Draft NIST Special Publication 800-186

Recommendations for Discrete Logarithm-Based Cryptography:

Elliptic Curve Domain Parameters

Lily Chen
Dustin Moody
Andrew Regenscheid
Karen Randall

COMPUTER SECURITY

National Institute of Standards and Technology
U.S. Department of Commerce

Draft NIST Special Publication 800-186

Recommendations for Discrete Logarithm-Based Cryptography:

Elliptic Curve Domain Parameters

Lily Chen
Dustin Moody
Andrew Regenscheid
Computer Security Division
Information Technology Laboratory

Karen Randall
Randall Consulting
Dover, NH

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-186-draft

October 2019

U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

Authority

This publication has been developed by NIST in accordance with its statutory responsibilities under the Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3551 *et seq.*, Public Law (P.L.) 113-283. NIST is responsible for developing information security standards and guidelines, including minimum requirements for federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-186
Natl. Inst. Stand. Technol. Spec. Publ. 800-186, 78 pages (October 2019)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-186-draft

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

Public comment period: *October 31, 2019* through *January 29, 2020*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: SP800-186-comments@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA).

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

This recommendation specifies the set of elliptic curves recommended for U.S. Government use. In addition to the previously recommended Weierstrass curves defined over prime fields and binary fields, this recommendation includes two newly specified Montgomery curves, which claim increased performance, side-channel resistance, and simpler implementation when compared to traditional curves. The recommendation also specifies alternative representations for these new curves to allow more implementation flexibility. The new curves are interoperable with those specified by the Crypto Forum Research Group (CFRG) of the Internet Engineering Task Force (IETF).

Keywords

Computer security; discrete logarithm-based groups; elliptic curve cryptography; domain parameters.

Acknowledgements

The authors of SP 800-186 (Lily Chen, Dustin Moody, Andrew Regenscheid, and Karen Randall) wish to thank Rene Struik, Rich Davis, and Scott Simon for their contributions to this document. The authors also acknowledge and appreciate the support from public and private sectors whose constructive and beneficial comments greatly enhanced the utility of this publication, as well as their colleagues who reviewed earlier versions of this document.

Audience

This document is intended for implementers of cryptographic schemes that include the use of elliptic curve cryptography.

Conformance Testing

Conformance testing for implementations of this Recommendation will be conducted within the framework of the Cryptographic Algorithm Validation Program (CAVP) and the Cryptographic Module Validation Program (CMVP). The requirements of this Recommendation are indicated by the word "**shall**." Some of these requirements may be out-of-scope for CAVP or CMVP validation testing, and thus are the responsibility of entities using, implementing, installing or configuring applications that incorporate this Recommendation.

Conformant implementations may perform procedures via an equivalent sequence of operations, provided that these include all cryptographic checks included with the specifications in this document. This is important because the checks are essential for the prevention of subtle attacks.

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

    i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
    ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: SP800-186-comments@nist.gov

Executive Summary

This recommendation specifies the set of elliptic curves recommended for U.S. Government use. It includes:

- Specification of elliptic curves previously specified in FIPS Publication 186-4, *Digital Signature Schemes* [FIPS 186-4]. This includes both elliptic curves defined over a prime field and curves defined over a binary field. Although the specifications for elliptic curves over binary fields are included, these curves are now deprecated.
- Specification of new Montgomery and Edwards curves, which are detailed in *Elliptic Curves for Security* [RFC 7748]. These curves are only to be used with the EdDSA digital signature scheme in FIPS 186-5.
- A reference for the Brainpool curves, specified in [RFC 5639]. These curves are allowed to be used for interoperability reasons.
- Elliptic curves in FIPS 186-4 that do not meet the current bit-security requirements put forward in NIST Special Publication 800-57, Part 1, *Recommendation for Key Management Part 1: General* [SP 800-57], are now legacy-use. They may be used to process already protected information (e.g., decrypt or verify) but not to apply protection to information (e.g., encrypt or sign). Also see NIST Special Publication 800-131A, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths* [SP 800-131A].

This recommendation provides details regarding the group operations for each of the specified elliptic curves and the relationship between the various curve models, allowing flexibility regarding the use of curves most suitable in particular applications. It also gives cryptographic criteria for the selection of suitable elliptic curves and provides more details on finite field arithmetic and data representation than were available in FIPS 186-4.

Table of Contents

1 Introduction

1.1 Background

Elliptic curve cryptography (ECC) has uses in applications involving digital signatures (e.g., Elliptic Curve Digital Signature Algorithm, or ECDSA) and key agreement schemes (e.g., Elliptic Curve Diffie-Hellman, or ECDH). The most widely used curves are usually expressed in short-Weierstrass format. However, curves that are expressed using a different format, such as Montgomery curves and twisted Edwards curves, have garnered academic interest. These curves are claimed to have better performance and increased side-channel resistance.

A number of organizations (e.g., NIST, ANSI X9F, ISO, SEC, and IETF) have developed elliptic curve standards. Other standards-setting organizations, such as the Crypto Forum Research Group (CFRG) of the IETF, have discussed ECC and made recommendations for alternate elliptic curves and digital signatures. In June 2015, NIST organized an ECC workshop to discuss the design of curves that are secure, efficient, and easy to use while also being resilient to a wide range of implementation attacks. Subsequently, NIST solicited public comments on the Digital Signature Standard (FIPS 186-4), requesting specific feedback regarding the digital signature schemes in FIPS 186 as well as possible new recommended elliptic curves. This publication is the result of that input.

1.2 Purpose and Scope

This recommendation provides updated specifications of elliptic curves that are appropriate for use by the U.S. Federal Government for digital signatures. It is intended for use in conjunction with other NIST publications, such as NIST Special Publication SP 800-56A, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm-Based Cryptography* [SP 800-56A]; Federal Information Processing Standard FIPS 186-5, *Digital Signature Standard* [FIPS 186-5]; and related specifications. The key pairs specified here are used for digital signature generation and verification or key agreement only and should not be used for any other purposes.

This recommendation is intended to provide sufficient information for a vendor to implement ECC using asymmetric algorithms in FIPS 140-3 [FIPS 140-3] validated modules.

1.3 Document Organization

The remainder of this document includes the following sections and appendices:

- **Section 2: Glossary of Terms, Symbols, and Abbreviations**
- **Section 3: Overview of Elliptic Curves** — This section details the different curve models being used with this recommendation, including notational conventions.
- **Section 4: Recommended Curves for Federal Government Use** — This section highlights the domain parameters for all elliptic curves recommended for U.S. Government use.
- **References** — This section contains references for additional information and links to documents referenced in the publication.

- **Appendix A: Details of Elliptic Curve Group Operations** — This appendix discusses the group laws for each of the different curve models specified in this recommendation.
- **Appendix B: Relationship Between Curve Models** — This appendix details how different curve models are related and how the coordinates of a point and the domain parameters of a curve in one curve model relate to those in another curve model.
- **Appendix C: Generation Details for Recommended Elliptic Curves** — This appendix describes the cryptographic criteria that guided the selection of suitable elliptic curves and the process by which one of many such suitable elliptic curves is selected.
- **Appendix D: Elliptic Curve Routines** — This appendix details elementary routines for elliptic curves, such as the verification that these curves are indeed well-formed, and point compression.
- **Appendix E: Auxiliary Functions** — This appendix covers mathematical functions that are used to describe elliptic curve operations and representation conversions, such as inversion, and taking square roots.
- **Appendix F: Data Conversion** — This appendix documents the detailed procedure for the conversion of data elements, such as integers, field elements, bit strings and octet strings, and elliptic curve points.
- **Appendix G: Implementation Aspects** — This appendix discusses various implementation aspects of binary curves, including conversions between different field representations; for prime curves, it indicates how the special form of the underlying prime field aids in efficient modular reduction.
- **Appendix H: Other Allowed Elliptic Curves** — This appendix lists other elliptic curves that may be used for interoperability reasons.

2 Glossary of Terms, Symbols, and Abbreviations

2.1 Glossary

Group Order	Cardinality of the group.
Identity	Unique group element 0 for which x+0=x for each group element x, relative to the binary group operator +.
Inverse	For some group element x, the unique element y for which x+y is the identity element relative to the binary group operator + (y is usually denoted as $-x$).
Isogeny	Morphism from a first elliptic curve to a second elliptic curve.
Isomorphism	Morphism that is, in fact, a bijection.
Kernel	For a morphism, the set of group elements that map to the identity element.
l-isogeny	Isogeny with kernel of size l (Note: if l=1, an l-isogeny is an isomorphism).
Morphism	Mapping from a first group to a second group that maintains the group structure.
Point at Infinity	Identity element of a Montgomery curve or a curve in short-Weierstrass form.
Point Order	Smallest multiple of a group element that results in the group's identity element.
Quadratic Twist	Certain elliptic curve related to a specified elliptic curve.
Square	The property that some element x of a finite field GF(q) can be written as $x=z^2$ for some element z in the same field GF(q).

2.2 Symbols and Abbreviations

Selected acronyms and abbreviations used in this publication are defined below.

a mod n	Smallest non-negative integer r so that $a-r$ is a multiple of n.
$\lfloor a \rfloor$	The floor of a; the largest integer that is less than or equal to a. For example, $\lfloor 5 \rfloor = 5$, $\lfloor 5.3 \rfloor = 5$, and $\lfloor -2.1 \rfloor = -3$.

$B_{a,b}$	Elliptic curve in short-Weierstrass form defined over the binary field $GF(2^m)$, with domain parameters a and b.
c	Parameter used in domain parameter generation for some curves $W_{a,b}$ in short-Weierstrass form, where $c = a^2/b^3$ (optional).
D	Domain parameters of elliptic curve.
$E_{a,d}$	Twisted Edwards curve, with domain parameters a and d.
G	Base point of order n of an elliptic curve.
$GF(q)$	Finite field of size q.
$GF(p)$	Prime field of size p, represented by the set of integers $\{0,1, \ldots, p-1\}$.
h	Co-factor of an elliptic curve.
Hf	Half-trace function (for binary fields).
len(a)	The length of a in bits; the integer L, where $2^{L-1} \leq a < 2^L$.
$M_{A,B}$	Montgomery curve, with domain parameters A and B.
n	Order of a prime-order subgroup of elliptic curve.
p	Prime Number.
RBG	Random Bit Generator.
Seed	String from which part of the domain parameters are derived (optional).
tr	Trace of an elliptic curve.
Tr	Trace function (for binary fields).
Type	Indication of elliptic curve type.
u, v	Coordinates on a Montgomery curve.
$W_{a,b}$	Elliptic curve in short-Weierstrass form, with domain parameters a and b.
x, y	Coordinates on a (twisted) Edwards or Weierstrass curve.
x', y'	Coordinates on an Edwards448 curve that correspond to the x,y coordinates on an E448 curve.

0x	Indication of a hexadecimal string.
$\varnothing$	Identity element of an elliptic curve.
\	Indication that an integer value stretches over several lines.

3 Overview of Elliptic Curves

3.1 Non-Binary Curves

3.1.1 Curves in Short-Weierstrass Form

Let GF(q) denote the finite field with q elements, where q is an odd prime power and where q is not divisible by three. Let $W_{a,b}$ be the Weierstrass curve with the defining equation $y^2 = x^3 + a\,x + b$, where a and b are elements of GF(q) with $4\,a^3 + 27\,b^2 \neq 0$. When selecting curve parameters, a *Seed* value may be used to generate the parameters a and b as described in Appendix C.2.1.1.

The points of $W_{a,b}$ are the ordered pairs (x, y) whose coordinates are elements of GF(q) and that satisfy the defining equation (i.e., the affine points), together with the special point ∅ (the "point at infinity"). This set forms a group under the operation of addition on elliptic curves via the "chord-and-tangent" rule, where the point at infinity serves as the identity element. See Appendix A.1.1 for details of the group operation.

3.1.2 Montgomery Curves

Let GF(q) denote the finite field with q elements, where q is an odd prime power. Let $M_{A,B}$ be the Montgomery curve with defining equation $B\,v^2 = u\,(u^2 + A\,u + 1)$, where A and B are elements of GF(q) with $A \neq \pm 2$ and $B \neq 0$. The points of $M_{A,B}$ are the ordered pairs (u, v) whose coordinates are elements of GF(q) and that satisfy the defining equation (i.e., the affine points), together with the special point ∅ (the "point at infinity"). This set forms a group under the operation of addition on elliptic curves via the "chord-and-tangent" rule, where the point at infinity serves as the identity element. See Appendix A.1.2 for details of the group operation.

3.1.3 Twisted Edwards Curves

Let GF(q) denote the finite field with q elements, where q is an odd prime power. Let $E_{a,d}$ be the twisted Edwards curve with defining equation $a\,x^2 + y^2 = 1 + d\,x^2\,y^2$, where a and d are elements of GF(q) with $a, d \neq 0$ and $a \neq d$. The points of $E_{a,d}$ are the ordered pairs (x, y) whose coordinates are elements of GF(q) and that satisfy the defining equation (i.e., the affine points). It can be shown that this set forms a group under the operation addition, where the point (0, 1) serves as the identity element. If a is a square in GF(q), and d is not, the addition formulae are complete, meaning that the formulae work for all inputs on the curve. See Appendix A.1.3 for details of the group operation.

An Edwards curve is a twisted Edwards curve with a=1. Edwards curves are to be used with the EdDSA digital signature scheme [FIPS 186-5].

3.2 Binary Curves

3.2.1 Curves in Short-Weierstrass Form

Let GF(q) denote the finite field with q elements, where $q=2^m$. Let $B_{a,b}$ be the Weierstrass curve with defining equation $y^2 + x\,y = x^3 + a\,x^2 + b$, where a and b are elements of GF(q) with $b \neq 0$.

The points of $B_{a,b}$ are the ordered pairs (x, y) whose coordinates are elements of GF(q) and that satisfy the defining equation (i.e., the affine points), together with the special point $\varnothing$ (the "point at infinity").This set forms a group under the operation of addition on elliptic curves via the "chord-and-tangent" rule, where the point at infinity serves as the identity element. See Appendix A.2.1 for details of the group operation.

4 Recommended Curves for U.S. Federal Government Use

This section specifies the elliptic curves recommended for U.S. Federal Government use and contains choices for the private key length and underlying fields. This includes elliptic curves over prime fields (Section 4.2) and elliptic curves over binary fields (Section 4.3) where each curve takes one of the forms described in Section 3 (referred to as "*Type*" below).

Each recommended curve is uniquely defined by its domain parameters D, which indicate the field GF(q) over which the elliptic curve is defined and the parameters of its defining equation, as well as principal parameters such as the co-factor h of the curve, the order n of its prime-order subgroup, and a designated point $G=(G_x, G_y)$ on the curve of order n (i.e., the "base point").

When ECDSA domain parameters are generated (i.e., the NIST-recommended curves for ECDSA are not used), the value of G **should** be generated canonically (verifiably random). An **approved** hash function (such as those specified in FIPS 180 or FIPS 202) **shall** be used during the generation of ECDSA domain parameters. When generating these domain parameters, the security strength of a hash function used **shall** meet or exceed the security strength associated with the bit length of n.[1]

Let E be an elliptic curve defined over the field GF(q).

The cardinality $|E|$ of the curve is equal to the number of points on the curve and satisfies the equation $|E|=(q+1)-tr$, where $|tr| \leq 2\sqrt{q}$. (Thus, $|E|$ and q have the same order of magnitude.)

The integer tr is called the trace of E over the field GF(q).

The points on E form a commutative group under addition (for the group law for each curve form, see Appendix A). Any point P on the curve is the generator of a cyclic subgroup $\langle P \rangle = \{kP \mid k = 0, 1, 2, \ldots\}$ of E. The *order* of P in E is defined as the cardinality of $<P>$. A curve is cyclic if it is generated by some point on E. All curves of prime order are cyclic, while all curves of order $|E|=h \cdot n$, where n is a large prime number and where h is small number, have a large cyclic subgroup of prime order n.

If R is a point on the curve that is also contained in $\langle P \rangle$, there is a unique integer k in the interval $[0, l-1]$ so that $R=kP$, where l is the order of P in E. This number is called the discrete logarithm of R to the base P. The discrete logarithm problem is the problem of finding the discrete logarithm of R to the base P for any two points P and R on the curve, if such a number exists.

A quadratic twist of E is a curve E' related to E, with cardinality $|E'|=(q+1)+tr$. If E is a curve in one of the curve forms specified in this Recommendation, a quadratic twist of this curve can be expressed using the same curve model, although (naturally) with different curve parameters.

[1] The NIST-recommended curves for ECDSA were generated prior to the formulation of this guidance and using SHA-1, which was the only approved hash function available at that time. Since SHA-1 was considered secure at the time of generation, the curves were made public, and SHA-1 will only be used to validate those curves, the NIST-recommended curves for ECDSA are still considered secure and appropriate for Federal Government use.

For details regarding the generation method of the elliptic curves, see Appendix C.

4.1 Choices of Key Lengths, Underlying Fields, Curves, and Base Points

4.1.1 Choice of Key Lengths

The principal parameters for elliptic curve cryptography are the elliptic curve E and a designated point G on E called the *base point*. The base point has order n, which is a large prime. The number of points on the curve is $h \cdot n$ for some integer h (the *cofactor*), which is not divisible by n. For efficiency reasons, it is desirable to have the cofactor be as small as possible.

