bit length L.

b) Compute $h = \mathrm{H}(X)$.

c) Let W_0 be the bit string obtained by taking the w rightmost bits of h.

d) Let $\mathrm{Z} = \mathrm{BS2IP}(x)$.

e) For i from 1 to s, do the following:

   1) Let $X_i = \mathrm{I2BSP}(Z + i \bmod 2^L)$.

   2) Compute $W_i = \mathrm{H}(X_i)$.

f) Let $W = W_0 \,||\, W_1 \,||\, \ldots \,||\, W_s$.

g) Let $b = \mathrm{OS2FEP}\,[\mathrm{BS2OSP}(W)]$.

h) If $b = 0_F$, then go to step a).

i) Let a be an arbitrary element in $F(2^m)$. Choosing $a = 0_F$ guarantees the conditions hold, and this choice is recommended.

   NOTE 2 It is possible that the default values are not chosen because of performance reasons.

   NOTE 3 If the default values are chosen as suggested, the randomness is explicitly guaranteed.

j) Compute the order $\#E(F(2^m))$ of the elliptic curve E over $F(2^m)$ given by $y^2 + xy = x^3 + ax^2 + b$.

   NOTE 4 Methods of computing the order $\#E(F(2^m))$ are described in References [12], [31] and [34].

k) Test whether $\#E(F(2^m))$ is a nearly prime number using the algorithm specified in 7.2.2. If so, the output of the algorithm specified in 7.2.2 consists of integers r, n. If not, then go to step a).

l) Check that $E(F(2^m))$ satisfies the MOV-condition specified in B.2.3. If not, then go to step a).

m) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, or ECDHP with auxiliary inputs as in B.1.5. If not, then go to step a).

n) Generate a point G on E of order n using the algorithm specified in 7.2.3.

o) Output X, a, b, n, G.

NOTE 5 The necessity of near primality is described in B.2.2.

7.3.2 Verification of elliptic curve pseudo-randomness

The following algorithm verifies the validity of a set of elliptic curve parameters. In addition, it determines whether an elliptic curve over $F(2^m)$ was generated using the method of 7.3.1.

The quantities L_{Hash}, L, s, and w, and the hash-function H, are as in 7.3.1.

Input: a bit string X of length L, EC parameters $q = 2^m$, a, b, n, and $G = (x_G, y_G)$, and a positive integer $n_{\min}$.

Output: “True” or “False”.

a) Compute $h = \mathrm{H}(X)$.

b) Let W_0 be the bit string obtained by taking the w rightmost bits of h.

c) Let $\mathrm{Z} = \mathrm{BS2IP}(x)$.

d) For i from 1 to s, do the following:

1) Let X_i = I2BSP($Z + i \bmod 2^L$).

2) Compute $W_i = H(X_i)$.

e) Let $W = W_0 \,||\, W_1 \,||\, \ldots \,||\, W_s$.

f) Let b' = OS2FEP [BS2OSP(W)].

g) Verify the following conditions:

1) $n \geq n_{\min}$

2) n is a prime

3) $b \neq 0_F$

4) $b = b'$

5) $G \neq O_E$

6) $y^2{}_G + x_G y_G = x^3{}_G + ax^2{}_G + b$

7) $nG = O_E$

h) If all the conditions in step g) hold, then output "True"; otherwise output "False".

8 Constructing elliptic curves by complex multiplication

8.1 General

This document defines how to construct two types of elliptic curves by complex multiplication: Barreto-Naehrig (BN) curves and Barreto-Lynn-Scott (BLS) curves. Methods for constructing more curves are presented in Annex C. Numerical examples and curve properties are given in Annex D and Annex E, respectively.

8.2 Barreto-Naehrig (BN) curve

The following algorithm produces an elliptic curve E over $F(p)$ with embedding degree $B = 12$, which is useful for cryptosystems based on bilinear pairings. The embedding degree is described in B.2.2.

NOTE 1 Detailed information is given in Reference [13].

NOTE 2 This method always generates at most one curve for a given value of m.

Input: the approximate desired size m of the curve order (in bits) and upper bound (odd integer) $p_{\max}$ for the definition field.

Output: prime p, curve parameters of elliptic curve $E/F(p)$, prime divisor n and basepoint G.

a) Let $P(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$.

b) Compute the smallest $u \approx 2^{m/4}$ such that $\lceil \log_2 P(-u) \rceil = m$.

c) While $p \leq p_{\max}$:

1) $t = 6u^2 + 1$;

2) $p = P(-u)$ and $N = p + 1 - t$;

3) if p and N are prime, then go to step e);

4) $p = P(u)$ and $N = p + 1 - t$;

5) if p and N are prime, then go to step e);

6) $u = u + 1$ and go to step 1).

d) Stop and output "fail".

e) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5. If not, then go to step a).

f) Set an elliptic curve $E: y^2 = x^3 + b$ with $b = 1$.

1) Test whether E has order n. If not, then $b = b + 1 \pmod{p}$.

g) $x_0 = 1$:

1) if ${x_0}^3 + b$ does not have square roots, then $x_0 = x_0 + 1$ and go to 1);

2) compute $y_0 = \sqrt{({x_0}^3 + b)} \bmod p$:

i) Set the basepoint $G = (x_0, y_0) \in E$;

ii) If $nG \neq O_E$, then $x_0 = x_0 + 1$ and go to 1).

h) Output p, E, n, and G.

NOTE 3 A technique for speeding up a protocol based on a bilinear pairing is described in Reference [13].