All of the curves given below have cofactors 1, 2, or 4. As a result, the private and public keys for a curve are approximately the same length.

4.1.2 Choice of Underlying Fields

For each key length, two kinds of fields are provided:

- A *prime field* is the field GF(p), which contains a prime number p of elements. The elements of this field are the integers modulo p, and the field arithmetic is implemented in terms of the arithmetic of integers modulo p.
- A *binary field* is the field GF(2^m), which contains 2^m elements for some m (called the *degree* of the field). The elements of this field are the bit strings of length m, and the field arithmetic is implemented in terms of operations on the bits.

The security strengths for four ranges of the bit length of n are provided in SP 800-57, Part 1. For the field GF(p), the security strength is dependent on the length of the binary expansion of p. For the field GF(2^m), the security strength is dependent on the value of m. Table 1 provides the bit lengths of the various underlying fields of the curves provided in this appendix. Column 1 lists the ranges for the bit length of n. Column 2 identifies the value of p used for the curves over prime fields, where len(p) is the length of the binary expansion of the integer p. Column 3 provides the value of m for the curves over binary fields.

Table 1: Bit Lengths of the Underlying Fields of the Recommended Curves

Bit Length of n	Prime Field	Binary Field
224 – 255	len(p) = 224	m = 233
256 – 383	len(p) = 256	m = 283
384 – 511	len(p) = 384	m = 409
≥ 512	len(p) = 521	m = 571

4.1.3 Choice of Basis for Binary Fields

To describe the arithmetic of a binary field, it is first necessary to specify how a bit string is to be interpreted. This is referred to as choosing a *basis* for the field. There are two common types of bases: a *polynomial basis* and a *normal basis*.

- A polynomial basis is specified by an irreducible polynomial modulo 2, called the *field polynomial*. The bit string (a_{m-1} ... a_2 a_1 a_0) is used to represent the polynomial

  $$a_{m-1}\, t^{m-1} + \ldots + a_2\, t^2 + a_1\, t + a_0$$

  over GF(2). The field arithmetic is implemented as polynomial arithmetic modulo $p(t)$, where $p(t)$ is the field polynomial.

- A normal basis is specified by an element θ of a particular kind. The bit string (a_0 a_1 a_2 ... a_{m-1}) is used to represent the element

  $$a_0\theta + a_1\theta^2 + a_2\theta^{2^2} + \ldots + a_{m-1}\theta^{2^{m-1}}.$$

  Normal basis field arithmetic is not easy to describe or efficient to implement in general except for a special class called *Type T low-complexity* normal bases. For a given field of degree *m*, the choice of *T* specifies the basis and the field arithmetic (see Appendix G.3).

There are many polynomial bases and normal bases from which to choose. The following procedures are commonly used to select a basis representation:

- *Polynomial Basis*: If an irreducible *trinomial* $t^m + t^k + 1$ exists over GF(2), then the field polynomial $p(t)$ is chosen to be the irreducible trinomial with the lowest-degree middle term t^k. If no irreducible trinomial exists, then a *pentanomial* $t^m + t^a + t^b + t^c + 1$ is selected. The particular pentanomial chosen has the following properties: the second term t^a has the lowest degree *m*; the third term t^b has the lowest degree among all irreducible pentanomials of degree *m* and the second term t^a; and the fourth term t^c has the lowest degree among all irreducible pentanomials of degree *m*, with the second term t^a, and third term t^b.
- *Normal Basis*: Choose the *Type T low-complexity* normal basis with the smallest *T*.

For each binary field, the parameters are given for the above basis representations.

4.1.4 Choice of Curves

Two kinds of curves are given:

- *Pseudorandom* curves are those whose coefficients are generated from the output of a seeded cryptographic hash function. If the domain parameter seed value is given along with the coefficients, it can be easily verified that the coefficients were generated by that method.
- *Special curves* are those whose coefficients and underlying field have been selected to

optimize the efficiency of the elliptic curve operations.

For each curve size range, the following curves are given:

→ A pseudorandom curve over GF(p).

→ A pseudorandom curve over GF(2^m).

→ Special curves over GF(p) called *Edwards curves* and *Montgomery curves*.

→ A special curve over GF(2^m) called a *Koblitz curve* or *anomalous binary curve*.

The pseudorandom curves were generated as specified in Appendix C.3.

4.1.5 Choice of Base Points

Since any point of order n can serve as the base point, users could, in principle, generate their own base points to ensure a cryptographic separation of networks, although this does result in another set of domain parameters. When generating base points, users **should** use a verifiably random method and check the validity of the point generated. See Appendix D.3 for more details. If a base point is generated by another entity, it is recommended that its validity be verified with the procedure in Appendix D.3.3 prior to use.

4.2 Curves over Prime Fields

This section specifies elliptic curves over prime fields recommended for U.S. Federal Government use, where each curve takes the form of a curve in short-Weierstrass form (Section 4.2.1), a Montgomery curve (Section 4.2.2), or a twisted Edwards curve (Section 4.2.3).

4.2.1 Weierstrass Curves

This specification includes pseudorandom Weierstrass curves generated over prime fields P-192, P-224, P-256, P-384, and P-521 (See Sections 4.2.1.1 - 4.2.1.5) and special Weierstrass curves over prime fields W-25519 (Section 4.2.1.6) and W-448 (Section 4.2.1.7). The curves W-25519 and W-448 may provide improved performance of the elliptic curve operations as well as increased resilience against side-channel attacks while allowing for ease of integration with existing implementations.

For each Weierstrass curve,

$$E: y^2 \equiv x^3 + ax + b \pmod{p},$$

the following domain parameters D=($p, h, n, Type, a, b, G, \{Seed, c\}$) are given:

- The prime modulus p
- The cofactor h
  - For pseudorandom curves, the cofactor $h = 1$ so the order n is prime

  - For special curves, the cofactor $h > 1$ so the order n is not prime

- The *Type* is “Weierstrass curve”
- The coefficient a
  - For pseudorandom curves, $a = -3$ was made for reasons of efficiency; see IEEE Std 1363-2000
- The coefficient b
  - For pseudorandom curves, the coefficient b satisfies $b^2 c \equiv -27 \pmod p$
- The base point G with x coordinate G_x and y coordinate G_y
- The 160-bit input *Seed* to the SHA-1 hash algorithm in Appendix C.3 for pseudorandom curves. *Seed* is not used with the special curves W-25519 (Section 4.2.1.6) and W-448 (Section 4.2.1.7).
- The output c of the SHA-1 hash algorithm used for pseudorandom curves. The value c is not used with the special curves W-25519 (Section 4.2.1.6) and W-448 (Section 4.2.1.7).

The integers p and n are given in decimal form; bit strings and field elements are given in hexadecimal.

4.2.1.1 P-192

The use of this curve is for legacy-use only. See [FIPS 186-4] for the specification.

4.2.1.2 P-224

The elliptic curve P-224 is a Weierstrass curve $W_{a,b}$ defined over the prime field GF(p) that has order $h \cdot n$, where h=1 and where n is a prime number. This curve has domain parameters D=(p, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is “Weierstrass curve” and the other parameters are defined as follows:

p: $2^{224} - 2^{96} + 1$
= 26959946667150639794667015087019630673557916260026308143510066298881
(=`0xffffffff ffffffff ffffffff ffffffff 00000000 00000000 00000001`)
h: 1
n: 26959946667150639794667015087019625940457807714424391721682722368061
(=`0xffffffff ffffffff ffffffff ffff16a2 e0b8f03e 13dd2945 5c5c2a3d`)
tr: 4733100108545601916421827343930821
(=$(p+1) - h \cdot n$ = `0xe95c 1f470fc1 ec22d6ba a3a3d5c5`)
a: -3
= 26959946667150639794667015087019630673557916260026308143510066298878
(=`0xffffffff ffffffff ffffffff fffffffe ffffffff ffffffff fffffffe`)

b: 18958286285566608000408668544493926415504680968679321075787234672564
(=0xb4050a85 0c04b3ab f5413256 5044b0b7 d7bfd8ba 270b3943 2355ffb4)
G_x: 19277929113566293071110308034699488026831934219452440156649784352033
(=0xb70e0cbd 6bb4bf7f 321390b9 4a03c1d3 56c21122 343280d6 115c1d21)
G_y: 19926808758034470970197974370888749184205991990603949537637343198772
(=0xbd376388 b5f723fb 4c22dfe6 cd4375a0 5a074764 44d58199 85007e34)
Seed: 0xbd713447 99d5c7fc dc45b59f a3b9ab8f 6a948bc5
c: 9585649763196999776159690989286240671136085803543320687376622326267
(=0x5b056c7e 11dd68f4 0469ee7f 3c7a7d74 f7d12111 6506d031 218291fb)

4.2.1.3 P-256

The elliptic curve P-256 is a Weierstrass curve $W_{a,b}$ defined over the prime field GF(*p*) that has order *h*·*n*, where *h*=1 and where *n* is a prime number. This curve has domain parameters *D*=(*p*, *h*, *n*, *Type*, *a*, *b*, *G*, {*Seed*, *c*}), where the *Type* is "Weierstrass curve" and the other parameters are defined as follows:

p: $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
= 115792089210356248762697446949407573530\
086143415290314195533631308867097853951
(=0xffffffff 00000001 00000000 00000000 00000000 ffffffff ffffffff
ffffffff)
h: 1
n: 115792089210356248762697446949407573529\
996955224135760342422259061068512044369
(=0xffffffff 00000000 ffffffff ffffffff bce6faad a7179e84 f3b9cac2
fc632551)
tr: 89188191154553853111372247798585809583
(=$(p+1) - h \cdot n$ = 0x43190553 58e8617b 0c46353d 039cdaaf)
a: −3
= 115792089210356248762697446949407573530\
086143415290314195533631308867097853948
(=0xffffffff 00000001 00000000 00000000 00000000 ffffffff ffffffff
fffffffc)
b: 41058363725152142129326129780047268409\
114441015993725554835256314039467401291
(=0x5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e
27d2604b)
G_x: 48439561293906451759052585252797914202\
762949526041747995844080717082404635286
(=0x6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945
d898c296)
G_y: 36134250956749795798585127919587881956\
611106672985015071877198253568414405109
(=0x4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068

37bf51f5)

Seed: 0xc49d3608 86e70493 6a6678e1 139d26b7 819f7e90

c: 57436011470200155964173534038266061871\
440426244159038175955947309464595790349
(=0x7efba166 2985be94 03cb055c 75d4f7e0 ce8d84a9 c5114abc af317768 0104fa0d)

4.2.1.4 P-384

The elliptic curve P-384 is a Weierstrass curve $W_{a,b}$ defined over the prime field GF(p) that has order $h·n$, where h=1 and where n is a prime number. This curve has domain parameters D=(p, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is "Weierstrass curve" and the other parameters are defined as follows:

p: $2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
= 39402006196394479212279040100143613805079739270465446667948\
293404245721771496870329047266088258938001861606973112319
(=0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe ffffffff 00000000 00000000 ffffffff)

h: 1

n: 39402006196394479212279040100143613805079739270465446667946\
905279627659399113263569398956308152294913554433653942643
(=0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff c7634d81 f4372ddf 581a0db2 48b0a77a ecec196a ccc52973)

tr: 1388124618062372383606759648309780106643088307173319169677
(=(p+1) − $h \cdot n$ = 0x389cb27e 0bc8d21f a7e5f24c b74f5885 1313e696 333ad68d)

a: −3
= 39402006196394479212279040100143613805079739270465446667948\
293404245721771496870329047266088258938001861606973112316
(=0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe ffffffff 00000000 00000000 fffffffc)

b: 27580193559959705877849011840389048093056905856361568521428\
707301988689241309860865136260764883745107765439761230575
(=0xb3312fa7 e23ee7e4 988e056b e3f82d19 181d9c6e fe814112 0314088f 5013875a c656398d 8a2ed19d 2a85c8ed d3ec2aef)

G_x: 26247035095799689268623156744566981891852923491109213387815\
615900925518854738050089022388053975719786650872476732087
(=0xaa87ca22 be8b0537 8eb1c71e f320ad74 6e1d3b62 8ba79b98 59f741e0 82542a38 5502f25d bf55296c 3a545e38 72760ab7)

G_y: 83257109614890299855467512895201081792878530488613155947092\
05902480503199884419224438643760392947333078086511627871
(=0x3617de4a 96262c6f 5d9e98bf 9292dc29 f8f41dbd 289a147c e9da3113 b5f0b8c0 0a60b1ce 1d7e819d 7a431d7c 90ea0e5f)

Seed: 0xa335926a a319a27a 1d00896a 6773a482 7acdac73

c: 1874980186709887347182107097135388878869033900306543902178\
0101954060871745882341382251168574711376101826101037376643
(=0x79d1e655 f868f02f ff48dcde e14151dd b80643c1 406d0ca1
0dfe6fc5 2009540a 495e8042 ea5f744f 6e184667 cc722483)

4.2.1.5 P-521

The elliptic curve P-521 is a Weierstrass curve $W_{a,b}$ defined over the prime field GF(p) that has order $h \cdot n$, where h=1 and where n is a prime number. This curve has domain parameters D=(p, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is "Weierstrass curve" and the other parameters are defined as follows:

p: $2^{521} - 1$
= 6864797660130609714981900799081393217269435300143305409394\
463459185543183397656052122559640661454554977296311391480\
858037121987999716643812574028291115057151
(=0x1ff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff)

h: 1

n: 686479766013060971498190079908139321726943530014330540939\
446345918554318339765539424505774633321719753296399637136\
3321113864768612440380340372808892707005449
(=0x1ff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff fffffffa 51868783 bf2f966b 7fcc0148 f709a5d0
3bb5c9b8 899c47ae bb6fb71e 91386409)

tr: 657877501894328237357444332315020117536\
923257219387276263472201219398408051703
(=$(p+1) - h \cdot n$ = 0x5 ae79787c 40d06994 8033feb7 08f65a2f
c44a3647 7663b851 449048e1 6ec79bf7)

a: −3
= 686479766013060971498190079908139321726943530014330540939\
446345918554318339765605212255964066145455497729631139148 \
0858037121987999716643812574028291115057148
(=0x1ff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff fffffffc)

b: 1093849038073734274511112390766805569936207598951683748994\
5863944959531161507350160137087375737596232485921322967063\
13309438452531591012912142327488478985984
(=0x051 953eb961 8e1c9a1f 929a21a0 b68540ee a2da725b 99b315f3
b8b48991 8ef109e1 56193951 ec7e937b 1652c0bd 3bb1bf07
3573df88 3d2c34f1 ef451fd4 6b503f00)

G_x: 2661740802050217063228768716723360960729859168756973147706\
6713684188029449964278084915450806277719023520942412250655\
58662157113545570916814161637315895999846
(=0xc6 858e06b7 0404e9cd 9e3ecb66 2395b442 9c648139 053fb521

`f828af60 6b4d3dba a14b5e77 efe75928 fe1dc127 a2ffa8de`
`3348b3c1 856a429b f97e7e31 c2e5bd66`)

G_y: 37571800257700204635455072244911836035944551347697624866945\
67779615544477440556316691234405012945539562144444537289428\
522585666729196580810124344277578376784
(=`0x118 39296a78 9a3bc004 5c8a5fb4 2c7d1bd9 98f54449 579b4468`
`17afbd17 273e662c 97ee7299 5ef42640 c550b901 3fad0761`
`353c7086 a272c240 88be9476 9fd16650`)

Seed: `0xd09e8800 291cb853 96cc6717 393284aa a0da64ba`

c: 24207366709569614705877518337783838722729492801746379711063l8\
22395601063635555733389903586634265037857522127726888618270 46\
4382885002006138325182692898444 6519

4.2.1.6 W-25519

The elliptic curve W-25519 is a Weierstrass curve $W_{a,b}$ defined over the prime field GF(p), with p=2^{255}-19, and that has order $h \cdot n$, where h=8 and where n is a prime number. The quadratic twist of this curve has order $h_1 \cdot n_1$, where h_1=4 and where n_1 is a prime number. This curve has domain parameters D=(p, h, n, *Type*, a, b, G), where the *Type* is "Weierstrass curve" and the other parameters are defined as follows:

p: $2^{255}-19$
(=`0x7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff`
`ffffffed`)

h: 8

n: 72370055773322622139731865630429942408\
57116359379907606001950938285454250989
(=2^{252} + `0x14def9de a2f79cd6 5812631a 5cf5d3ed`)

tr: −221938542218978828286815502327069187962
(=$(p+1) - h \cdot n = -$ `0xa6f7cef5 17bce6b2 c09318d2 e7ae9f7a`)

a: 19298681539552699237261830834781317975\
544997444273427339909597334573241639236
(=`0x2aaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaa98`
`4914a144`)

b: 55751746669818908907645289078257140818\
241103727901012315294400837956729358436
(=`0x7b425ed0 97b425ed 097b425e d097b425 ed097b42 5ed097b4 260b5e9c`
`7710c864`)

G_x: 19298681539552699237261830834781317975\
544997444273427339909597334652188435546
(=`0x2aaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa`
`aaaaaaaa aaad245a`)

G_y: 43114425171068552920764898935933967039\

370386198203806730763910166200978582548
(=0x5f51e65e 475f794b 1fe122d3 88b72eb3 6dc2b281 92839e4d d6163a5d 81312c14)

The curve W-25519 is isomorphic to the curve Curve25519 specified in Section 4.2.2.1, where the base point of Curve25519 corresponds to the base point of W-25519, where the point at infinity ∅ of Curve25519 corresponds to the point at infinity ∅ on W-25519 and where the point (u, v) on Curve25519 corresponds to the point $(x, y)=(u+A/3, v)$ on $W_{a,b}$.

See Appendix B.2 for more details.

Note that Curve25519 is <u>not</u> isomorphic with a Weierstrass curve with domain parameter $a = -3$. In particular, this means that one cannot reuse an implementation for elliptic curves with short-Weierstrass form that hard-codes the domain parameter a to -3 to implement Curve25519.

4.2.1.7 W-448

The elliptic curve Curve448 is the Weierstrass curve $W_{a,b}$ defined over the prime field GF(p), with $p=2^{448}-2^{224}-1$, and that has order $h \cdot n$, where h=4 and where n is a prime number. The quadratic twist of this curve has order $h_1 \cdot n_1$, where h_1 =4 and where n_1 is a prime number. This curve has domain parameters D=(p, h, n, *Type*, a, b, G), where the *Type* is "Weierstrass curve" and the other parameters are defined as follows:

p: $2^{448}-2^{224}-1$
(=0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff)
h: 4
n: 181709681073901722637330951972001133588410340171829515070372549795146003961539585716195755291692375963310293709091662304773755859649779
(=2^{446} − 0x8335dc16 3bb124b6 5129c96f de933d8d 723a70aa dc873d6d 54a7bb0d)
tr: 28312320572429821613362531907042076847709625476988141958474579766324
(=$(p+1) - h \cdot n$ =0x1 0cd77058 eec492d9 44a725bf 7a4cf635 c8e9c2ab 721cf5b5 529eec34)
a: 484559149530404593699549205258669689569094240458212040187660132787074885444487181790930922465784363953392589641229091574035657199637535
(=0xaaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaa9 ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe 1a76d41f)
b: 269199527516891440944194002921483160871719022476784466770922295992819380802492878772739401369880202196329216467349495319191685664513904
(=0x5ed097b4 25ed097b 425ed097 b425ed09 7b425ed0 97b425ed 097b425e 71c71c71 c71c71c7 1c71c71c 71c71c71 c71c71c7 1c72c87b 7cc69f70)
G_x: 484559149530404593699549205258669689569094240458212040187660132787074885444487181790930922465784363953392589641229091574035665345629 07
(=0xaaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa aaaaaaaa 00000000 00000000 00000000 00000000 00000000 00000000 0000cb91)

G_y: 3552939267855681752641275020637833348089763993877142718318808984351\
6908878696741000293267376586455091014277414726810583898559529060636 2
(=0x7d235d12 95f5b1f6 6c98ab6e 58326fce cbae5d34 f55545d0 60f75dc2
8df3f6ed b8027e23 46430d21 1312c4b1 50677af7 6fd7223d 457b5b1a)

The curve W-448 is isomorphic to the curve Curve448 specified in Section 4.2.2.2, where the base point of Curve448 corresponds to the base point of W-448, where the point at infinity ∅ of Curve448 corresponds to the point at infinity ∅ on W-448 and where the point (u, v) on Curve448 corresponds to the point $(x, y)=(u+A/3, v)$ on $W_{a,b}$.

See Appendix B.2 for more details.

Note that Curve448 is <u>not</u> isomorphic with a Weierstrass curve with domain parameter $a = -3$. In particular, this means that one cannot reuse an implementation for curves with short-Weierstrass form that hard-codes the domain parameter a to −3 to implement Curve448.

4.2.2 Montgomery Curves

Similar to W-25519 and W-448, Montgomery curves may offer improved performance with improved resistance to side-channel attacks. These curves can also provide a bridge between short-Weierstrass curves and Edwards curves.

4.2.2.1 Curve25519

The elliptic curve Curve25519 is the Montgomery curve $M_{A,B}$ defined over the prime field GF(p), with $p=2^{255}-19$, and with parameters A=486662 and B=1 [RFC 7748]. This curve has order $h \cdot n$, where h=8 and where n is a prime number. For this curve, A^2-4 is not a square in GF(p), whereas A+2 is. The quadratic twist of this curve has order $h_1 \cdot n_1$, where h_1=4 and where n_1 is a prime number. This curve has domain parameters D=(p, h, n, *Type*, A, B, G), where the *Type* is "Montgomery curve" and where the other parameters are defined as follows:

p: $2^{255}-19$
(=0x7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffed)
h: 8
n: 7237005577332262213973186563042994240 8\
5711635937990760600195093828545425098 9
(=2^{252} + 0x14def9de a2f79cd6 5812631a 5cf5d3ed)
tr: −221938542218978828286815502327069187962
(=$(p+1) - h \cdot n = -$ 0xa6f7cef5 17bce6b2 c09318d2 e7ae9f7a)
A: 486662
B: 1
G_u: 9
(=0x9)
G_v: 4311442517106855292076489893593396703 9\
3703861982038067307639101662009785825 48

(=`0x5f51e65e 475f794b 1fe122d3 88b72eb3 6dc2b281 92839e4d d6163a5d 81312c14`)

4.2.2.2 Curve448

The elliptic curve Curve448 is the Montgomery curve $M_{A,B}$ defined over the prime field GF(p), with $p=2^{448}-2^{224}-1$, and with parameters A=156326 and B=1 [RFC 7748]. This curve has order $h\cdot n$, where h=4 and where n is a prime number. For this curve, A^2-4 is not a square in GF(p), whereas A-2 is. The quadratic twist of this curve has order $h_1\cdot n_1$, where $h_1 = 4$ and where n_1 is a prime number. This curve has domain parameters D=(p, h, n, *Type*, A, B, G), where the *Type* is "Montgomery curve" and where the other parameters are defined as follows:

p: $2^{448}-2^{224}-1$
(=`0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff`)

h: 4

n: 181709681073901722637330951972001133588410340171829515070372549795146003961539585716195755291692375963310293709091662304773755859649779
(=2^{446} − `0x8335dc16 3bb124b6 5129c96f de933d8d 723a70aa dc873d6d 54a7bb0d`)

tr: 28312320572429821613362531907042076847709625476988141958474579766324
(=$(p+1)-h\cdot n$ =`0x1 0cd77058 eec492d9 44a725bf 7a4cf635 c8e9c2ab 721cf5b5 529eec34`)

A: 156326

B: 1

G_u: 5
(=`0x5`)

G_v: `355293926785568175264127502063783334808976399387714271831880898435169088786967410002932673765864550910142774147268105838985595290606362`
(=`0x7d235d12 95f5b1f6 6c98ab6e 58326fce cbae5d34 f55545d0 60f75dc2 8df3f6ed b8027e23 46430d21 1312c4b1 50677af7 6fd7223d 457b5b1a`)

The base point of Curve448 corresponds to the base point of E448 and the point at infinity ∅, and the point (0,0) of order two of Curve448 correspond to, respectively, the point (0, 1) and the point (0, −1) of order two on E448. Each other point (u, v) on Curve448 corresponds to the point ($\alpha\, u/v$, $(u+1)/(u-1)$) on E448, where α is the element of GF(p) defined by

α: 197888467295464439538354009753858038256835152591059802148199779196087404232002515713604263127793030747855424464185691766453844835192428
(=`0x45b2c5f7 d649eed0 77ed1ae4 5f44d541 43e34f71 4b71aa96 c945af01 2d182975 0734cde9 faddbda4 c066f7ed 54419ca5 2c85de1e 8aae4e6c`)

See Appendix B.1 for more details.

4.2.3 Twisted Edwards Curves

Edwards curves offer high performance for elliptic curve calculations and protection against side-channel attacks. The Edwards Curve Digital Signature Algorithm (EdDSA) is a digital signature scheme based on twisted Edwards curves and is specified in FIPS 186-5.

4.2.3.1 Edwards25519

The elliptic curve Edwards25519 is the twisted Edwards curve $E_{a,d}$ defined over the prime field GF(p), with $p=2^{255}-19$, and with parameters $a=-1$ and $d=-121665/121666$ (i.e., 37095705934669439343138083508754565189542113879843219016388785533085940283555) [RFC 8032]. This curve has order $h\cdot n$, where h=8 and where n is a prime number. For this curve, a is a square in GF(p), whereas d is not. The quadratic twist of this curve has order $h_1\cdot n_1$, where h_1=4 and where n_1 is a prime number. This curve has domain parameters D=(p, h, n, *Type*, a, d, G), where the *Type* is "twisted Edwards curve" and where the other parameters are defined as follows:

p: $2^{255}-19$
(=0x7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffed)
h: 8
n: 7237005577332262213973186563042994240\
857116359379907606001950938285454250989
(=2^{252} + 0x14def9de a2f79cd6 5812631a 5cf5d3ed)
tr: −221938542218978828286815502327069187962
(=$(p+1) - h\cdot n = -$ 0xa6f7cef5 17bce6b2 c09318d2 e7ae9f7a)
a: −1
d: −121665/121666 = 37095705934669439343138083508754565189\
542113879843219016388785533085940283555
(=0x52036cee 2b6ffe73 8cc74079 7779e898 00700a4d 4141d8ab 75eb4dca 135978a3)
G_x: 15112221349535400772501151409588531511\
454012693041857206046113283949847762202
(=0x216936d3 cd6e53fe c0a4e231 fdd6dc5c 692cc760 9525a7b2 c9562d60 8f25d51a)
G_y: 4/5 = 46316835694926478169428394003475163141\
307993866256225615783033603165251855960
(=0x66666666 66666666 66666666 66666666 66666666 66666666 66666666 66666658)

The curve Edwards25519 is isomorphic to the curve Curve25519 specified in Section 4.2.2.1, where

- the base point of Curve25519 corresponds to the base point of Edwards25519;
- the point at infinity ∅ and the point (0,0) of order two of Curve25519 correspond to, respectively, the point (0, 1) and the point (0, −1) of order two on Edwards25519; and

- each other point (u, v) on Curve25519 corresponds to the point $(\alpha\, u/v, (u-1)/(u+1))$ on Edwards25519, where α is the element of GF(p) defined by

α: 51042569399160536130206135233146329284\
152202253034631822681833788666877215207
(=`0x70d9120b 9f5ff944 2d84f723 fc03b081 3a5e2c2e b482e57d 3391fb55 00ba81e7`).

The inverse mapping from Edwards25519 to Curve25519 is defined by

- mapping the point (0, 1) and the point (0, −1) of order two on Edwards25519 to, respectively, the point at infinity ∅ and the point (0,0) of order two of Curve25519 and
- having each other point (x, y) on Edwards25519 correspond to the point $((1+y)/(1-y), \alpha(1+y)/(1-y)x)$.

See Appendix B.1 for more details.

4.2.3.2 Edwards448

The elliptic curve Edwards448 is the Edwards curve $E_{a,d}$ defined over the prime field GF(p), with $p=2^{448}-2^{224}-1$, and with parameters a=1 and d= −39081 [RFC 8032]. This curve has order $h \cdot n$, where h=4 and where n is a prime number. For this curve, a is a square in GF(p), whereas d is not. The quadratic twist of this curve has order $h_1 \cdot n_1$, where h_1 =4 and where n_1 is a prime number. This curve has domain parameters D=(p, h, n, $Type$, a, d, G), where the $Type$ is "twisted Edwards curve" and where the other parameters are defined as follows:

p: $2^{448}-2^{224}-1$
(=`0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff`)

h: 4

n: 181709681073901722637330951972001133588410340171829515070372549795146003961539585716195755291692375963310293709091662304773755859649779
(=2^{446} − `0x8335dc16 3bb124b6 5129c96f de933d8d 723a70aa dc873d6d 54a7bb0d`)

tr: 28312320572429821613362531907042076847709625476988141958474579766324
(=$(p+1) - h \cdot n$ =`0x1 0cd77058 eec492d9 44a725bf 7a4cf635 c8e9c2ab 721cf5b5 529eec34`)

a: 1

d: −39081
= 726838724295606890549323807888004534353641360687318060281490199180612328166730772686396383698676545930088884461843637361053498018326358
(=`0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffff6756`)

G_x: 224580040295924300187604334099896036246789641632564134246125461686950415467406032909029192869357953282578032075146446173674602635247710
(=0x`4f1970c6 6bed0ded 221d15a6 22bf36da 9e146570 470f1767 ea6de324`

a3d3a464 12ae1af7 2ab66511 433b80e1 8b00938e 2626a82b c70cc05e)

G_y: 29881921007848149267601793044393067343754404015408024209592824137233\
150618983587600353687865541878473398230323350346250053154506283266 0

(=0x693f4671 6eb6bc24 88762037 56c9c762 4bea7373 6ca39840 87789c1e
05a0c2d7 3ad3ff1c e67c39c4 fdbd132c 4ed7c8ad 9808795b f230fa14)

4.2.3.3 E448

The elliptic curve E448 is the Edwards curve $E_{a,d}$ defined over the prime field GF(p), with p=2^{448}–2^{224}–1, and with parameters a=1 and d=39082/39081. This curve has order $h{\cdot}n$, where h=4 and where n is a prime number. For this curve, a is a square in GF(p), whereas d is not. The quadratic twist of this curve has order $h_1{\cdot}n_1$, where h_1 =4 and where n_1 is a prime number. This curve has domain parameters D=(p, h, n, $Type$, a, d, G), where the $Type$ is "twisted Edwards curve" and where the other parameters are defined as follows:

p: 2^{448}–2^{224}–1
(=0xffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff)

h: 4

n: 181709681073901722637330951972001133588410340171829515070372549795146003961539585716195755291692375963310293709091662304773755859649779
(=2^{446} – 0x8335dc16 3bb124b6 5129c96f de933d8d 723a70aa dc873d6d
54a7bb0d)

tr: 28312320572429821613362531907042076847709625476988141958474579766324
(=(p+1) – $h \cdot n$ =0x1 0cd77058 eec492d9 44a725bf 7a4cf635 c8e9c2ab
721cf5b5 529eec34)

a: 1

d: 39082/39081 =
611975850744529176160423220965553317543219696871016626328968936415 0\
87860042636474891785599283666020414768678979989378147065462815545017
(=0xd78b4bdc 7f0daf19 f24f38c2 9373a2cc ad461572 42a50f37 809b1da3
412a12e7 9ccc9c81 264cfe9a d0809970 58fb61c4 243cc32d baa156b9)

G_x: 345397493039729516374008604150537410266655260075183290216406970281 6\
4569507367234443048178775934063322170839158342404178892412456770073 2
(=0x79a70b2b 70400553 ae7c9df4 16c792c6 1128751a c9296924 0c25a07d
728bdc93 e21f7787 ed697224 9de732f3 8496cd11 69871309 3e9c04fc)

G_y: 3/2 =
363419362147803445274661903944002267176820680343659030140745099590 3\
06164083365386343198191849338272965044442230921818680526749009182718
(=0x7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffeL)

The mapping from E448 to Curve448 is defined by mapping the point (0, 1) and the point (0, –1) of order two on E448 to, respectively, the point at infinity ∅ and the point (0,0) of order two of

Curve448 and having each other point (x, y) on E448 correspond to the point $((y + 1)/(y - 1), \alpha(y + 1)/(y-1)x)$. The value of α is specified in 4.2.2.2. See Appendix B.1 for more details.

The curve Edwards448 (specified in Section 4.2.3.2) is 4-isogenous to the curve E448. See Appendix B.4 for further information.

4.3 Curves over Binary Fields

This section specifies elliptic curves over binary fields where each curve takes the form of a curve in short-Weierstrass form and is either a Koblitz curve (Section 4.3.1) or a pseudorandom curve (Section 4.3.2). Due to their limited adoption, elliptic curves over binary fields (i.e., all the curves specified in Section 4.3) are deprecated and may be removed from a subsequent revision to these guidelines to facilitate interoperability and simplify elliptic curve standards and implementations. New implementations should select an appropriate elliptic curve over a prime field from Section 4.2.

Here, the domain parameters a and b for Koblitz curves are elements of the base field GF(2), i.e., b=1 and a=0 or a=1, whereas, for pseudorandom curves, a=1 and b is a nonzero element of $GF(2^m)$.

For each field degree m, a pseudorandom curve is given, along with a Koblitz curve. The pseudorandom curve has the form

$$E: y^2 + x y = x^3 + x^2 + b,$$

and the Koblitz curve has the form

$$E_a: y^2 + x y = x^3 + ax^2 + 1,$$

where a = 0 or 1.

For each pseudorandom curve, the cofactor is h = 2. The cofactor of each Koblitz curve is h = 2 if a = 1, and h = 4 if a = 0.

The coefficients of the pseudorandom curves and the coordinates of the base points of both kinds of curves are given in terms of both the polynomial and normal basis representations discussed in Section 4.1.3.