8.3 Barreto-Lynn-Scott (BLS) curve

The following algorithm produces an elliptic curve E over $F(p)$ with embedding degree $B = 12$, 24 or 48 which is useful for cryptosystems based on a bilinear pairing. The embedding degree is described in B.2.2.

NOTE 1 Detailed information is given in[35] and[36].

NOTE 2 This method always generates at most one curve for a given value of m.

Input: the approximate desired size m of the curve order (in bits) and upper bound (odd integer) p_{max} for the definition field.

Output: prime p, curve parameter of elliptic curve $E/F(p)$, prime divisor n and base point G.

a) Set $B = 12$, 24 or 48.

1) If $B = 12$, then $L(u) = (u^4 - u^2 + 1)$ and the smallest $u = 2^{m/4}$ for $\lceil \log_2 L(-u) \rceil = m$.

2) If $B = 24$, then $L(u) = (u^8 - u^4 + 1)$ and the smallest $u = 2^{m/8}$ for $\lceil \log_2 L(-u) \rceil = m$.

3) If $B = 48$, then $L(u) = (u^{16} - u^8 + 1)$ and the smallest $u = 2^{m/16}$ for $\lceil \log_2 L(-u) \rceil = m$.

b) Set $P(u) = (u - 1)^2 L(u)/3 + u$ and $R(u) = (u-1)^2/3$.

c) While $p \leq p_{max}$:

1) $t = u + 1$;

2) $p = P(-u)$, $n = L(-u)$ and $r = R(-u)$;

3) if p and n are primes then go to step e);

4) $p = P(u)$, $n = L(u)$, and $r = R(u)$;

5) if p and n are primes, then go to step e);

6) $u = u + 1$ and go to step 1).

d) Stop and output "fail".

e) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5. If not, then go to step a).

f) Set an elliptic curve $E: y^2 = x^3 + b$ with $b = 1$.

1) Test whether E has order n. If not, then $b = b + 1 \pmod{p}$.

g) $x_0 = 1$.

h) While $x_0 \leq p$:

1) if ${x_0}^3 + b$ does not have square roots, then $x_0 = x_0 + 1$ and go to 1);

2) compute $y_0 = \sqrt{({x_0}^3 + b)} \bmod p$.

i) Put $G_0 = (x_0, y_0) \in E$.

ii) If $rG = O_E$ then $x_0 = x_0 + 1$ and go to 1).

iii) Set $G = rG_0$.

iv) If $nG \neq O_E$, then $x_0 = x_0 + 1$ and go to 1).

i) Stop and output "fail".

j) Output p, E, n and G.

9 Constructing elliptic curves by lifting

The following algorithm produces an elliptic curve E over $F(p^m)$ by lifting an elliptic curve E over $F(p)$.

NOTE 1 The algorithm is based on Reference [33].

Input: small finite field $F(p)$, elliptic curve E over $F(p)$, lower and upper bound $N_{\min}$ and $N_{\max}$ for the order of elliptic curve (in bits).

Output: extension degree m, order $N_m = \#E(F(p^m))$, basepoint G, and order n of G.

a) Count the order of $N = \#E(F(p))$, which is easily executed since $F(p)$ is small.

b) Set $t = p + 1 - N$ and compute algebraic integers α and β that satisfy $t = \alpha + \beta$ and $p = \alpha\beta$.

c) Set $m = 1$.

Find a triple of (m, N_m, n) as follows:

1) Compute $N_m = pm + 1 - (\alpha^m + \beta^m)$ and $q = p^m$, which is an integer.

2) If $N_m < N_{\min}$, then $m = m + 1$ and go to Step 1).

3) If $N_m > N_{\max}$, then stop and output "fail".

4) Test whether N_m is a nearly prime number using the algorithm specified in 7.2.2. If so, the output of 7.2.2 consists of the integers r and n. If not, then $m = m + 1$ and go to step 1).

5) Check whether $E(F(q))$ satisfies the MOV-condition, that is the smallest integer B such that n divides $q^B - 1$ ensures the desirable security level. If not, then $m = m + 1$ and go to step 1).

NOTE 2 See the MOV-condition in B.2.3.

e) Generate a point G on $E(F(q))$ of order n using the algorithm specified in 7.2.3.

f) Output an extension degree m, the order $N_m = \#E(F(q))$, a basepoint G and the order n.

Annex A
(informative)

Background information on elliptic curves

A.1 *j*-invariant

Let $F(q)$ be a finite field with $q = p^m$, where prime $p > 3$. Let E be an elliptic curve over $F(q)$ given by the short Weierstrass equation:

$Y^2 = X^3 + aX + b$ with $a, b \in F(q)$,

where the inequality $4a^3 + 27b^2 \neq 0_F$ holds in $F(q)$. Then, the j-invariant is defined as:

$j = 1\,728 \cdot (4a^3)/(4a^3 + 27b^2)$

Let $F(2^m)$, for some $m \geq 1$, be a finite field. Let E be an elliptic curve over $F(2^m)$ given by:

$Y^2 + XY = X^3 + a\,X^2 + b$ with $a, b \in F(2^m)$,

where $b \neq 0_F$. Then, the j-invariant is defined as:

$j = 1/b$

Let $F(3^m)$, for some $m \geq 1$, be a finite field. Let E be an elliptic curve over $F(3^m)$ given by:

$Y^2 = X^3 + aX^2 + b$ with $a, b \in F(3^m)$

such that $a, b \neq 0_F$. Then, the j-invariant is defined as:

$j = -a^3/b$

A.2 Hilbert class polynomial

The construction of elliptic curves by complex multiplication uses the theory of imaginary quadratic fields $Q(\sqrt{-D})$. In the case of the imaginary quadratic field $Q(\sqrt{-D})$, the Hilbert class field K is the extension field of $Q(\sqrt{-D})$, which is the unramified abelian extension of $Q(\sqrt{-D})$. The Hilbert class polynomial $P_D(X)$ is defined by the minimum polynomial of K over $Q(\sqrt{-D})$. In the construction of elliptic curves by complex multiplication, the fact that the j-invariants of elliptic curves $E/F(p)$ are given as a solution of a Hilbert class polynomial $P_D(X)$ mod p is used.

NOTE 1 These facts are described in References [14] and [17].

NOTE 2 Online databases of Hilbert class polynomials are available in Reference [22].

A.3 Cryptographic pairing

A cryptographic pairing e_n satisfies the conditions of non-degeneracy, bilinearity, and computability. A pairing e_n is defined over $< G_1 > \times < G_2 >$ as follows:

$e_n: < G_1 > \times < G_2 > \rightarrow \mu_n$

where

$< G_1 >$ and $< G_2 >$ are the cyclic groups of order n;

μ_n is the cyclic group of the nth roots of unity.

A pairing e_n is realized by restricting the domain of the Weil, Tate pairings or optimal Ate pairings[37].

A.4 Pell equation

The Pell equation is of the form:

$$T^2 - dU^2 = \pm 1$$

where d is a fixed integer. In the construction of elliptic curves by complex multiplication, the Pell equation with a positive integer d that is not a perfect square is used. Then, all positive integer solutions of (T, U) are given by using the least positive solution (T_0, U_0) with the smallest $U_0 > 0$ as follows:

$$T + U\sqrt{d} = (T_0 + U_0\sqrt{d})^k$$

where $k = 1, 2, \ldots$.

NOTE These facts are described in Reference [29].

A.5 Diophantine equation, $x^2 - dy^2 = n$

In the construction of elliptic curves by complex multiplication, the Diophantine equation, $x^2 - dy^2 = n$, is used where n is an integer and d is a positive integer that is not a perfect square. The number of integer solutions of this formula is zero or infinite. An infinite number of integer solutions (x, y) are given by using the least positive solution (T_0, U_0) with the smallest $U_0 > 0$ of the related Pell equation of $T^2 - dU^2 = 1$.

NOTE Details are described in Reference [29].

Annex B
(informative)

Background information on elliptic curve cryptosystems

B.1 Definition of cryptographic problems

B.1.1 Elliptic curve discrete logarithm problem (ECDLP)

For an elliptic curve $E/F(q)$, the base point $G \in E(F(q))$ with order n, and a point $P \in E(F(q))$, the elliptic curve discrete logarithm problem (with respect to the base point G) is to find the integer $x \in (0, n-1)$ such that $P = xG$ if such an x exists.

The security of elliptic curve cryptosystems is based on the believed hardness of the elliptic curve discrete logarithm problem.

B.1.2 Computational elliptic curve Diffie-Hellman problem (ECDHP)

For an elliptic curve $E/F(q)$, the base point $G \in E(F(q))$ with order n, and points $aG, bG \in E(F(q))$, the computational elliptic curve Diffie-Hellman problem is to compute abG.

The security of some elliptic curve cryptosystems is based on the believed hardness of the computational elliptic curve Diffie-Hellman problem.

B.1.3 Decisional elliptic curve Diffie-Hellman problem (ECDDHP)

For an elliptic curve $E/F(q)$, the base point $G \in E(F(q))$ with order n, and points $aG, bG, Y \in E(F(q))$, the decisional elliptic curve Diffie-Hellman problem is to decide whether $Y = abG$ or not.

The security of some elliptic curve cryptosystems is based on the believed hardness of the decisional elliptic curve Diffie-Hellman problem.

B.1.4 Bilinear Diffie-Hellman problem (BDHP)

The bilinear Diffie-Hellman problems are described in two ways according to the corresponding cryptographic bilinear maps.

— For two groups $< G_1 >$ and $< G_2 >$ with order n, a cryptographic bilinear map $e_n : < G_1 > \times < G_2 > \rightarrow \mu_n$, $aG_1, bG_1 \in < G_1 >$, and $aG_2, cG_2 \in < G_2 >$, the bilinear Diffie-Hellman problem is to compute $e_n(G_1, G_2)^{abc}$.

— For a group $< G_1 >$ with order n, a cryptographic bilinear map $e_n : < G_1 > \times < G_1 > \rightarrow \mu_n$, and $aG_1, bG_1, cG_1 \in < G_1 >$, the bilinear Diffie-Hellman problem is to compute $e_n(G_1, G_1)^{abc}$.

The security of some elliptic curve cryptosystems is based on the believed hardness of the elliptic curve bilinear Diffie-Hellman problem.

B.1.5 Elliptic curve discrete logarithm problem with auxiliary inputs (ECDLP with auxiliary inputs)

The security of some cryptosystems is based on the elliptic curve discrete logarithm problem with auxiliary inputs.

— ECDLP with additional inputs $x^2G, x^3G, \ldots, x^kG$.

— ECDHP with additional inputs a^2G, a^3G, …, a^kG.