For each m, the following parameters are given:

Field Representation:

- The normal basis type T
- The field polynomial (a trinomial or pentanomial)

Koblitz Curve:

- The coefficient a
- The base point order n

- The base point *x* coordinate G_x
- The base point *y* coordinate G_y

Pseudorandom curve:

- The base point order *n*

Pseudorandom curve (Polynomial Basis representation):

- The coefficient *b*
- The base point *x* coordinate G_x
- The base point *y* coordinate G_y

Pseudorandom curve (Normal Basis representation):

- The 160-bit input *Seed* to the SHA-1 based algorithm (i.e., the domain parameter seed)
- The coefficient *b* (i.e., the output of the SHA-1 based algorithm)
- The base point *x* coordinate G_x
- The base point *y* coordinate G_y

Integers (such as *T*, *m*, and *n*) are given in decimal form; bit strings and field elements are given in hexadecimal.

4.3.1 Koblitz Curves

4.3.1.1 Curve K-163

The use of this curve is for legacy-use only. See FIPS 186-4 for the specification.

4.3.1.2 Curve K-233

The elliptic curve K-233 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with *m*=233, and with parameters *a*=0 and *b*=1. This curve has order $h \cdot n$, where *h*=4 and where *n* is a prime number. This curve has domain parameters *D*=(*m*, *f*(*z*), *h*, *n*, *Type*, *a*, *b*, *G*, {*Seed*, *c*}), where the *Type* is "Weierstrass curve" and where the other parameters are defined as follows:

f(*z*): $z^{233} + z^{74} + 1$
h: 4
n: 3450873173395281893717377931138512760570940988862252126\
6328087024741343
(=`0x80 00000000 00000000 00000000 00069d5b b915bcd4 6efb1ad5 f173abdf`)
tr: −137381546011108235394987299651366779
(=$(2^m+1) - h \cdot n =$ `-0x1a756e e456f351 bbec6b57 c5ceaf7b`)
a: 0
(=`0x000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`)
b: 1
(=`0x000 00000000 00000000 00000000 00000000 00000000 00000000 00000001`)

Polynomial basis:

G_x: `0x172 32ba853a 7e731af1 29f22ff4 149563a4 19c26bf5 0a4c9d6e efad6126`

G_y: `0x1db 537dece8 19b7f70f 555a67c4 27a8cd9b f18aeb9b 56e0c110 56fae6a3`

Normal basis:

G_x: `0x0fd e76d9dcd 26e643ac 26f1aa90 1aa12978 4b71fc07 22b2d056 14d650b3`

G_y: `0x064 3e317633 155c9e04 47ba8020 a3c43177 450ee036 d6335014 34cac978`

Seed: n/a (binary Koblitz curve)

4.3.1.3 Curve K-283

The elliptic curve K-283 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with m=283, and with parameters a=0 and b=1. This curve has order $h \cdot n$, where h=4 and where n is a prime number. This curve has domain parameters D=(m, $f(z)$, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is “Weierstrass curve” and where the other parameters are defined as follows:

$f(z)$: $z^{283} + z^{12} + z^7 + z^5 + 1$

h: 4

n: 388533778445145814183892381364703781328481\
1733793061324295874997529815829704422603873

(=`0x1ffffff ffffffff ffffffff ffffffff ffffe9ae 2ed07577 265dff7f 94451e06 1e163c61`)

tr: 7777244870872830999287791970962823977569917

($=(2^m+1) - h \cdot n =$ `0x5947 44be2a23 66880201 aeeb87e7 87a70e7d`)

a: 0

(=`0x0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`)

b: 1

(=`0x0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001`)

Polynomial basis:

G_x: `0x503213f 78ca4488 3f1a3b81 62f188e5 53cd265f 23c1567a 16876913 b0c2ac24 58492836`

G_y: `0x1ccda38 0f1c9e31 8d90f95d 07e5426f e87e45c0 e8184698 e4596236 4e341161 77dd2259`

Normal basis:

G_x: `0x3ab9593 f8db09fc 188f1d7c 4ac9fcc3 e57fcd3b db15024b 212c7022 9de5fcd9 2eb0ea60`

G_y: `0x2118c47 55e7345c d8f603ef 93b98b10 6fe8854f feb9a3b3 04634cc8 3a0e759f 0c2686b1`

Seed: n/a (binary Koblitz curve)

4.3.1.4 Curve K-409

The elliptic curve K-409 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with m=409, and with parameters a=0 and b=1. This curve has order $h \cdot n$, where h=4 and where n is a prime number. This curve has domain parameters D=(m, $f(z)$, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is “Weierstrass curve” and where the other parameters are defined as follows:

f(*z*): $z^{409} + z^{8} + 1$
h: 4
n: 3305279843951242994759576540163855199142023414821406096423243\
95022880711289249191050673258457777458014096366590617731358671
(= `0x7fffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffe5f 83b2d4ea 20400ec4 557d5ed3 e3e7ca5b 4b5c83b8 e01e5fcf`)
tr: 10457288737315625927447685387048320737638796957687575791173829
($=(2^{m}+1) - h \cdot n =$ `0x681 f134ac57 7effc4ee aa0a84b0 7060d692 d28df11c 7f8680c5`)
a: 0
(=`0x0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`)
b: 1
(=`0x0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001`)
Polynomial basis:
G_x: `0x060f05f 658f49c1 ad3ab189 0f718421 0efd0987 e307c84c 27accfb8 f9f67cc2 c460189e b5aaaa62 ee222eb1 b35540cf e9023746`
G_y: `0x1e36905 0b7c4e42 acba1dac bf04299c 3460782f 918ea427 e6325165 e9ea10e3 da5f6c42 e9c55215 aa9ca27a 5863ec48 d8e0286b`
Normal basis:
G_x: `0x1b559c7 cba2422e 3affe133 43e808b5 5e012d72 6ca0b7e6 a63aeafb c1e3a98e 10ca0fcf 98350c3b 7f89a975 4a8e1dc0 713cec4a`
G_y: `0x16d8c42 052f07e7 713e7490 eff318ba 1abd6fef 8a5433c8 94b24f5c 817aeb79 852496fb ee803a47 bc8a2038 78ebf1c4 99afd7d6`
Seed: n/a (binary Koblitz curve)

4.3.1.5 Curve K-571

The elliptic curve K-571 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with *m*=571, and with parameters *a*=0 and *b*=1. This curve has order $h \cdot n$, where *h*=4 and where *n* is a prime number. This curve has domain parameters *D*=(*m*, *f*(*z*), *h*, *n*, *Type*, *a*, *b*, *G*, {*Seed*, *c*}), where the *Type* is "Weierstrass curve" and where the other parameters are defined as follows:

f(*z*): $z^{571} + z^{10} + z^{5} + z^{2} + 1$
h: 4
n: 1932268761508629172347675945465993672149463664853217499328\
617625725759571144780212268133978522706711834706712800825\
351461273674974066617311929682421617092503555733685276673
(=`0x 2000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 131850e1 f19a63e4 b391a8db 917f4138 b630d84b e5d63938 1e91deb4 5cfe778f 637c1001`)

tr: −148380926981691413899619140297051490364542\
574180493936232912339534208516828973111459843
($=(2^{m}+1) - h \cdot n =$ `-0x4c614387 c6698f92 ce46a36e 45fd04e2 d8c3612f`

9758e4e0 7a477ad1 73f9de3d 8df04003)
a: 0
(=0x0000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000)
00000000 00000000 00000000 00000000 00000000)
b: 1
(=0x0000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000)
00000000 00000000 00000000 00000000 00000001)
Polynomial basis:
G_x: 0x26eb7a8 59923fbc 82189631 f8103fe4 ac9ca297 0012d5d4 60248048
01841ca4 43709584 93b205e6 47da304d b4ceb08c bbd1ba39
494776fb 988b4717 4dca88c7 e2945283 a01c8972
G_y: 0x349dc80 7f4fbf37 4f4aeade 3bca9531 4dd58cec 9f307a54 ffc61efc
006d8a2c 9d4979c0 ac44aea7 4fbebbb9 f772aedc b620b01a
7ba7af1b 320430c8 591984f6 01cd4c14 3ef1c7a3
Normal basis:
G_x: 0x04bb2db a418d0db 107adae0 03427e5d 7cc139ac b465e593 4f0bea2a
b2f3622b c29b3d5b 9aa7a1fd fd5d8be6 6057c100 8e71e484
bcd98f22 bf847642 37673674 29ef2ec5 bc3ebcf7
G_y: 0x44cbb57 de20788d 2c952d7b 56cf39bd 3e89b189 84bd124e 751ceff4
369dd8da c6a59e6e 745df44d 8220ce22 aa2c852c fcbbef49
ebaa98bd 2483e331 80e04286 feaa2530 50caff60
Seed: n/a (binary Koblitz curve)

4.3.2 Pseudorandom Curves

4.3.2.1 Curve B-163

The use of this curve is for legacy-use only. See FIPS 186-4 for the specification.

4.3.2.2 Curve B-233

The elliptic curve B-233 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with m=233, and with parameter a=1. This curve has order $h \cdot n$, where h=2 and where n is a prime number. This curve has domain parameters D=(m, $f(z)$, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is “Weierstrass curve” and where the other parameters are defined as follows:

$f(z)$: $z^{233} + z^{74} + 1$
h: 2
n: 69017463467905637874347558622770255558398127373450135\
5379383634485463
(=0x100 00000000 00000000 00000000 0013e974 e72f8a69 22031d26 03cfe0d7)
tr: −206777407530349254000433718821372333
(=$(2^m+1) - h \cdot n$ = −0x27d2e9 ce5f14d2 44063a4c 079fc1ad)
a: 1
(=0x000 00000000 00000000 00000000 00000000 00000000 00000000 00000001)
Polynomial basis:

b: 0x066 647ede6c 332c7f8c 0923bb58 213b333b 20e9ce42 81fe115f 7d8f90ad
G_x: 0x0fa c9dfcbac 8313bb21 39f1bb75 5fef65bc 391f8b36 f8f8eb73 71fd558b
G_y: 0x100 6a08a419 03350678 e58528be bf8a0bef f867a7ca 36716f7e 01f81052
Normal basis:
b: 0x1a0 03e0962d 4f9a8e40 7c904a95 38163adb 82521260 0c7752ad 52233279
G_x: 0x18b 863524b3 cdfefb94 f2784e0b 116faac5 4404bc91 62a363ba b84a14c5
G_y: 0x049 25df77bd 8b8ff1a5 ff519417 822bfedf 2bbd7526 44292c98 c7af6e02
Seed: 0x74d59ff0 7f6b413d 0ea14b34 4b20a2db 049b50c3

4.3.2.3 Curve B-283

The elliptic curve B-283 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with m=283, and with parameter a=1. This curve has order $h \cdot n$, where h=2 and where n is a prime number. This curve has domain parameters D=(m, $f(z)$, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is "Weierstrass curve" and where the other parameters are defined as follows:

$f(z)$: $z^{283} + z^{12} + z^7 + z^5 + 1$
h: 2
n: 77706755689029162836778476272940756265696259243769048891\
10919652677004427778737869287 1
(=0x3ffffff ffffffff ffffffff ffffffff ffffef90 399660fc
938a9016 5b042a7c efadb307)
tr: 2863663306391796106224371145726066910599667
(=(2^m+1) − $h \cdot n$ = 0x 20df8cd33e06d8eadfd349f7ab0620a499f3)
a: 1
(=0x0000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000001)
Polynomial basis:
b: 0x27b680a c8b8596d a5a4af8a 19a0303f ca97fd76 45309fa2
a581485a f6263e31 3b79a2f5
G_x: 0x5f93925 8db7dd90 e1934f8c 70b0dfec 2eed25b8 557eac9c
80e2e198 f8cdbecd 0x86b12053
G_y: 0x3676854 fe24141c b98fe6d4 b20d02b4 516ff702 350eddb0
826779c8 13f0df45 be8112f4
Normal basis:
b: 0x157261b 894739fb 5a13503f 55f0b3f1 0c560116 66331022
01138cc1 80c0206b dafbc951
G_x: 0x749468e 464ee468 634b21f7 f61cb700 701817e6 bc36a236
4cb8906e 940948ea a463c35d
G_y: 0x62968bd 3b489ac5 c9b859da 68475c31 5bafcdc4 ccd0dc90
5b70f624 46f49c05 2f49c08c
Seed: 0x77e2b073 70eb0f83 2a6dd5b6 2dfc88cd 06bb84be

4.3.2.4 Curve B-409

The elliptic curve B-409 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with m=409, and with parameter a=1. This curve has order $h \cdot n$, where h=2 and where n is a prime number. This curve has domain parameters D=(m, $f(z)$, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is "Weierstrass curve" and where the other parameters are defined as follows:

$f(z)$: $z^{409} + z^{87} + 1$
h: 2
n: 661055968790248598951915308032771039828404682964281219284648\
798304157774827374805208143723762179110965979867288366567526771
(=0x1000000 00000000 00000000 00000000 00000000 00000000 000001e2
aad6a612 f33307be 5fa47c3c 9e052f83 8164cd37 d9a21173)
tr: -6059503967182126918765909026644927652236777310526686418445029
(=$(2^m+1) - h \cdot n$ = -0x3c5 55ad4c25 e6660f7c bf48f879 3c0a5f07
02c99a6f b34422e5)
a: 1
(=0x0000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000001)

Polynomial basis:
b: 0x021a5c2 c8ee9feb 5c4b9a75 3b7b476b 7fd6422e f1f3dd67 4761fa99
d6ac27c8 a9a197b2 72822f6c d57a55aa 4f50ae31 7b13545f
G_x: 0x15d4860 d088ddb3 496b0c60 64756260 441cde4a f1771d4d b01ffe5b
34e59703 dc255a86 8a118051 5603aeab 60794e54 bb7996a7
G_y: 0x061b1cf ab6be5f3 2bbfa783 24ed106a 7636b9c5 a7bd198d 0158aa4f
5488d08f 38514f1f df4b4f40 d2181b36 81c364ba 0273c706

Normal basis:
b: 0x124d065 1c3d3772 f7f5a1fe 6e715559 e2129bdf a04d52f7 b6ac7c53
2cf0ed06 f610072d 88ad2fdc c50c6fde 72843670 f8b3742a
G_x: 0x0ceacbc 9f475767 d8e69f3b 5dfab398 13685262 bcacf22b 84c7b6dd
981899e7 318c96f0 761f77c6 02c016ce d7c548de 830d708f
G_y: 0x199d64b a8f089c6 db0e0b61 e80bb959 34afd0ca f2e8be76 d1c5e9af
fc7476df 49142691 ad303902 88aa09bc c59c1573 aa3c009a
Seed: 0x4099b5a4 57f9d69f 79213d09 4c4bcd4d 4262210b

4.3.2.5 Curve B-571

The elliptic curve B-571 is a Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), with m=571, and with parameter a=1. This curve has order $h{\cdot}n$, where h=2 and where n is a prime number. This curve has domain parameters D=(m, $f(z)$, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is "Weierstrass curve" and where the other parameters are defined as follows:

$f(z)$: $z^{571} + z^{10} + z^5 + z^2 + 1$
h: 2
n: 386453752301725834469535189093198734429892732970643499865\
723525145151914228956042453614399938941577308313388112192\
6944486246872462816813070234528288303332411393191105285703
(=0x3fffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff e661ce18 ff559873 08059b18 6823851e
c7dd9ca1 161de93d 5174d66e 8382e9bb 2fe84e47)

tr: 9953438501360975865946981915046538223641239\

645234917101676077032749667460757941907544
(=$(2^m+1) - h \cdot n$ = 0x333c63ce 0154cf19 eff4c9cf 2fb8f5c2 7044c6bd
d3c42d85 5d165322 f8fa2c89 a02f6373)

a: 1
(=0x0000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000001)

Polynomial basis:

b: 0x2f40e7e 2221f295 de297117 b7f3d62f 5c6a97ff cb8ceff1 cd6ba8ce
4a9a18ad 84ffabbd 8efa5933 2be7ad67 56a66e29 4afd185a
78ff12aa 520e4de7 39baca0c 7ffeff7f 2955727a

G_x: 0x303001d 34b85629 6c16c0d4 0d3cd775 0a93d1d2 955fa80a a5f40fc8
db7b2abd bde53950 f4c0d293 cdd711a3 5b67fb14 99ae6003
8614f139 4abfa3b4 c850d927 e1e7769c 8eec2d19

G_y: 0x37bf273 42da639b 6dccfffe b73d69d7 8c6c27a6 009cbbca 1980f853
3921e8a6 84423e43 bab08a57 6291af8f 461bb2a8 b3531d2f
0485c19b 16e2f151 6e23dd3c 1a4827af 1b8ac15b

Normal basis:

b: 0x3762d0d 47116006 179da356 88eeaccf 591a5cde a7500011 8d9608c5
9132d434 26101a1d fb377411 5f586623 f75f0000 1ce61198
3c1275fa 31f5bc9f 4be1a0f4 67f01ca8 85c74777

G_x: 0x0735e03 5def5925 cc33173e b2a8ce77 67522b46 6d278b65 0a291612
7dfea9d2 d361089f 0a7a0247 a184e1c7 0d417866 e0fe0feb
0ff8f2f3 f9176418 f97d117e 624e2015 df1662a8