— BDHP with additional inputs a^2G_1, a^3G_1, …, a^kG_1.

Three examples of elliptic curve problems with auxiliary inputs are as follows (the notation follows from the original definitions of the problems in B.1.1, B.1.2, and B.1.4).

B.2 Algorithms to determine discrete logarithms on elliptic curves

B.2.1 Hardness of ECDLP

The hardness of ECDLP depends on the selected elliptic curves $E/F(q)$ and the size n of the order of the base point G. The size of n should be chosen large enough to achieve the desired level of security in cryptosystems based on the hardness of the ECDLP, but cannot be less than 160 bits.

The elliptic curve $E/F(q)$ should be chosen to meet the defined security objectives against the following algorithms to solve ECDLP. The size of n should be set to meet the defined security objectives against the baby-step-giant-step algorithm and various variants of the Pollard-ρ algorithm.

B.2.2 Overview of algorithms

The following techniques are available to determine discrete logarithms on an elliptic curve.

— The Pohlig-Silver-Hellman algorithm. This is a divide-and-conquer method which reduces the discrete logarithm problem for an elliptic curve E defined over $F(q)$ to the discrete logarithm in the cyclic subgroups of prime order dividing $\#E(F(q))$.

— The baby-step-giant-step algorithm and various variants of the Pollard-ρ algorithm.

NOTE 1 Different variants of the Pollard-ρ algorithm are described in Reference [34].

— The algorithm of Frey-Rück[20] and the Menezes-Okamoto-Vanstone algorithm[24] which both transform the discrete logarithm problem in a cyclic subgroup of E with prime order n to the smallest extension field $F(q^B)$ of $F(q)$ such that n divides $(q^B - 1)$, where B is called the embedding degree. The Frey-Rück algorithm runs under weaker conditions than the algorithm published by Menezes-Okamoto-Vanstone.

— The algorithm of Araki-Satoh,[30] Smart[33] and Semaev[32] which solves the discrete logarithm problem for an elliptic curve E defined over $F(p^m)$ in the case $\#E(F(p^m)) = p^m$.

Unlike the situation of the discrete logarithm in the multiplicative group of some finite field, there is no known "index-calculus" available in the case of elliptic curves. As for attacks using covering for special type of covers, e.g. the Weil descent attack, the GHS attack, see Chapter 22 of Reference [12].

NOTE 2 The Pohlig-Silver-Hellman and baby-step-giant-step algorithms generally work on all kinds of elliptic curves while the Frey-Rück, the Menezes-Okamoto-Vanstone, Araki-Satoh, Smart, and Semaev algorithms work only on curves with special properties.

B.2.3 MOV-condition

Let n be as defined in the set of elliptic curve domain parameters, where n is a prime divisor of $\#E(F(q))$ and q is a power of a prime p. A value B, used for the MOV-condition, is given as the smallest integer such that n divides $q^B - 1$. As mentioned above, the Frey-Rück and Menezes-Okamoto-Vanstone algorithms reduce the discrete logarithm problem in an elliptic curve over $F(q)$ to the discrete logarithm in the finite field $F(q^B)$ for some $B \geq 1$. By using the attack, the difficulty of the discrete logarithm problem in an elliptic curve $E/F(q)$ is related to the discrete logarithm problem in a finite field $F(q^B)$. The subfield-adjusted MOV-condition describes the degree B that ensures the security level of the discrete logarithm

problem in an elliptic curve by the discrete logarithm problem in finite field. For some applications based on the Weil and Tate pairing, a reasonably small value of B such as 6 or more is preferable.

NOTE Information on the degree B is described in Reference [21].

B.2.4 Condition of prime divisor, n

For some cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5, the prime divisor n should satisfy the following conditions: there is no divisor d of $n - 1$ such that $(\log n)^2 < d < n^{1/2}$ and there is no divisor e of $n + 1$ such that $(\log n)^2 < e < n^{1/2}$. The divisors d and e are possibly composite.

NOTE The size of d is related with k in B.1.5, which is the maximum of the largest divisor of $n - 1$ not exceeding the minimum of k and $\sqrt{n}$. Further detailed information on d and e is given in Reference [15].

B.2.5 Pairing–friendly curves

Because of the existence of the MOV reduction, the security of pairings defined over $F(q)$ is determined by both the hardness of ECDLP on a subgroup or quotient group of the elliptic curve and the hardness of the discrete logarithms problem on a quotient of the multiplicative group $F(q^B)$. The following techniques are available to determine discrete logarithms on a quotient of the multiplicative group $F(q^B)$:

— the algorithm of Barbulescu-Gaudry-Joux-Thomé[39] which solves the discrete logarithm problem in finite fields of small characteristics;

— the algorithm Joux-Pierrot,[43] Barbulescu-Gaudry-Kleinjung,[40] Barbulescu-Gaudry-Guillevic-Morain,[41] Sarkar-Singh,[47] Kim-Barbulescu[44] and Kim-Jeong[45] which solves the discrete logarithm problems in the non-prime finite fields.

The exTNFS attack proposed by Kim et al[44] affects the security level for BN and BLS. Thus, the security parameter should be chosen considering this attack. For example, Reference [39] suggests that a characteristic p of BN and BLS of embedding degree 12 should be 461 bits or large to reach the 128-bit security level. Also, already existing curves are affected by the exTNFS attack, so there is a possibility that these curves no longer satisfy their previous security levels. Another precise analysis of the complexity of the discrete logarithm problem over various pairing-friendly curves can be found in Reference [42].