G_y: 0x04a3642 0572616c df7e606f ccadaecf c3b76dab 0eb1248d d03fbdfc
9cd3242c 4726be57 9855e812 de7ec5c5 00b4576a 24628048
b6a72d88 0062eed0 dd34b109 6d3acbb6 b01a4a97

Seed: 0x2aa058f7 3a0e33ab 486b0f61 0410c53a 7f132310

References

[FIPS 140-3] National Institute of Standards and Technology (2019) Security Requirements for Cryptographic Modules. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 140-3. https://doi.org/10.6028/NIST.FIPS.140-3

[FIPS 186-4] National Institute of Standards and Technology (2013) Digital Signature Standard (DSS). (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 186-4. https://doi.org/10.6028/NIST.FIPS.186-4

[FIPS 186-5] National Institute of Standards and Technology (2019) Digital Signature Standard (DSS). (U.S. Department of Commerce, Washington, DC), Draft Federal Information Processing Standards Publication (FIPS) 186-5. https://doi.org/10.6028/NIST.FIPS.186-5-draft

[IEEE 1363] IEEE (2000) *IEEE 1363-2000 – IEEE Standard Specifications for Public Key Cryptography* (IEEE Standards Association, Piscataway, NJ). Available at https://standards.ieee.org/standard/1363-2000.html

[RFC 5639] Lochter M, Merkle J (2010) Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. (Internet Engineering Task Force (IETF)), Request for Comments (RFC) 5639. https://tools.ietf.org/html/rfc5639

[RFC 7748] Langley A, Hamburg M, Turner S (2016) Elliptic Curves for Security. (Internet Research Task Force (IRTF)), Request for Comments (RFC) 7748. https://doi.org/10.17487/RFC7748

[RFC 8032] Josefsson S, Liusvaara I (2017) Edwards-Curve Digital Signature Algorithm (EdDSA). (Internet Research Task Force (IRTF)), Request for Comments (RFC) 8032. https://doi.org/10.17487/RFC8032

[SP 800-56A] Barker EB, Chen L, Roginsky AL, Vassilev A, Davis R (2018) Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-56A, Rev. 3. https://doi.org/10.6028/NIST.SP.800-56Ar3

[SP 800-57] Barker EB (2016) Recommendation for Key Management, Part 1: General. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-57 Part 1, Rev. 4. https://doi.org/10.6028/NIST.SP.800-57pt1r4

[SP 800-131A] Barker EB, Roginsky AL (2019) Transitioning the Use of Cryptographic Algorithms and Key Lengths. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-131A, Rev. 2. https://doi.org/10.6028/NIST.SP.800-131Ar2

Appendix A – Details of Elliptic Curve Group Operations

A.1 Non-Binary Curves

A.1.1 Group Law for Weierstrass Curves

For each point P on the Weierstrass curve $W_{a,b}$, the point at infinity ∅ serves as the identity element, i.e., $P + \varnothing = \varnothing + P = P$.

For each point $P=(x, y)$ on the Weierstrass curve $W_{a,b}$, the point −P is the point $(x, -y)$, and one has $P + (-P) = \varnothing$.

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on the Weierstrass curve $W_{a,b}$, where $P_1 \neq \pm P_2$, and let $Q = P_1 + P_2$. Then $Q = (x, y)$, where

$$x + x_1 + x_2 = \lambda^2 \text{ and } y + y_1 = \lambda(x_1 - x), \text{where } \lambda = (y_2 - y_1)/(x_2 - x_1).$$

Let $P = (x_1, y_1)$ be a point on the Weierstrass curve $W_{a,b}$, where $P \neq -P$, and let $Q = 2P$. Then $Q = (x, y)$, where

$$x + 2x_1 = \lambda^2 \text{ and } y + y_1 = \lambda(x_1 - x), \text{where } \lambda = (3\,x_1^2 + a)/2y_1.$$

A.1.2 Group Law for Montgomery Curves

For each point P on the Montgomery curve $M_{A,B}$, the point at infinity ∅ serves as the identity element, i.e., $P + \varnothing = \varnothing + P = P$.

For each point $P =(u, v)$ on the Montgomery curve $M_{A,B}$, the point −P is the point $(u, -v)$, and one has $P + (-P) = \varnothing$.

Let $P_1 = (u_1, v_1)$ and $P_2 = (u_2, v_2)$ be points on the Montgomery curve $M_{A,B}$, where $P_1 \neq \pm P_2$, and let $Q = P_1 + P_2$. Then $Q = (u, v)$, where

$$u + u_1 + u_2 = \mathrm{B}\,\lambda^2 - \mathrm{A} \text{ and } v + v_1 = \lambda(u_1 - u), \text{where } \lambda = (v_2 - v_1)/(u_2 - u_1).$$

Let $P= (u_1, v_1)$ be a point on the Montgomery curve $M_{A,B}$, where $P \neq -P$, and let $Q = 2P$. Then $Q = (u, v)$, where

$$u + 2u_1 = \mathrm{B}\,\lambda^2 - \mathrm{A} \text{ and } v + v_1 = \lambda(u_1 - u), \text{where } \lambda = (3\,u_1^2 + 2\mathrm{A}u_1 + 1)/2\mathrm{B}v_1.$$

A.1.3 Group Law for Twisted Edwards Curves

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on the twisted Edwards curve $E_{a,d}$ and let $Q = P_1 + P_2$. Then $Q = (x, y)$, where

$$(x, y) = \left(\frac{x_1 y_2 + x_2 y_1}{1 + d x_1 x_2 y_1 y_2}, \frac{y_1 y_2 - a x_1 x_2}{1 - d x_1 x_2 y_1 y_2}\right).$$

For the twisted Edwards curves specified in this recommendation, the domain parameter a is always a square in GF(q), whereas d is not. In this case, the addition formula above is defined for each pair of points. In particular, for each point $P = (x_1, y_1)$ on the twisted Edwards curve $E_{a,d}$, point doubling yields the point $Q = 2P$, where $Q = (x, y)$ and

$$(x, y) = \left(\frac{2x_1y_1}{1 + dx_1{}^2y_1{}^2}, \frac{y_1{}^2 - a\,x_1{}^2}{1 - dx_1{}^2y_1{}^2}\right).$$

Note that (0, 1) is the identity element, since for each point $P = (x, y)$ on the twisted Edwards curve $E_{a,d}$, one has $P + (0, 1) = (x, y) + (0, 1) = (x, y) = P$.

For each point $P= (x, y)$ on the twisted Edwards curve $E_{A,B}$, the inverse point −P is the point $(-x, y)$ and one has $P + (-P) = \varnothing$. The point $(0, -1)$ has order 2.

A.2 Binary Curves

A.2.1 Group Law for Weierstrass Curves

For each point P on the Weierstrass curve $B_{a,b}$, the point at infinity $\varnothing$ serves as the identity element, i.e., $P + \varnothing = \varnothing + P = P$.

For each point $P = (x, y)$ on the Weierstrass curve $B_{a,b}$, the point −P is the point $(x, x + y)$ and one has $P + (-P) = \varnothing$.

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on the Weierstrass curve $B_{a,b}$, where $P_1 \neq \pm P_2$, and let $Q = P_1 + P_2$. Then $Q = (x, y)$, where

$$x + x_1 + x_2 = \lambda^2 + \lambda + a \text{ and } (x + y) + y_1 = \lambda(x_1 + x), \text{where } \lambda = (y_2 + y_1)/(x_2 + x_1).$$

Let $P = (x_1, y_1)$ be a point on the Weierstrass curve $B_{a,b}$, where $P \neq -P$, and let $Q = 2P$. Then $Q = (x, y)$, where

$$x = \lambda^2 + \lambda + a = x_1^2 + b/x_1^2 \text{ and } (x + y) + y_1 = \lambda(x_1 + x), \text{where } \lambda = x_1 + y_1/x_1.$$

Appendix B – Relationship Between Curve Models

The non-binary curves specified in this recommendation are expressed in different curve models defined over the same field GF(q)—namely as curves in short-Weierstrass form, as Montgomery curves, or as twisted Edwards curves. These curve models are related, as follows.

B.1 Mapping Between Twisted Edwards Curves and Montgomery Curves

One can map points on the Montgomery curve $M_{A,B}$ to points on the twisted Edwards curve $E_{a,d}$, where a=(A+2)/B and d=(A-2)/B and, conversely, map points on the twisted Edwards curve $E_{a,d}$ to points on the Montgomery curve $M_{A,B}$, where A=2(a+d)/(a-d) and where B=4/(a-d). For the curves in this specification, this defines a one-to-one correspondence, which is an isomorphism between $M_{A,B}$ and $E_{a,d}$, thereby showing that the discrete logarithm problem in either curve model is equally hard.

For the Montgomery curves and twisted Edwards curves in this specification, the mapping from $M_{A,B}$ to $E_{a,d}$ is defined by mapping the point at infinity ∅ and the point (0, 0) of order two on $M_{A,B}$ to, respectively, the point (0, 1) and the point (0, −1) of order two on $E_{a,d}$, while mapping every other point (u, v) on $M_{A,B}$ to the point $(x, y)=(u/v, (u-1)/(u+1))$ on $E_{a,d}$. The inverse mapping from $E_{a,d}$ to $M_{A,B}$ is defined by mapping the point (0, 1) and the point (0, −1) of order two on $E_{a,d}$ to, respectively, the point at infinity ∅ and the point (0, 0) of order two on $M_{A,B}$, while every other point (x, y) on $E_{a,d}$ is mapped to the point $(u, v)=((1+y)/(1-y), (1+y)/(1-y)x)$ on $M_{A,B}$.

Implementations may take advantage of this mapping to carry out elliptic curve group operations originally defined for a twisted Edwards curve on the corresponding Montgomery curve, or vice-versa, and translating the result back to the original curve to potentially allow code reuse.

B.2 Mapping Between Montgomery Curves and Weierstrass Curves

One can map points on the Montgomery curve $M_{A,B}$ to points on the Weierstrass curve $W_{a,b}$, where $a=(3-A^2)/3B^2$ and $b=(2A^3-9A)/27B^3$. For the curves in this specification, this defines a one-to-one correspondence, which is an isomorphism between $M_{A,B}$ and $W_{a,b}$, thereby showing that the discrete logarithm problem in either curve model is equally hard.

For the Montgomery curves in this specification, the mapping from $M_{A,B}$ to $W_{a,b}$ is defined by mapping the point at infinity ∅ on $M_{A,B}$ to the point at infinity ∅ on $W_{a,b}$, while mapping every other point (u, v) on $M_{A,B}$ to the point $(x, y)=(u/B+A/3B, v/B)$ on $W_{a,b}$.

Note that not all Weierstrass curves can be mapped to Montgomery curves since the latter have a point of order two and the former may not. In particular, if a Weierstrass curve has prime order—as in the case with the curves P-224, P-256, P-385, and P-521 specified in this recommendation—this mapping is not defined.

This mapping can be used to implement elliptic curve group operations originally defined for a twisted Edwards curve or for a Montgomery curve using group operations on the corresponding elliptic curve in short-Weierstrass form and translating the result back to the original curve to potentially allow code reuse.

Note that implementations for elliptic curves with short-Weierstrass form that hard-code the domain parameter a to $a = -3$ cannot always be used this way since the curve $W_{a,b}$ may not always be expressed in terms of a Weierstrass curve with $a=-3$ via a coordinate transformation. This is, unfortunately, the case with the Montgomery curves and twisted Edwards curves specified in this recommendation.

B.3 Mapping Between Twisted Edwards Curves and Weierstrass Curves

A straightforward method to map points on a twisted Edwards curve to points on a Weierstrass curve is to convert the curve to Montgomery format first. Use the mapping described in Appendix B.1 to map points on a twisted Edwards curve to points on a Montgomery curve. Then use the mapping described in Appendix B.2 to convert points on the Montgomery curve to points on a Weierstrass curve.

B.4 4-Isogenous Mapping

The 4-isogeny map between the Montgomery curve Curve448 and the Edwards curve Edwards448 is given in [RFC 7748] to be:

$$(u, v) = \left(\frac{y^2}{x^2}, \frac{(2 - x^2 - y^2)y}{x^3}\right)$$

$$(x, y) = \left(\frac{4v(u^2-1)}{u^4-2u^2+4v^2+1}\right), \frac{-(u^5-2u^3-4uv^2+u)}{(u^5-2u^2v^2-2u^3v^2+u)}$$

The curve Edwards448 (Section 4.2.3.2) is 4-isogenous to the curve E448 (Section 4.2.3.3), where the base point of Edwards448 corresponds to the base point of E448 and where the identity element (0, 1) and the point (0, –1) of order two of Edwards448 correspond to the identity element (0, 1) on E448. Every other point (x, y) on Edwards448 corresponds to the point on E448, where α is the element of GF(p) defined in Section 4.2.2.2:

$$(x', y') = \left(\frac{\alpha xy}{1 - d\,x^2y^2}, \frac{1 + d\,x^2y^2}{y^2 - x^2}\right)$$

Appendix C – Generation Details for Recommended Elliptic Curves

C.1 General Cryptographic Criteria

All curves recommended in this specification satisfy the following general cryptographic criteria:

1. *Underlying finite field*. The underlying finite field GF(q) **shall** be either a prime number or $q=2^m$ where m is a prime number.
2. *Curve order.* Each curve E defined over the finite field GF(q) **shall** have order $|E|=h\cdot n$, where n is a large prime number, where h is co-prime with n, and where h is small (h is called the co-factor of E). Each curve **shall** have co-factor $h \leq 2^{10}$.
3. *Base point.* Each curve E **shall** have a fixed base point G of prime order n.
4. *Avoiding anomalous curve attack*. Each curve E defined over the finite field GF(q) **shall** have order $|E|\neq q$ so as to avoid attacks using additive transfers.
5. *Large embedding degree.* The elliptic curve discrete logarithm problem in E can be converted to an ordinary discrete logarithm problem defined over the finite field GF(q^t) where t is the smallest positive integer so that $q^t \equiv 1(\text{mod } n)$, called the embedding degree. Each curve **shall** have embedding degree $t \geq 2^{10}$.
6. *Endomorphism field*. For each curve E over GF(q) with trace tr, the (negative) number $Disc=tr^2-4q$ is closely related to the discriminant of the endomorphism field of E. As of the publication of this document, there is no technical rationale for imposing a large lower bound on the square-free part of $|Disc|$, although—except for curves used in pairing-based cryptography—this value is often large. This recommendation does not impose restrictions on the value of the square-free part of $|Disc|$.

C.1.1 Implementation Security Criteria

Each field **shall** have a fixed representation.

C.2 Curve Generation Details

C.2.1 Weierstrass Curves over Prime Fields

C.2.1.1 Curves P-224, P-256, P-384, P-521

Each of the curves P-224 (Section 4.2.1.2), P-256 (Section 4.2.1.3), P-384 (Section 4.2.1.4), and P-521 (Section 4.2.1.5) is a curve $W_{a,b}$ in short-Weierstrass form with prime order (and, thus, co-factor h=1). Each curve is defined over a prime field GF(p) where the prime number is of a special form to allow efficient modular reduction (see Appendix G.1).

The NIST prime curves were generated using the procedure in C.3.1 with *hdigest* = 160 and SHA-1 hash function. The curve parameters a and b are:

1. The parameter a was set to $a \equiv -3 \pmod p$ (this allows optimizations of the group law if implemented via projective coordinates in Weierstrass form);

2. The parameter b was derived in a hard-to-invert way using the procedure in Appendix C.3.1 from a pseudorandom *Seed* value so that the following conditions are satisfied simultaneously:
    a. $4a^3 + 27b^2 \neq 0$ in GF(p);
    b. The curve has prime order n (this implies that $h = 1$); and
    c. The curve satisfies the cryptographic criteria in Appendix C.1.
3. Select a base point $G = (G_x, G_y)$ of order n.

C.2.1.2 Curves W-25519, W-448

The curves W-25519 (Section 4.2.1.6) and W-448 (Section 4.2.1.7) were obtained via an isomorphic mapping (see Appendix B.1).

C.2.2 Montgomery Curves

C.2.2.1 Curve25519

Curve25519 was specified in IETF 7748 by the Crypto Forum Research Group (CFRG). This curve is a Montgomery curve $M_{A,B}$ defined over the field GF(p), where $p=2^{255}-19$ and where the curve has co-factor h=8 and the quadratic twist E_1 has co-factor h_1=4. The prime number is of a special form to allow efficient modular reduction and finite field operations that try and minimize carry effects of operands. The curve parameters A and B are:

1. The parameter B was set to B = 1.
2. The parameter A was selected as the minimum value of |A| so that the following conditions are satisfied simultaneously:
    a. The curve is cyclic (this implies that A^2-4 is not a square in GF(p));
    b. The curve has co-factor h=8 (this implies that A+2 is a square in GF(p));
    c. The quadratic twist has co-factor h'=4;
    d. A has the form A ≡ 2 (mod 4) (this allows optimized implementations of implementations of the group law using the Montgomery ladder); and
    e. The curve and the quadratic twist both satisfy the cryptographic criteria in Appendix C.1.
3. Select the base point $G = (G_x, G_y)$ of order n, where $|G_x|$ is minimal and where G_y is odd.