Annex C
(informative)

Background information on constructing elliptic curves by complex multiplication

C.1 General construction (prime case)

The following algorithm produces an elliptic curve E over $F(p)$ with the given number of rational points N.

NOTE 1 The algorithm is based on Reference [18] which is applied to the primality proving Reference [11].

Input: the prime p, the definition field $F(p)$ and the number of points $N = rn$, where n is the largest prime divisor of N and r is a cofactor.

Output: curve parameters of elliptic curve E with $\#E(F(p)) = N$ and base point G.

a) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5. If not, then execute a new input.

b) Set $t = p + 1 - N$.

c) Choose a pair of integers (D,V) such that $4p - t^2 = DV^2$.

d) Construct the Hilbert class polynomial $P_D(X)$.

e) Find a solution j_0 in $F(p)$ of $P_D(X) = 0 \bmod p$.

f) Choose $c \in F(p)^*$ and construct an elliptic curve $E_{D,j_0,c}$ over $F(p)$ with the j-invariant j_0, coefficient c, and $P_D(X)$.

   1) $E_{D,j_0,c} : y^2 = x^3 + [3c^2 j_0 / (1\,728 - j_0)]x + 2c^3 j_0 / (1\,728 - j_0)$ (if $j_0 \neq 0_F$, 1 728).

   2) $E_{D,j_0,c} : y^2 = x^3 + c$ (if $j_0 = 0_F$).

   3) $E_{D,j_0,c} : y^2 = x^3 + cx$ (if $j_0 = 1\,728$).

g) Construct a random point G on $E_{D,j_0,c}$ $[F(p)]$ such that $G \neq O_E$ and $rG \neq O_E$.

h) Set $G = rG$.

i) If $nG = O_E$, output curve parameters of $E_{D,j_0,c}$ and the base point G. If $nG \neq O_E$, go to step f) to choose another c.

NOTE 2 Any pair of integers (D,V) such that $4p - t^2 = DV^2$ can be used in step c).

NOTE 3 The definition of the Diophantine equation used in step c) is given in A.5.

NOTE 4 The definition of the Hilbert class polynomial $P_D(X)$ is given in A.2.

C.2 Miyaji-Nakabayashi-Takano (MNT) curve

The following algorithm produces an elliptic curve E over $F(p)$ with embedding degree $B = 6$. The pairing and the embedding degree are described in A.3 and B.2.2 respectively.

NOTE 1 Some information and an algorithm for generating an MNT curve with $B = 3$ are given in Reference [25].

NOTE 2 MNT curves can be constructed, not only with $B = 6$, but also with $B = 3$ and 4.

Input: lower and upper bound (odd integer) p_{min} and p_{max} for the definition field (in bits) and upper bound D_{max} for size of D.

Output: prime p, curve parameters of elliptic curve $E/F(p)$, the order $n = \#E(F(p))$, and basepoint G.

a) Choose a small positive integer $D < D_{max}$ such that $D \equiv 3 \pmod 8$ and go to step c).

b) If such D does not exist, then stop and output "fail".

c) Find a pair of integers (T,U) with the smallest $U > 0$ that satisfies $T^2 - 3DU^2 = 1$ using the continued fraction algorithm.

d) Find a pair of integers (x, y) that satisfies $x^2 - 3Dy^2 = -8$ and $0 \le x < 2U\sqrt{(2D)}, 2\sqrt{(2/D)} \le y < 2T\sqrt{(2/D)}$, using the algorithm of Lagrange. If not, go to step a).

e) $i = 0$.

f) Find a pair of primes (p,n) as follows:

   1) Compute integers x_i and y_i such that $x_i + y_i\sqrt{(3D)} = [x + y\sqrt{(3D)}]\,[T + U\sqrt{(3D)}]^i$.

      NOTE 3 Not all solutions can be derived in this way.

   2) If $x_i \equiv 1 \pmod 6$, then $s = (x_i - 1)/6$ and $p = 4s^2 + 1$;

      i) or if $x_i \equiv -1 \pmod 6$, then $s = (x_i + 1)/6$ and $p = 4s^2 + 1$;

      ii) or, $i = i + 1$ and go to step 1).

   3) If $p < p_{min}$, then $i = i + 1$ and go to step 1).

   4) If $p > p_{max}$, then go to step a).

   5) If p is prime, then $n_1 = 4s^2 + 2s + 1$ and $n_2 = 4s^2 - 2s + 1$;

   6) If p is not prime, then $i = i + 1$ and go to step 1).

   7) If $x_i \equiv 1 \pmod 6$, then $n = n_1$;

      i) or $n = n_2$.

   8) If n is prime, then go to step g);

      i) or, $i = i + 1$ and go to step 1).

g) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5. If not, then go to step a).

h) Construct the Hilbert class polynomial $P_D(X)$.

i) Find a solution j_0 in $F(p)$ of $P_D(X) = 0 \bmod p$.

j) Choose $c \in F(p)^*$ and construct an elliptic curve over $F(p)$ with the j-invariant j_0.

1) $E_{D,j_0,c} : y^2 = x^3 + [3c^2 j_0 / (1\,728 - j_0)]x + 2c^3 j_0 / (1\,728 - j_0)$ (if $j_0 \neq 0_F$, 1 728).

2) $E_{D,j_0,c} : y^2 = x^3 + c$ (if $j_0 = 0_F$).