C.2.2.2 Curve448

This curve is a Montgomery curve $M_{A,B}$ defined over the field GF(p), where $p=2^{448}-2^{224}-1$ and where the curve has co-factor h=4 and the quadratic twist E_1 has co-factor h_1=4. The prime number is of a special form to allow efficient modular reduction and finite field operations that try to minimize the carry effects of operands. The curve parameters A and B are:

1. The parameter B was set to B = 1.
2. The parameter A was selected as the minimum value of |A| so that the following conditions are satisfied simultaneously:
    a. The curve is cyclic (this implies that A^2-4 is not a square in GF(p));
    b. The curve has co-factor $h = 4$ (this implies that A+2 is not a square in GF(p));
    c. The quadratic twist has co-factor $h' = 4$;

   d. A has the form A ≡ 2 (mod 4) (this allows optimized implementations of implementations of the group law using the Montgomery ladder); and
   e. The curve and the quadratic twist both satisfy the cryptographic criteria in Appendix C.1.
3. Select the base point $G = (G_x, G_y)$ of order n, where $|G_x|$ is minimal and where G_y is even.

C.2.3 Twisted Edwards Curves

The twisted Edwards curve Edwards25519 (Section 4.2.3.1) was obtained from the Montgomery curve Curve25519 (Section 4.2.2.1) via an isomorphic mapping.

The Edwards curve E448 Section 4.2.3.3) was obtained from the Montgomery curve Curve448 (Section 4.2.2.2) via an isomorphic mapping.

The Edwards curve Edwards448 (Section 4.2.3.2) was obtained from the curve E448 (Section 4.2.3.3) via a 4-isogenous mapping (see Appendix B.4).

C.2.4 Weierstrass Curves over Binary Fields

C.2.4.1 Koblitz Curves K-233, K-283, K-409, K-571

Each of the curves K-233 (Section 4.3.1.2), K-283 (Section 4.3.1.3), K-409 (Section 4.3.1.4), and K-571 (Section 4.3.1.5) is a curve $B_{a,b}$ in short-Weierstrass form with co-factor h=2 or h=4. Each curve is defined over a binary field GF(2^m), where m is a prime number. For Koblitz curves, the curve parameters a and b are elements of GF(2), with $b = 1$. Hence, for each parameter m, there are only two Koblitz curves, viz. with $a = 0$ and with $a = 1$. Koblitz curves with $a = 0$ have order 0 (mod 4), while those with $a = 1$ have order 2 (mod 4).

The curve parameters a and m are:

1. The parameter a was set to $a = 0$.
2. The set of integers m in the interval [160,600] was determined, so that the following conditions are satisfied simultaneously:
   a. m is a prime number;
   b. The curve has co-factor $h = 4$ or the quadratic twist of this curve has co-factor $h = 2$ (the latter implies that the Koblitz curve defined over the binary field GF(2^m) with $a = 1$ has co-factor $h = 2$); and
   c. The thus determined curve satisfies the cryptographic criteria in Appendix C.1.
3. Select a pair (a, m) from the set determined above.
4. Select an irreducible polynomial $f(z)$ of degree m, where $f(z)$ is selected of a special form so as to allow efficient modular reduction ($f(z)$ is a trinomial or pentanomial).
5. Select a base point $G = (G_x, G_y)$ of order n.

C.2.4.2 Pseudorandom Curves B-233, B-283, B-409, B-571

Each of the curves B-233 (Section 4.3.2.2), B-283 (Section 4.3.2.3), B-409 (Section 4.3.2.4), and B-571 (Section 4.3.2.5) is a curve $B_{a,b}$ in short-Weierstrass form with co-factor $h = 2$. Each curve is defined over a binary field GF(2^m), where m is a prime number, where the prime number is amongst those values for which a binary Koblitz curve exists. The NIST prime curves were generated using the procedure in C.3.3, with *hdigest* = 160 and SHA-1 hash function. The curve parameters *a* and *b* are:

1. The parameter *a* was set to $a = 1$ (this ensures that curves with co-factor $h = 2$ may exist).
2. The parameter *b* was derived in a hard-to-invert way using the procedure in Appendix C.3.3 from a pseudorandom *Seed* value so that the following conditions are satisfied simultaneously:
    a. $b \neq 0$ in GF(p);
    b. The curve has co-factor $h = 2$; and
    c. The curve satisfies the cryptographic criteria in Appendix C.1.
3. Select a base point $G = (G_x, G_y)$ of order *n*.

C.3 Generation and Verification of Pseudorandom Curves

C.3.1 Generation of Pseudorandom Curves (Prime Case)

When generating the NIST pseudo-random curves (i.e, those in Section 4.2.1), *hdigest* = 160 and SHA-1 hash were used.

Inputs:

1. Positive integer *l*
2. Bit-string *s* of length *hdigest*
3. Approved hash function *HASH* with output length of *hdigest* bits and security design strength of at least *requested_security_strength*.

Output: Coefficient *b* used to generate a pseudorandom prime curve.

Process:

Let *l* be the bit length of *p*, and define

$$v = \lfloor (l - 1) / hdigest \rfloor$$
$$w = l - hdigest * v - 1.$$

1. Choose an arbitrary *hdigest*-bit string *s* as the domain parameter *Seed*.
2. Compute $h = HASH(s)$.
3. Let h_0 be the bit string obtained by taking the *w* rightmost bits of *h*.
4. Let *z* be the integer whose binary expansion is given by the *hdigest*-bit string *s*.
5. For *i* from 1 to *v* do:
    5.1 Define the *hdigest*-bit string s_i to be binary expansion of the integer $(z + i) \bmod (2^{hdigest})$.
    5.2 Compute $h_i = HASH(s_i)$.
6. Let *h* be the bit string obtained by the concatenation of $h_0, h_1, \ldots, h_v$ as follows:

$$h = h_0 \parallel h_1 \parallel \ldots \parallel h_v.$$

7. Let *c* be the integer whose binary expansion is given by the bit string *h*.
8. If (($c = 0$ or $4c + 27 \equiv 0 \pmod{p}$)), then go to Step 1.

9. Choose integers a, $b \in \mathrm{GF}(p)$ such that

$$c\, b^2 \equiv a^3 \pmod{p}.$$

(The simplest choice is $a = c$ and $b = c$. However, they may be chosen differently for performance reasons.)

10. Check that the elliptic curve E over GF(p) given by $y^2 = x^3 + ax + b$ has suitable order. If not, go to Step 1.

C.3.2 Verification of Curve Pseudorandomness (Prime Case)

Given the *hdigest* domain parameter seed value *s*, verify that the coefficient *b* was obtained from *s* via the cryptographic hash function *HASH* as follows.

Inputs:

1. Positive integer *l*
2. Bit-string *s* of length *hdigest*
3. Approved hash function *HASH* with output length of *hdigest* bits and security design strength of at least *requested_security_strength*

Output: Verification that the coefficient *b* was obtained from *s* via the cryptographic hash function *HASH*.

Process:

Let *l* be the bit length of *p*, and define

$$v = \lfloor (l - 1) / hdigest \rfloor,$$
$$w = l - hdigest * v - 1.$$

1. Compute $h = HASH(s)$.
2. Let h_0 be the bit string obtained by taking the *w* rightmost bits of *h*.
3. Let *z* be the integer whose binary expansion is given by the *hdigest* -bit string *s*.
4. For $i = 1$ to *v* do
   4.1 Define the *hdigest* -bit string s_i to be binary expansion of the integer $(z + i) \bmod (2^{hdigest})$.
   4.2 Compute $h_i = HASH(s_i)$.
5. Let *h* be the bit string obtained by the concatenation of h_0, h_1, . . . , h_v as follows:

$$h = h_0 \,||\, h_1 \,||\, \ldots \,||\, h_v.$$

6. Let *c* be the integer whose binary expansion is given by the bit string *h*.
7. Verify that $b^2 c \equiv -27 \pmod{p}$.

C.3.3 Generation of Pseudorandom Curves (Binary Case)

Inputs:

1. Prime number *m*
2. Bit-string *s* of length *hdigest*

3. Approved hash function *HASH* with output length of *hdigest* bits and security design strength of at least *requested_security_strength*

Output: Coefficient b used to generate a pseudorandom binary curve.

Process:

Let:

$v = \lfloor (m-1) / hdigest \rfloor$

$w = m - hdigest * v$.

1. Choose an arbitrary *hdigest* -bit string s as the domain parameter seed.
2. Compute $h = HASH(s)$.
3. Let h_0 be the bit string obtained by taking the w rightmost bits of h.
4. Let z be the integer whose binary expansion is given by the *hdigest*-bit string s.
5. For i from 1 to v do:
   5.1 Define the *hdigest* -bit string s_i to be binary expansion of the integer $(z + i) \bmod (2^{hdigest})$.
   5.2 Compute $h_i = HASH(s_i)$.
6. Let h be the bit string obtained by the concatenation of $h_0, h_1, \ldots, h_v$ as follows: $h = h_0 \parallel h_1 \parallel \ldots \parallel h_v$.
7. Let b be the element of $GF(2^m)$ which is represented by the bit string h in the Gaussian Normal Basis (see Appendix G.3.1).
8. Choose an element a of $GF(2^m)$.
9. Check that the elliptic curve E over $GF(2^m)$ given by $y^2 + xy = x^3 + ax^2 + b$ has suitable order. If not, go to Step 1.

C.3.4 Verification of Curve Pseudorandomness (Binary Case)

Given the *hdigest*-bit domain parameter seed value s, verify that the coefficient b was obtained from s via the cryptographic hash function *HASH* as follows.

Inputs:

1. Prime number m
2. Bit-string s of length *hdigest*
3. Approved hash function *HASH* with output length of *hdigest* bits and security design strength of at least *requested_security_strength*

Output: Verification that the coefficient b was obtained from s via the cryptographic hash function HASH.

Process:

Define

$v = \lfloor (m-1) / hdigest \rfloor$

$w = m - hdigest\ v$

1. Compute $h = HASH(s)$.

2. Let h_0 be the bit string obtained by taking the w rightmost bits of h.
3. Let z be the integer whose binary expansion is given by the *hdigest*-bit string s.
4. For $i = 1$ to v do
   4.1 Define the *hdigest*-bit string s_i to be binary expansion of the integer $(z + i) \bmod (2^{160})$.
   4.2 Compute $h_i = HASH(s_i)$.
5. Let h be the bit string obtained by the concatenation of $h_0, h_1, \ldots, h_v$ as follows:
   $$h = h_0 \parallel h_1 \parallel \ldots \parallel h_v.$$
6. Let c be the element of $GF(2^m)$ which is represented by the bit string h in the Gaussian Normal Basis (see Section G.3.1).
7. Verify that $c = b$.

Appendix D — Elliptic Curve Routines

D.1 Public Key Validation

D.1.1 Non-Binary Curves in Short-Weierstrass Form

D.1.1.1 Partial Public Key Validation

Inputs:

1. Weierstrass curve $W_{a,b}$ defined over the prime field GF(p)
2. Point $Q=(x,y)$

Output: ACCEPT or REJECT Q as an affine point on $W_{a,b}$.

Process:

1. If Q is the point at infinity ∅, output REJECT.
2. Verify that x and y are integers in the interval $[0, p-1]$. Output REJECT if verification fails.
3. Verify that (x, y) is a point on the $W_{a,b}$ by checking that (x, y) satisfies the defining equation $y^2 = x^3 + a\,x + b$ where computations are carried out in GF(p). Output REJECT if verification fails.
4. Otherwise output ACCEPT.

D.1.1.2 Full Public Key Validation

Inputs:

1. Weierstrass curve $W_{a,b}$ defined over the prime field GF(p)
2. Point Q

Output: ACCEPT or REJECT Q as a point on $W_{a,b}$ of order n.

Process:

1. Perform partial public key validation on Q using the procedure of Appendix D.1.1.1. Output REJECT if this procedure outputs REJECT.
2. Verify that $n\,Q = ∅$. Output REJECT if verification fails.
3. Otherwise, output ACCEPT.

D.1.2 Montgomery Curves

D.1.2.1 Partial Public Key Validation

Inputs:

1. Montgomery curve $M_{A,B}$ defined over the prime field GF(p)

2. Point $Q=(u, v)$

Output: ACCEPT or REJECT Q as an affine point on $M_{A,B}$.

Process:

1. If Q is the point at infinity ∅, output REJECT.
2. Verify that both u and v are integers in the interval $[0, p-1]$. Output REJECT if verification fails.
3. Verify that (u, v) is a point on the $M_{A,B}$ by checking that (u, v) satisfies the defining equation $v^2 = u (u^2 + A\, u + 1)$ where computations are carried out in GF(p). Output REJECT if verification fails.
4. Otherwise output ACCEPT.

D.1.2.2 Full Public Key Validation

Inputs:

1. Montgomery curve $M_{A,B}$ defined over the prime field GF(p)
2. Point Q

Output: ACCEPT or REJECT Q as a point on $M_{A,B}$ of order n.

Process:

1. Perform partial public key validation on Q using the procedure of Appendix D.1.2.1. Output REJECT if this procedure outputs REJECT.
2. Verify that $n\,Q = \varnothing$. Output REJECT if verification fails.
3. Otherwise output ACCEPT.

D.1.3 Twisted Edwards Curves

D.1.3.1 Partial Public Key Validation

Inputs:

1. Edwards curve $E_{a,d}$ defined over the prime field GF(p)
2. Point $Q=(x, y)$

Output: ACCEPT or REJECT Q as an affine point on $E_{a,d}$.

Process:

1. Verify that both x and y are integers in the interval $[0, p-1]$. Output REJECT if verification fails.
2. Verify that (x, y) is a point on the $E_{a,d}$ by checking that (x, y) satisfies the defining equation $a\, x^2 + y^2 = 1 + d\, x^2 y^2$ where computations are carried out in GF(p). Output REJECT if verification fails.
3. Otherwise output ACCEPT.

D.1.3.2 Full Public Key Validation

Inputs:

1. Edwards curve $E_{a,d}$ defined over the prime field GF(p)
2. Point Q

Output: ACCEPT or REJECT Q as a point on $E_{a,d}$ of order n.

Process:

1. Perform partial public key validation on Q using the procedure of Appendix D.1.3.1. Output REJECT if this procedure outputs REJECT.
2. If Q is the point at identity element (0,1), output REJECT.
3. Verify that $n\ Q = (0,1)$. Output REJECT if verification fails.
4. Otherwise output ACCEPT.

D.1.4 Binary Curves in Short-Weierstrass Form

D.1.4.1 Partial Public Key Validation

Inputs:

1. Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m)
2. Point $Q=(x, y)$

Output: ACCEPT or REJECT Q as an affine point on $B_{a,b}$.

Process:

1. If Q is the point at infinity ∅, output REJECT;
2. Verify that both x and y are binary polynomials in GF(2^m) according to the field representation indicated by the parameter *FR*. Output REJECT if verification fails.
3. Verify that (x, y) is a point on the $B_{a,b}$ by checking that (x, y) satisfies the defining equation $y^2 + x\,y = x^3 + a\,x^2 + b$, where computations are carried out in GF(2^m) according to the field representation indicated by the parameter *FR.* Output REJECT if verification fails.
4. Otherwise output ACCEPT.

D.1.4.2 Full Public Key Validation

Inputs:

1. Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m);
2. Point Q.

Output: ACCEPT or REJECT Q as a point on $B_{a,b}$ of order n.

Process:

1. Perform partial public key validation on Q using the procedure of Appendix D.1.4.1. Output REJECT if this procedure outputs REJECT.
2. Verify that $n\,Q = \varnothing$. Output REJECT if verification fails.
3. Otherwise output ACCEPT.

D.2 Point Compression

Point compression allows a shorter representation of elliptic curve points in affine coordinates by exploiting algebraic relationships between the coordinate values based on the defining equation of the curve in question. Point compression followed by its inverse, "point decompression," is the identity map.

D.2.1 Prime Curves in Short-Weierstrass Form

Point compression for non-binary curves in short-Weierstrass form is defined as follows.

Inputs:

1. Weierstrass curve $W_{a,b}$ defined over the prime field GF(p)
2. Point P on $W_{a,b}$

Output: Compressed point $\underline{P}$.

Process:

1. If P is the point at infinity $\varnothing$, set $\underline{P} = P$.
2. If $P = (x, y)$, set $\underline{P} = (x, \underline{y})$, where $\underline{y} = y \pmod 2$.
3. Output $\underline{P}$.

Point decompression of an object $\underline{P}$ with respect to this Weierstrass curve is defined as follows.

Inputs:

1. Object $\underline{P}$
2. Weierstrass curve $W_{a,b}$ defined over the prime field GF(p)

Output: Point P on $W_{a,b}$ or INVALID.