3) $E_{D,j_0,c} : y^2 = x^3 + cx$ (if $j_0 = 1\,728$).

k) Construct a random point G on $E_{D,j_0,c}$ $[F(p)]$, not equal to the point at infinity O_E.

l) If $nG = O_E$, output p, E, n, and G.

m) If $nG \neq O_E$, go to step j) to choose another $c \in F(p)^*$.

NOTE 4 The definition of the Hilbert class polynomial $P_D(X)$ is given in A.2.

NOTE 5 The continued fraction algorithm in step c) is given in Reference [28].

NOTE 6 The algorithm of Lagrange in step d) is given in References [23] and [26].

NOTE 7 A technique for speeding up a protocol based on a bilinear pairing is described in Reference [13].

C.3 Freeman curve (F curve)

The following algorithm produces an elliptic curve E over $F(p)$ with embedding degree $B = 10$. The embedding degree is described in B.2.2.

NOTE 1 Detailed information is given in Reference [19].

Input: lower and upper bound $p_{\min}$ and $p_{\max}$ for the size of the definition field (in bits) and upper bound $D_{\max}$ for size of D.

Output: prime p, curve parameters of elliptic curve $E/F(p)$, the order $n = \#E(F(p))$, and basepoint G.

a) Choose a small positive integer $D < D_{\max}$ such that $D \equiv 43$ or $67 \pmod{120}$ and $15D$ is square-free and go to step c).

b) If such D does not exist, then stop and output "fail".

c) Find a pair of integers (T,U) with the smallest $U > 0$ that satisfies $T^2 - 15DU^2 = 1$ using the continued fraction algorithm.

d) Let $g = T - U\sqrt{(15D)}$.

e) Find a pair of integers (x,y) that satisfies $x^2 - 15Dy^2 = -20$ and $0 \leq x < 10U\sqrt{(3D)}, 2\sqrt{1}/(3D) \leq y < 2T\sqrt{1}/(3D)$ using the algorithm of Lagrange. If not, go to step a).

f) For the current solution (x, y).

1) If $x = \pm 5 \pmod{15}$, then:

i) Let $s = (-5 \pm x)/ 15$.

ii) Let $p = 25s^4 + 25s^3 + 25s^2 + 10s + 3$.

iii) Let $n = 25s^4 + 25s^3 + 15s^2 + 5s + 1$.

2) or, go to step 6).

3) If $p > p_{\max}$, go to step a) to choose a new D.

4) or if $p < p_{\min}$, then go to step 6).

5) If p and n are primes, go to step g).

6) Find a pair of integers (x', y') such that $x' + y'\sqrt{15D} = (x + y\sqrt{15D})\cdot g$.

NOTE 2 Not all solutions can be derived in this way.

7) Let $x = x'$ and $y = y'$ and return to step 1).

g) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5. If not, then go to step a).

h) Construct the Hilbert class polynomial $P_D(X)$.

i) Find a solution j_0 in $F(p)$ of $P_D(X) = 0 \bmod p$.

j) Choose $c \in F(p)^*$ and construct an elliptic curve E over $F(p)$ with j-invariant j_0:

1) $E_{D,j_0,c} : y^2 = x^3 + [3c^2 j_0 / (1\,728 - j_0)]x + 2c^3 j_0 / (1\,728 - j_0)$ (if $j_0 \neq 0_F$, 1 728).

2) $E_{D,j_0,c} : y^2 = x^3 + c$ (if $j_0 = 0_F$).

3) $E_{D,j_0,c} : y^2 = x^3 + cx$ (if $j_0 = 1\,728$).

k) Construct a random point G on $E_{D,j_0,c}$ $[F(p)]$, not equal to the point at infinity O_E.

l) If $nG = O_E$, output p, E, n, and G.

m) or, go to Step j) to choose another $c \in F(p)^*$.

NOTE 3 The definition of the Hilbert class polynomial $P_D(X)$ in step i) is given in A.2.

NOTE 4 The continued fraction algorithm in step c) is given in Reference [28].

NOTE 5 The algorithm of Lagrange in step f) is given in References [23] and [26].

NOTE 6 A technique to speed up a protocol based on a bilinear pairing is described in Reference [13].

C.4 Cocks-Pinch (CP) curve

The following algorithm produces an elliptic curve E over $F(p)$ with arbitrary embedding degree B. The embedding degree is described in B.2.2.

NOTE 1 Detailed information is given in Reference [14].

Input: a positive integer B and a set R of prime numbers n (n–1 is divisible by B).

Output: prime p, curve parameters of elliptic curve $E/F(p)$, the order $n\cdot r = \#E(F(p))$, and basepoint G.

a) Choose a small square-free positive integer D and n in R such that $-D$ is a square mod n.

b) Find a B-th primitive root of unity z in $F(n)$.

c) $t' = z + 1$.

d) $y' = (t'-2)/\sqrt{(-D)} \pmod n$.

e) Let t be an integer such that t is equal to t' mod n, and let y be an integer such that y is equal to y' mod n.

f) $p = (t^2 + Dy^2) / 4$.

NOTE 2 $t = t'$ and $y = y'$ can be used.

g) If p is not prime, then go to step a).

h) Test whether the prime divisor n satisfies the condition described in B.2.4 for cryptosystems based on ECDLP, ECDHP, or BDHP with auxiliary inputs as in B.1.5. If not, then go to step a).

i) Construct the Hilbert class polynomial $P_D(X)$.

j) Find a solution j_0 in $F(p)$ of $P_D(X) = 0 \bmod p$.

k) Choose $c \in F(p)^*$ and construct an elliptic curve over $F(p)$ with the j-invariant j_0.

   1) $E_{D,j_0,c} : y^2 = x^3 + [3c^2 j_0 / (1\,728 - j_0)]x + 2c^3 j_0 / (1\,728 - j_0)$ (if $j_0 \neq 0_F$, 1 728).

   2) $E_{D,j_0,c} : y^2 = x^3 + c$ (if $j_0 = 0_F$).

   3) $E_{D,j_0,c} : y^2 = x^3 + cx$ (if $j_0 = 1\,728$).

l) Set a cofactor $r = (p + 1 - t) / n$.

m) Construct a random point G on $E_{D,j_0,c}$ $[F(p)]$ such that $G \neq O_E$ and $rG \neq O_E$.

n) Set $G = rG$.

o) If $nG = O_E$, output n, G, and the elliptic curve E.

p) or, go to step k) to choose another $c \in F(p)^*$.

NOTE 3 The definition of the Hilbert class polynomial $P_D(X)$ in step c) is given in A.2.

NOTE 4 A technique for speeding up a protocol based on a bilinear pairing is described in Reference [13].

Annex D
(informative)

Numerical examples

D.1 Numerical examples of verifiably pseudo-random elliptic curves

D.1.1 General

The parameters are chosen from a seed using SHA-1.

NOTE A hash-function used in these examples requires only pre-image and second-pre-image resistance property. Parameters are explained in Reference [9].

D.1.2 Elliptic curve over a prime field (192 bits)

p	ffffffff ffffffff ffffffff fffffffe ffffffff ffffffff
	$2^{192}-2^{64}-1$
a	ffffffff ffffffff ffffffff fffffffe ffffffff fffffffc
b	64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
(seed) X	3045ae6f c8422f64 ed579528 d38120ea e12196d5
(compressed) G	03 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
(uncompressed) G	04 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
n	ffffffff ffffffff ffffffff 99def836 146bc9b1 b4d22831
(cofactor) r	1

D.1.3 Elliptic curve over a prime field (224 bits)

p	ffffffff
	ffffffff ffffffff ffffffff 00000000 00000000 00000001
	$2^{224}-2^{96}+1$
a	ffffffff
	ffffffff ffffffff fffffffe ffffffff ffffffff fffffffe
b	b4050a85
	0c04b3ab f5413256 5044b0b7 d7bfd8ba 270b3943 2355ffb4
(seed) X	bd713447 99d5c7fc dc45b59f a3b9ab8f 6a948bc5
(compressed) G	02 b70e0cbd
	6bb4bf7f 321390b9 4a03c1d3 56c21122 343280d6 115c1d21

(uncompressed) G

04 b70e0cbd 6bb4bf7f
321390b9 4a03c1d3 56c21122 343280d6 115c1d21 bd376388
b5f723fb 4c22dfe6 cd4375a0 5a074764 44d58199 85007e34

n

ffffffff
ffffffff ffffffff ffff16a2 e0b8f03e 13dd2945 5c5c2a3d

(cofactor) r

1

D.1.4 Elliptic curve over a prime field (256 bits)

p

ffffffff 00000001
00000000 00000000 00000000 ffffffff ffffffff ffffffff

$2^{224}(2^{32}-1) + 2^{192}+2^{96}-1$

a

ffffffff 00000001
00000000 00000000 00000000 ffffffff ffffffff fffffffc

b

5ac635d8 aa3a93e7
b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b

(seed) X

c49d3608 86e70493 6a6678e1 139d26b7 819f7e90

(compressed) G

03 6b17d1f2 e12c4247
f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296

(uncompressed) G

04 6b17d1f2 e12c4247 f8bce6e5 63a440f2
77037d81 2deb33a0 f4a13945 d898c296 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16
2bce3357 6b315ece cbb64068 37bf51f5

n

ffffffff 00000000
ffffffff ffffffff bce6faad a7179e84 f3b9cac2 fc632551

(cofactor) r

1

D.1.5 Elliptic curve over a prime field (384 bits)

p

ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff fffffffe ffffffff 00000000 00000000 ffffffff

$2^{384}-2^{128}-2^{96}+2^{32}-1$

a

ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff fffffffe ffffffff 00000000 00000000 fffffffc

b

b3312fa7 e23ee7e4 988e056b e3f82d19 181d9c6e fe814112
0314088f 5013875a c656398d 8a2ed19d 2a85c8ed d3ec2aef

(seed) X

a335926a a319a27a 1d00896a 6773a482 7acdac73

(compressed) G

03 aa87ca22 be8b0537 8eb1c71e f320ad74 6e1d3b62 8ba79b98
59f741e0 82542a38 5502f25d bf55296c 3a545e38 72760ab7

(uncompressed) G

```
04 aa87ca22 be8b0537 8eb1c71e f320ad74 6e1d3b62 8ba79b98
59f741e0 82542a38 5502f25d bf55296c 3a545e38 72760ab7 3617de4a 96262c6f
5d9e98bf 9292dc29 f8f41dbd 289a147c e9da3113 b5f0b8c0 0a60b1ce 1d7e819d
                                                 7a431d7c 90ea0e5f
```

n

```
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
c7634d81 f4372ddf 581a0db2 48b0a77a ecec196a ccc52973
```