Process:

1. If $\underline{P}$ is the point at infinity $\varnothing$, output $P = \underline{P}$.
2. If $\underline{P}$ is the ordered pair (x, t), where x is an element of GF(p) and where t is an element of GF(2):
    2.1. Compute $w = x^3 + a\,x + b$
    2.2. Compute a square root y of w in GF(p) using the procedure of Appendix E.3; output INVALID if that procedure outputs INVALID
    2.3. If $y = 0$ and $t = 1$, output INVALID

2.4. If $t \neq y$ (mod 2), set $y = p-y$
2.5. Output $P = (x, y)$
3. Output INVALID

D.2.2 Binary Curves in Short-Weierstrass Form

Point compression for binary curves in short-Weierstrass form is defined as follows.

Inputs:

1. Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m)
2. Point P on $B_{a,b}$

Output: Compressed point $\underline{P}$.

Process:

1. If P is the point at infinity $\varnothing$, set $\underline{P} = P$.
2. If $P = (x, y)$ and x=0, set $\underline{P} = (x, \underline{y})$, where $\underline{y} = 0$ (mod 2).
3. If $P = (x, y)$ and $x \neq 0$:
   3.1. Compute $\alpha = y/x$, where $\alpha = \alpha_0 + \alpha_1 z + \cdots + \alpha_{m-1} z^{m-1}$
   3.2. Set $\underline{P} = (x, \underline{y})$, where $\underline{y} = \alpha_0$
4. Output $\underline{P}$.

Consequently, for each affine point $P = (x, y)$ on the Weierstrass curve $B_{a,b}$, the compressed point $\underline{P}$ is an ordered pair (x, t) where x is an element of GF(2^m) and where t is an element of GF(2).

Point decompression of an object $\underline{P}$ with respect to this Weierstrass curve is defined as follows.

Inputs:

1. Object $\underline{P}$
2. Weierstrass curve $B_{a,b}$ defined over the binary field GF(2^m), where m is an odd integer

Output: Point P on $B_{a,b}$ or INVALID.

Process:

1. If $\underline{P}$ is the point at infinity $\varnothing$, output $P = \underline{P}$.
2. If $\underline{P}$ is the ordered pair (x, t), where x is an element of GF(2^m) and where t is an element of GF(2), perform the following:
   2.1. If $x = 0$, perform the following steps:
      2.1.1. If $t = 1$, output INVALID
      2.1.2. Set y to the square root of b in GF(2^m) using the algorithm of Appendix E.1
   2.2. If $x \neq 0$, perform the following steps:
      2.2.1. Compute $w = (x^3 + a\,x^2 + b)/x^2 = x + a + b/x^2$

2.2.2. Compute a solution α in GF(2^m) of the equation $\alpha^2 + \alpha = w$ using the algorithm of Appendix E.2; output INVALID if that procedure outputs INVALID

2.2.3. If $t \neq \alpha_0$, where $\alpha = \alpha_0 + \alpha_1 z + \cdots + \alpha_{m-1} z^{m-1}$, set $\alpha = \alpha + 1$

2.2.4. Set $y = \alpha x$

2.3. Output $P = (x, y)$

3. Output INVALID.

D.3 Base Point (Generator) Selection

For user-generated base points, use a verifiably random method and check the validity of the point generated. This Appendix describes these methods.

D.3.1 Generation of Base Points

A base point **should** be generated as follows.

Input: Elliptic curve $E = (F_q, a, b)$, cofactor h, prime n, and, optionally, a bit string *Seed*, which indicates that verifiably random G is desired.

Output: A base point G on the curve of order n, or FAILURE.

Process: The following or its equivalent:

1. Set *base* = 1.
2. Select elements x and y in the field F_q, doing so verifiably at random using Appendix D.4.2 or by any desired method if *Seed* is not provided.

   Comment: The pair (x, y) **should** be chosen to lie on the curve E, or else the process could loop forever.

3. Let $G = hR$, where $R = (x, y)$.
4. If G is not a valid base point (see Appendix D.4.3), then increment *base* and go back to Step 1 unless *base* > $10h^2$, in which case, output FAILURE.

   Comment: The validity of G as a point is partially assured by R having valid coordinates and belonging to the curve. The verifiable random nature of G is also assured, so this does not need to be checked. Therefore, when validating G, it is only necessary to check that $G \neq O$ and $nG = O$.

If the elliptic curve E does not have a multiple of n points, then the output will generally be FAILURE. Conversely, if the algorithm outputs FAILURE, generally the elliptic curve does not have $h \cdot n$ points. If the elliptic curve E has exactly $h \cdot n$ points but n is composite, then G is not guaranteed to have order exactly n but will have an order dividing n. The probability that G has an order exactly n depends on the factorization of n. If the elliptic curve E has $k \cdot n$ points where $k \neq h$, then the order G is not guaranteed to have order n. If n is prime, then G will generally have an order which is a multiple of n. If the elliptic curve E has exactly $h \cdot n$ points, then *base* will generally never be incremented.

D.3.2 Verifiably Random Base Points

This procedure will generate a verifiably random candidate point.

Inputs: Bit string *Seed*, integer counter *base*, selected hash function with output length *hashlen* bits, field size q, cofactor h

Output: Candidate point (x, y)

Process: The following or its equivalent:

1. Set *element* = 1.
2. Convert *base* and *element* to octet strings *Base* and *Element*, respectively.
3. Compute $H = Hash$ ("Base point" || *Base* || *Element* || *Seed*).
4. Convert H to an integer e.
5. If $\lfloor e / 2q \rfloor = \lfloor 2^{hashlen} / 2q \rfloor$, then increment *element* and go to Step 2.
6. Let $t = e \bmod 2q$, so that t is an integer in the interval $[0, 2q – 1]$.
7. Let $x = t \bmod q$ and $z = \lfloor t / q \rfloor$.
8. Convert x to field element in F_q using the routine in Appendix F.2.
9. Recover the field element y from (x, z) using an appropriate compression method from Appendix D.2.
10. If the result is an error, then increment *element* and go to Step 2.

D.3.3 Validity of Base Points

A base point generator is valid if the following routine results in VALID.

Input: Elliptic curve domain parameters

Output: VALID or INVALID

Process: The following or its equivalent:

1. If $G = O$, then stop and output INVALID.
2. If either of the base point coordinates x_G and y_G are invalid as elements of F_q (that is: if q is odd, then either x_G or y_G is not an integer in the interval $[0, q–1]$; or if $q = 2^m$, then either x_G or y_G is not a bit string of length m), then stop and output INVALID.
3. If G is not on the elliptic curve, that is, $y_G^2 \neq x_G^3 + ax_G + b$ if q is odd, or $y_G^2 + x_Gy_G \neq x_G^3 + ax_G^2 + b$ if q is even, then stop and output INVALID.
4. If $nG \neq O$, then stop and output INVALID. A full scalar multiplication **shall** be used.

   Comment: Shortcuts for validating the order of point that assume a value for the cofactor would not be considered a full scalar multiplication.

5. If the input indicates that the base point G is generated verifiably at random, then do the following:
   5.1. Set *base* = 1.
   5.2. With *Seed* and *base* values, generate a point $R = (x, y)$, using the routine in Appendix D.4.2.

5.3. Compute $G' = hR$.

5.4. If $nG' \neq O$, then increment *base* and go back to Step 2.

Comment: The counter value *base* will generally never be incremented

5.5. If $base > 10h^2$, then stop and output INVALID.

5.6. Compare G' with G. If not equal, then stop and output INVALID.

6. Otherwise, output VALID.

Appendix E Auxiliary Functions

E.1 Computing Square Roots in Binary Fields

If x is an element of GF(2^m), then its square root is the element $x^{2^{m-1}}$.

E.2 Solving the Equation $x^2 + x = w$ in Binary Fields

Input: Field element w in GF(2^m), where m is an odd integer.

Output: Solution α in GF(2^m) of the equation $\alpha^2 + \alpha = w$, or INVALID.

Process:

1. Compute $\mathrm{Tr}(w) = w^{2^0} + w^{2^1} + w^{2^2} + w^{2^3} + \ldots + w^{2^{m-1}}$ (the trace of w);
2. If Tr(w)=1, output INVALID;
3. Compute $\alpha := \mathrm{Hf}(x) = w^{2^0} + w^{2^2} + w^{2^4} + \ldots + w^{2^{m-1}}$ (the half-trace of w);
4. Output α.

E.3 Computing Square Roots in non-Binary Fields GF(q)

The Tonelli-Shanks algorithm can be used to compute a square root given an equation of the form $x^2 \equiv n \pmod p$ where n is an integer, which is a quadratic residue (mod p), and p is an odd prime.

Find Q and S (with Q odd) such that $p - 1 = Q2^S$ by factoring out the powers of 2.

Note that if S = 1, as for primes p ≡ 3 (mod 4), this reduces to finding $x = n^{(p+1)/4} \pmod p$

Check to see if n^Q=1; if so then the root $x = n^{(Q+1)/2} \pmod p$.

Otherwise select a z which is a quadratic non-residue modulo p. The Legendre symbol $\left(\frac{a}{p}\right)$ where p is an odd prime and a is an integer can be used to test candidate values for z to see if a value of -1 is returned.

Search for a solution as follows:

Set $x = n^{(Q+1)/2} \pmod p$

Set t = $n^Q \pmod p$

Set $M = S$

Set $c = z^Q \pmod p$

While $t \neq 1$, repeat the following steps:

a) Using repeated squaring, find the smallest i such that $t^{2^i} = 1$, where $0 < i < M$. For example:

Let $e = 2$

Loop for $i = 1$ until $i = M$:

If $t^e \pmod p = 1$ then exit the loop.

Set $e = 2e$

b) Update values:

$b = c^{2^{M-i-1}} (\bmod\, p)$

$x = xb (\bmod\, p)$

$t = tb^2 (\bmod\, p)$

$c = b^2 (\bmod\, p)$

$M = i$

The solution is x and the second solution is $p - x$. If the least i found is M, then no solution exists.

Square roots in a non-binary field GF(q) are relatively efficient to compute if q has the special form $q \equiv 3 \pmod 4$ or $q \equiv 5 \pmod 8$. All but one of the elliptic curves recommended in this recommendation are defined over such fields. The following routines describe simplified cases to compute square roots for $p \equiv 3 \pmod 4$ or $p \equiv 5 \pmod 8$.

Let $u = y^2 - 1$ and $v = d\, y^2 + 1$.

To find a square root of (u/v) if $p \equiv 3 \pmod 4$ (as in E448), first compute the candidate root $x = (u/v)^{(p+1)/4} = u^3\, v\, (u^5 v^3)^{(p-3)/4}$ (mod p). If $v\, x^2 = u$, the square root is x. Otherwise, no square root exists, and the decoding fails.

To find a square root of (u/v) if $p \equiv 5 \pmod 8$ (as in Edwards25519), first compute the candidate root $x = (u/v)^{(p+3)/8} = u\, v^3\, (u\, v^7)^{(p-5)/8} \pmod p$. To find the root, check three cases:

- If $v\, x^2 = u \pmod p$, the square root is x.
- If $v\, x^2 = -u \pmod p$, the square root is $x * 2^{((p-1)/4)}$.
- Otherwise, no square root exists for modulo p, and decoding fails.

If $x = 0$ and $x_0 = 1$, point decoding fails. If $x \pmod 2 = x_0$, then the x-coordinate is x. Otherwise, the x-coordinate is $p - x$.

E.4 Computing Inverses in GF(q)

If x is an element of GF(q) and $x \neq 0$, its (multiplicative) inverse is the element x^{q-2}.

If one is concerned about side-channel leakage, one **should** compute u^{-1} indirectly by first computing the inverse of the blinded element λu, where λ is a random nonzero element of GF(q), and subsequently computing $\lambda(\lambda u)^{-1} = u^{-1}$. This yields an inversion routine where the inversion operation itself does not require side-channel protection and which may have relatively low computational complexity.

Appendix F Data Conversion

F.1 Conversion of a Field Element to an Integer

Field elements **shall** be converted to integers according to the following procedure.

Input: An element a of the field GF(q)

Output: A non-negative integer x in the interval $[0, q-1]$

Process:

1. If q is an odd prime, a is an integer in the interval $[0, q-1]$. In this case, set $x = a$.
2. If $q = 2^m$, a must be a binary polynomial of degree smaller than m, i.e.,
   $a = a(z) = a_{m-1}\, z^{m-1} + a_{m-2}\, z^{m-2} + \ldots + a_1\, z + a_0$, where each coefficient a_i is 0 or 1.
   In this case, set $x = a(2) = a_{m-1}\, 2^{m-1} + a_{m-2}\, 2^{m-2} + \ldots + a_1\, 2^1 + a_0\, 2^0$;
3. Output x.

F.2 Conversion of an Integer to a Field Element

Integers **shall** be converted to field elements according to the following procedure.

Inputs: Non-negative integer x and q, where q is an odd prime or $q=2^m$

Output: An element a of the field GF(q)

Process:

1. Set $x = x \pmod{q}$;
2. If q is an odd prime, x is an integer in the interval $[0, q-1]$. In this case, set $a = x$;
3. If $q = 2^m$, x can be uniquely written as $x = a_{m-1}\, 2^{m-1} + a_{m-2}\, 2^{m-2} + \ldots + a_1\, 2 + x_0$, where each coefficient x_i is 0 or 1. In this case, set $x = a(z) = a_{m-1}\, z^{m-1} + a_{m-2}\, z^{m-2} + \ldots + a_1\, z^1 + a_0\, 2^0$;
4. Output a.

F.3 Conversion of an Integer to a Bit String

Integers **shall** be converted to bit strings according to the following procedure.

Inputs: Non-negative integer x in the range $0 \le x < 2^l$

Output: Bit-string X of length l

Process:

1. The integer x can be uniquely written as $x = x_{l-1}\, 2^{l-1} + x_{l-2}\, 2^{l-2} + \ldots + x_1\, 2 + x_0$, where each coefficient x_i is 0 or 1.
2. Set X to the bit string $(x_{l-1}, x_{l-2}, \ldots, x_1, x_0)$;

3. Output X.

F.4 Conversion of a Bit String to an Integer

Bit strings **shall** be converted to integers according to the following procedure.

Input: Bit-string X of length l

Output: Non-negative integer x, where $x<2^l$

Process:

1. Let X be the bit string $(x_{l-1}, x_{l-2}, \ldots, x_1, x_0)$, where each coefficient x_i is 0 or 1;
2. Set x to the integer value $x = x_{l-1}\, 2^{l-1} + x_{l-2}\, 2^{l-2} + \ldots + x_1\, 2 + x_0$;
3. Output x.

Appendix G Implementation Aspects

G.1 Implementation of Modular Arithmetic

The prime moduli of the above recommended curves are of a special type (*generalized Mersenne numbers* and *Crandall primes*) for which modular multiplication can be carried out more efficiently than in general. This section provides the rules for implementing this faster arithmetic for each of these recommended prime moduli.

The usual way to multiply two integers (mod m) is to take the integer product and reduce it (modulo m). One, therefore, has the following problem: given an integer A less than m^2, compute

$$B = A \pmod{m}.$$

In general, one must obtain B as the remainder of an integer division. If m is a generalized Mersenne number, however, then B can be expressed as a sum or difference (mod m) of a small number of terms. To compute this expression, the integer sum or difference can be evaluated, and the result reduced modulo m. The latter reduction can be accomplished by adding or subtracting a few copies of m.

The prime modulus p for each of the four recommended P-x curves is a generalized Mersenne number.