(cofactor) r

```
1
```

D.1.6 Elliptic curve over a prime field (521 bits)

p

```
01ff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
                                                              2^521-1
```

a

```
01ff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffc
```

b

```
0051 953eb961 8e1c9a1f 929a21a0 b68540ee
a2da725b 99b315f3 b8b48991 8ef109e1 56193951 ec7e937b 1652c0bd 3bb1bf07
                                    3573df88 3d2c34f1 ef451fd4 6b503f00
```

(seed) X

```
                                    d09e8800 291cb853 96cc6717 393284aa a0da64ba
```

(compressed) G

```
                                    0200c6 858e06b7 0404e9cd 9e3ecb66 2395b442
9c648139 053fb521 f828af60 6b4d3dba a14b5e77 efe75928 fe1dc127 a2ffa8de
                                    3348b3c1 856a429b f97e7e31 c2e5bd66
```

(uncompressed) G

```
                                          04 00c6858e 06b70404 e9cd9e3e
cb662395 b4429c64 8139053f b521f828 af606b4d 3dbaa14b 5e77efe7 5928fe1d
c127a2ff a8de3348 b3c1856a 429bf97e 7e31c2e5 bd660118 39296a78 9a3bc004
5c8a5fb4 2c7d1bd9 98f54449 579b4468 17afbd17 273e662c 97ee7299 5ef42640
                  c550b901 3fad0761 353c7086 a272c240 88be9476 9fd16650
```

n

```
                                    01ff ffffffff ffffffff ffffffff ffffffff
fffffffff ffffffff ffffffff fffffffa 51868783 bf2f966b 7fcc0148 f709a5d0
                                    3bb5c9b8 899c47ae bb6fb71e 91386409
```

(cofactor) r

```
1
```

D.2 Numerical examples of BN curve

D.2.1 General

All of the following examples are chosen so that p is the prime satisfying $p = 3 \pmod 4$ and $p = 4 \pmod 9$ or $p = 1 \pmod 9$ for the parameter u with low Hamming weight of the non-adjacent form, allowing the extension field $F(p^2)$ to be represented as $F(p)[i]/(i^2 + \lambda)$ with a positive integer and the extension field $F(p^{2m})$ to be represented as $F(p^2)[z]\ /(z^m - v(i))$ with a polynomial $v(i)$ for $m = 2, 3, 6$. Computation of square (or cube) roots needed for point and/or pairing compression is also simplified in both $F(p)$ and $F(p^2)$. Furthermore, the curve equation has the form E: $y^2 = x^3 + b$ and the sextic twist $E'/F(p^2)$ of the form E': $y'^2 = x'^3 + b/v$ or E': $y'^2 = x'^3 + bv$ contains a subgroup of order n and cofactor $h = 2p - n$. Finally, the isomorphism ψ: $E'/F(p^2) \rightarrow E/F(p^{12})$ takes the form $\psi(x', y') = (x'vz^{-4}, y'vz^{-3})$ or $\psi(x', y') = (x'v^{-1}z^4, y'v^{-1}z^3)$, with $z^6 = v^{-1}$ or $z^6 = v$, respectively. These properties effectively facilitate the implementation of the (plain or compressed) Tate or Weil pairing e: $E \times E' \rightarrow F(p^{2m})$, with optimal pairings especially

benefiting from the sparse form of u. A detailed information on these examples is given in Reference [13].

D.2.2 Elliptic curve over a prime field (446 bits)

p

```
                                                                24000000 00000000
                                  00240000 0002d000 00000d80 0000021c 00000018 00000000
                                  87000000 0b040000 0057c000 00015c00 00001320 00000067
```

u $2^{110} + 2^{36} + 1$

a 0

b 1d

(compressed) G

```
                                                          02 13fbccdb 660511f0
                                  87aee13e 7f60130f 2092aa3e 06e28b6b 4120ef8d 014340b4
                                  0461fd63 43257d87 55b9ebd7 6d82b03b 2b8c6170 b6053d4a
```

(uncompressed) G

```
                                                          04 13fbccdb 660511f0
                                  87aee13e 7f60130f 2092aa3e 06e28b6b 4120ef8d 014340b4
                                  0461fd63 43257d87 55b9ebd7 6d82b03b 2b8c6170 b6053d4a
                                                             210e6d3b 923a4f44
                                  95494f02 8274c2c2 4531435e 7c4bc1ee fbc9c19e f812f3e9
                                  49d327bf 08d200df ac31fda8 ccd3f42c 5bd3ec1d 79eef088
```

n

```
                                                                24000000 00000000
                                  00240000 0002d000 00000d80 0000021c 00000017 a0000000
                                  87000000 0ad40000 0054c000 00015600 00001260 00000061
```

(cofactor) r 1

NOTE This curve parameter first appeared in Reference [46]. The complexity of the discrete logarithm problem over the curve was further analysed in Reference [42].

D.2.3 Elliptic curve over a prime field (462 bits)

p

```
                                                             2404 80360120 023fffff
                                  fffff6ff 0cf6b7d9 bfca0000 000000d8 12908f41 c8020fff
                                  fffffff6 ff66fc6f f687f640 00000000 2401b008 40138013
```

u $2^{114}+2^{101}-2^{14}-1$

a 0

b 5

(compressed) G

```
                                                       0221a6 d67ef250 191fadba
                                  34a0a301 60b9ac92 64b6f95f 63b3edbe c3cf4b2e 689db1bb
                                  b4e69a41 6a0b1e79 239c0372 e5cd7011 3c98d91f 36b6980d
```