G.1.1 Curve P-224

The modulus for this curve is $p = 2^{224} - 2^{96} + 1$. Each integer A less than p^2 can be written as

$$A = A_{13} \cdot 2^{416} + A_{12} \cdot 2^{384} + A_{11} \cdot 2^{352} + A_{10} \cdot 2^{320} + A_9 \cdot 2^{288} + A_8 \cdot 2^{256} + A_7 \cdot 2^{224} + A_6 \cdot 2^{192} + A_5 \cdot 2^{160} + A_4 \cdot 2^{128} + A_3 \cdot 2^{96} + A_2 \cdot 2^{64} + A_1 \cdot 2^{32} + A_0,$$

where each A_i is a 32-bit integer. As a concatenation of 32-bit words, this can be denoted by

$$A = (A_{13} \,\|\, A_{12} \,\|\, \ldots \,\|\, A_0).$$

The expression for B is

$$B = T + S_1 + S_2 - D_1 - D_2 \pmod{p},$$

where the 224-bit terms are given by

$$T = (A_6 \,\|\, A_5 \,\|\, A_4 \,\|\, A_3 \,\|\, A_2 \,\|\, A_1 \,\|\, A_0)$$

$$S_1 = (A_{10} \,\|\, A_9 \,\|\, A_8 \,\|\, A_7 \,\|\, 0 \,\|\, 0 \,\|\, 0)$$

$$S_2 = (0 \,\|\, A_{13} \,\|\, A_{12} \,\|\, A_{11} \,\|\, 0 \,\|\, 0 \,\|\, 0)$$

$$D_1 = (A_{13} \,\|\, A_{12} \,\|\, A_{11} \,\|\, A_{10} \,\|\, A_9 \,\|\, A_8 \,\|\, A_7)$$

$D_2 = \quad (0 \| 0 \| 0 \| 0 \| A_{13} \| A_{12} \| A_{11}).$

G.1.2 Curve P-256

The modulus for this curve is $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. Each integer A less than p^2 can be written as

$$A = A_{15} \cdot 2^{480} + A_{14} \cdot 2^{448} + A_{13} \cdot 2^{416} + A_{12} \cdot 2^{384} + A_{11} \cdot 2^{352} + A_{10} \cdot 2^{320} + A_9 \cdot 2^{288} + A_8 \cdot 2^{256} + A_7 \cdot 2^{224} + A_6 \cdot 2^{192} + A_5 \cdot 2^{160} + A_4 \cdot 2^{128} + A_3 \cdot 2^{96} + A_2 \cdot 2^{64} + A_1 \cdot 2^{32} + A_0,$$

where each A_i is a 32-bit integer. As a concatenation of 32-bit words, this can be denoted by

$$A = (A_{15} \| A_{14} \| \cdots \| A_0).$$

The expression for B is

$$B = T + 2S_1 + 2S_2 + S_3 + S_4 - D_1 - D_2 - D_3 - D_4 \pmod{p},$$

where the 256-bit terms are given by

$T = (A_7 \| A_6 \| A_5 \| A_4 \| A_3 \| A_2 \| A_1 \| A_0)$

$S_1 = (A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{11} \| 0 \| 0 \| 0)$

$S_2 = (0 \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| 0 \| 0 \| 0)$

$S_3 = (A_{15} \| A_{14} \| 0 \| 0 \| 0 \| A_{10} \| A_9 \| A_8)$

$S_4 = (A_8 \| A_{13} \| A_{15} \| A_{14} \| A_{13} \| A_{11} \| A_{10} \| A_9)$

$D_1 = (A_{10} \| A_8 \| 0 \| 0 \| 0 \| A_{13} \| A_{12} \| A_{11})$

$D_2 = (A_{11} \| A_9 \| 0 \| 0 \| A_{15} \| A_{14} \| A_{13} \| A_{12})$

$D_3 = (A_{12} \| 0 \| A_{10} \| A_9 \| A_8 \| A_{15} \| A_{14} \| A_{13})$

$D_4 = (A_{13} \| 0 \| A_{11} \| A_{10} \| A_9 \| 0 \| A_{15} \| A_{14})$

G.1.3 Curve P-384

The modulus for this curve is $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$. Each integer A less than p^2 can be written as

$$A = A_{23} \cdot 2^{736} + A_{22} \cdot 2^{704} + A_{21} \cdot 2^{672} + A_{20} \cdot 2^{640} + A_{19} \cdot 2^{608} + A_{18} \cdot 2^{576} + A_{17} \cdot 2^{544} + A_{16} \cdot 2^{512} + A_{15} \cdot 2^{480} + A_{14} \cdot 2^{448} + A_{13} \cdot 2^{416} + A_{12} \cdot 2^{384} + A_{11} \cdot 2^{352} + A_{10} \cdot 2^{320} + A_9 \cdot 2^{288} + A_8 \cdot 2^{256} + A_7 \cdot 2^{224} + A_6 \cdot 2^{192} + A_5 \cdot 2^{160} + A_4 \cdot 2^{128} + A_3 \cdot 2^{96} + A_2 \cdot 2^{64} + A_1 \cdot 2^{32} + A_0,$$

where each A_i is a 32-bit integer. As a concatenation of 32-bit words, this can be denoted by

$$A = (A_{23} \,\|\, A_{22} \,\|\, \cdots \,\|\, A_0).$$

The expression for B is

$$B = T + 2S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - D_1 - D_2 - D_3 \pmod p,$$

where the 384-bit terms are given by

$$T = (A_{11} \| A_{10} \| A_9 \| A_8 \| A_7 \| A_6 \| A_5 \| A_4 \| A_3 \| A_2 \| A_1 \| A_0)$$

$$S_1 = (0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| 0 \| 0 \| 0 \| 0)$$

$$S_2 = (A_{23} \| A_{22} \| A_{21} \| A_{20} \| A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12})$$

$$S_3 = (A_{20} \| A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{23} \| A_{22} \| A_{21})$$

$$S_4 = (A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{20} \| 0 \| A_{23} \| 0)$$

$$S_5 = (0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| A_{20} \| 0 \| 0 \| 0 \| 0)$$

$$S_6 = (0 \| 0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| 0 \| 0 \| A_{20})$$

$$D_1 = (A_{22} \| A_{21} \| A_{20} \| A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{23})$$

$$D_2 = (0 \| 0 \| 0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| A_{20} \| 0)$$

$$D_3 = (0 \| 0 \| 0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{23} \| 0 \| 0 \| 0).$$

G.1.4 Curve P-521

The modulus for this curve is $p = 2^{521} - 1$. Each integer A less than p^2 can be written as

$$A = A_1 \cdot 2^{521} + \mathrm{A}_0,$$

where each A_i is a 521-bit integer. As a concatenation of 521-bit words, this can be denoted by

$$A = (A_1 \,\|\, A_0).$$

The expression for B is

$$B = (A_0 + A_1) \pmod p.$$

G.1.5 Curve Curve448

The modulus for this curve is $p = 2^{448} - 2^{224} - 1$. Each integer A less than p^2 can be written

$$A = A_3 \cdot 2^{672} + A_2 \cdot 2^{448} + A_1 \cdot 2^{224} + \mathrm{A}_0,$$

where each A_i is a 224-bit integer. As a concatenation of 224-bit words, this can be denoted by

$$A = (A_3 \,\|\, A_2 \,\|\, A_1 \,\|\, A_0).$$

The expression for B is

$$B = (S_1 + S_2 + S_3 + S_4) \pmod p,$$

where the 448-bit terms are given by

$S_1 = \quad (A_1 \,\|\, A_0)$

$S_2 = \quad (A_2 \,\|\, A_2)$

$S_3 = \quad (A_3 \,\|\, A_3)$

$S_4 = \quad (A_3 \,\|\, 0)$.

G.1.6 Curve Curve25519

The modulus for this curve is $p = 2^{255} – 19$. Each integer A less than p^2 can be written

$$A = A_1 \cdot 2^{256} + A_0,$$

where each A_i is a 256-bit integer. As a concatenation of 256-bit words, this can be denoted by

$$A = (A_1 \,\|\, A_0).$$

The expression for B is

$$B = (38 \cdot A_1 + A_0) \pmod{2p},$$

where all computations are carried out modulo $2p$ rather than modulo p.

This allows efficient modular reduction and finite field operations that try and minimize carry-effects of operands if each integer X less than $2p$ is represented as

$X = X_9 \cdot 2^{234} + X_8 \cdot 2^{208} + X_7 \cdot 2^{182} + X_6 \cdot 2^{156} + X_5 \cdot 2^{130} + X_4 \cdot 2^{104} + X_3 \cdot 2^{78} + X_2 \cdot 2^{52} + X_1 \cdot 2^{26} + X_0$,

where each X_i is a 26-bit integer and where X_9 is a 22-bit integer. Note that in this case, multiplication by the small constant 38 does not lead to overflows if each X_i is stored as a 32-bit word. It turns out that the cost of occasional resizing of X, represented this way, is outweighed by savings due to the possibility of postponing 'carry' operations. This representation can also be used to efficiently compute $–X$ so that intermediate integer segments are always non-negative integers.

G.2 Scalar Multiplication for Koblitz Curves

This section describes a particularly efficient method of computing the scalar multiple $Q:=kP$ on the Koblitz curve $W_{a,b}$ over GF(2^m).

The operation τ is defined by

$$\tau(x, y) := (x^2, y^2).$$

When the normal basis representation is used, then the operation τ is implemented by performing right circular shifts on the bit strings representing x and y.

Given m and a, define the following parameters:

- C is some integer greater than 5.
- $\mu = (-1)^{1-a}$.
- For $i = 0$ and $i = 1$, define the sequence $s_i(m)$ by:

$$s_i(0) := 0, \qquad s_i(1) := 1 - i,$$

$$s_i(m) = \mu \cdot s_i(m-1) - 2 \cdot s_i(m-2) + (-1)^i.$$

- Define the sequence $V(m)$ by

$$V(0) := 2, \qquad V(1) := \mu,$$

$$V(m) = \mu \cdot V(m-1) - 2 \cdot V(m-2).$$

For the recommended Koblitz curves, the quantities $s_i(m)$ and $V(m)$ are as follows.

Curve K-163:

$s_0(163) =$ 2579386439110731650419537

$s_1(163) =$ –755360064476226375461594

$V(163) =$ –4845466632539410776804317

<u>Curve K-233:</u>

$s_0(233) =$ –27859711741434429761757834964435883

$s_1(233) =$ –44192136247082304936052160908934886

$V(233) =$ –137381546011108235394987299651366779

<u>Curve K-283:</u>

$s_0(283) =$ –665981532109049041108795536001591469280025

$s_1(283) =$ 1155860054909136775192281072591609913945968

$V(283) =$ 7777244870872830999287791970962823977569917

<u>Curve K-409:</u>

$s_0(409) =$ –18307510456002382137810317198756461378590542487556869338419259

$s_1(409) =$ –8893048526138304097196653241844212679626566100996606444816790

$V(409) =$ 10457288737315625927447685387048320737638796957687575791173829

<u>Curve K-571:</u>

$s_0(571) =$ –3737319446876463692429385892476115567147293964596131024123406420\
235241916729983261305

$s_1(571) =$ –3191857706446416099583814595948959674131968912148564658610565117\
58982848515832612248752

$V(571) =$ –1483809269816914138996191402970514903645425741804939362329123395\
34208516828973111459843

The following algorithm computes the scalar multiple $Q:=kP$ on the Koblitz curve $W_{a,b}$ over GF(2^m). The average number of elliptic additions and subtractions is at most ~ 1 + (m/3) and is at most ~ m/3 with probability at least $1 - 2^{5-C}$.

1. For i := 0 to 1 do

   1.1 $k' \leftarrow \lfloor k / 2^{a-C+(m-9)/2} \rfloor$.

   1.2 $g' \leftarrow s_i(m) \cdot k'$.

   1.3 $h' \leftarrow \lfloor g' / 2^m \rfloor$.

1.4 $j' \leftarrow V(m) \cdot h'$.

1.5 $l' \leftarrow \text{Round}((g' + j') / 2^{(m+5)/2})$.

1.6 $\lambda_i \leftarrow l' / 2^C$.

1.7 $f_i \leftarrow \text{Round}(\lambda_i)$.

1.8 $\eta_i \leftarrow \lambda_i - f_i$.

1.9 $h_i \leftarrow 0$.

2. $\eta \leftarrow 2\,\eta_0 + \mu\,\eta_1$.

3. If ($\eta \geq 1$),

then

if ($\eta_o - 3\,\mu\eta_1 < -1$)

then set $h_1 \leftarrow \mu$

else set $h_0 \leftarrow 1$.

else

if ($\eta_0 + 4\,\mu\,\eta_1 \geq 2$)

then set $h_1 \leftarrow \mu$.

4. If ($\eta < -1$)

then

if ($\eta_0 - 3\,\mu\,\eta_1 \geq 1$)

then set $h_1 \leftarrow -\mu$

else set $h_0 \leftarrow -1$.

else

if ($\eta_0 + 4\,\mu\,\eta_1 < -2$)

then set $h_1 \leftarrow -\mu$.

5. $q_0 \leftarrow f_0 + h_0$.

6. $q_1 \leftarrow f_1 + h_1$.

7. $r_0 \leftarrow n - (s_0 + \mu s_1) q_0 - 2s_1 q_1$.

8. $r_1 \leftarrow s_1 q_0 - s_0 q_1$.

9. Set $Q \leftarrow O$.

10. $P_0 \leftarrow P$.

11. While (($r_0 \neq 0$) or ($r_1 \neq 0$))

    11.1 If (r_0 odd), then

        11.1.1 set $u \leftarrow 2 - (r_0 - 2 r_1 \bmod 4)$.

        11.1.2 set $r_0 \leftarrow r_0 - u$.

        11.1.3 if ($u = 1$), then set $Q \leftarrow Q + P_0$.

        11.1.4 if ($u = -1$), then set $Q \leftarrow Q - P_0$.

    11.2 Set $P_0 \leftarrow \tau P_0$.

    11.3 Set $(r_0, r_1) \leftarrow (r_1 + \mu r_0 /2, - r_0 /2)$.

    Endwhile

12. Output Q.

G.3 Polynomial and Normal Bases for Binary Fields

G.3.1 Normal Bases

The elements of $GF(2^m)$, where m is odd, are expressed in terms of the type T normal[2] basis B for $GF(2^m)$, for some T. Each element has a unique representation as a bit string:

$$(a_0\, a_1\, \ldots\, a_{m-1}).$$

The arithmetic operations are performed as follows.

Addition: Addition of two elements is implemented by bit-wise addition modulo 2. Thus, for example,

$$(1100111) + (1010010) = (0110101).$$

[2] It is assumed in this section that m is odd and T is even since this is the only case considered in this standard.

Squaring: if

$$\alpha = (a_0\ a_1\ \ldots\ a_{m-2}\ a_{m-1}),$$

then

$$\alpha^2 = (a_{m-1}\ a_0\ a_1\ \ldots\ a_{m-2}).$$

Multiplication: Multiplication depends on the following function $F(\underline{u},\underline{v})$ on inputs

$$\underline{u} = (u_0\ u_1\ \ldots\ u_{m-1}) \quad \text{and} \quad \underline{v} = (v_0\ v_1\ \ldots\ v_{m-1}),$$

which is constructed s follows.

1. Set $p = Tm + 1$;
2. Let u be an integer having order T modulo p;
3. Compute the sequence $F(1), F(2), \ldots, F(p-1)$ as follows:
   a. Set $w = 1$;
   b. For j from 0 to $T-1$ do
      i. Set $n = w$;
      ii. For $i = 0$ to $m-1$ do
         1. Set $F(n) = i$;
         2. Set $n = 2n \pmod p$;
      1.2.3 Set $w = uw \pmod p$;
2. Output the formulae $F(u, v)$, where

$$F(u,v) := \sum_{k=1}^{p-2} u_{F(k+1)} v_{F(p-k)}.$$

This computation only needs to be performed once per basis.

Given the function F for B, the product

$$(c_0\ c_1\ \ldots\ c_{m-1}) = (a_0\ a_1\ \ldots\ a_{m-1}) * (b_0\ b_1\ \ldots\ b_{m-1})$$

is computed as follows:

1. Set $(u_0\ u_1\ \ldots\ u_{m-1}) = (a_0\ a_1\ \ldots\ a_{m-1})$;
2. Set $(v_0\ v_1\ \ldots\ v_{m-1}) = (b_0\ b_1\ \ldots\ b_{m-1})$;
3. For $k = 0$ to $m - 1$ do
   a. Compute $c_k = \mathrm{F}(\underline{u}, \underline{v})$.
   b. Set u = **LeftShift** (u) and v := **LeftShift** (v), where **LeftShift** denotes the circular left shift operation.
4. Output $c = (c_0\ c_1\ \ldots\ c_{m-1})$.

<u>Example:</u>
For the type-4 normal basis for $GF(2^7)$, one has $p = 29$ and $u = 12$ or $u = 17$. Thus, the values of F are given by:

$F(1) = 0$	$F(8) = 3$	$F(15) = 6$	$F(22) = 5$
$F(2) = 1$	$F(9) = 3$	$F(16) = 4$	$F(23) = 6$
$F(3) = 5$	$F(10) = 2$	$F(17) = 0$	$F(24) = 1$
$F(4) = 2$	$F(11) = 4$	$F(18) = 4$	$F(25) = 2$
$F(5) = 1$	$F(12) = 0$	$F(19) = 2$	$F(26) = 5$
$F(6) = 6$	$F(13) = 4$	$F(20) = 3$	$F(27) = 1$
$F(7) = 5$	$F(14) = 6$	$F(21) = 3$	$F(28) = 0$

Therefore,

$$F(\underline{u}, \underline{v}) = u_0\, v_1 + u_1\, (v_0 + v_2 + v_5 + v_6) + u_2\, (v_1 + v_3 + v_4 + v_5) + u_3\, (v_2 + v_5) +$$

$$u_4\, (v_2 + v_6) + u_5\, (v_1 + v_2 + v_3 + v_6) + u_6\, (v_1 + v_4 + v_5 + v_6).$$

As a result, if

$$a = (1\ 0\ 1\ 0\ 1\ 1\ 1) \text{ and } b = (1\ 1\ 0\ 0\ 0\ 0\ 1),$$

then

$$c_0 = F\,((1\ 0\ 1\ 0\ 1\ 1\ 1), (1\ 1\ 0\ 0\ 0\ 0\ 1)) = 1,$$
$$c_1 = F\,((0\ 1\ 0\ 1\ 1\ 1\ 1), (1\ 0\ 0\ 0\ 0\ 1\ 1)) = 0,$$
$$\vdots$$
$$c_6 = F\,((1\ 1\ 0\ 1\ 0\ 1\ 1), (1\ 1\ 1\ 0\ 0\ 0\ 0)) = 1,$$

so that $c = a*b = (1\ 0\ 1\ 1\ 0\ 0\ 1)$.

For the binary curves recommended in this specification, the values of T are, respectively, $T = 2$ ($m = 233$), $T = 6$ ($m = 283$), $T = 4$ ($m = 409$), and $T = 10$ ($m = 571$).

G.3.2 Polynomial Basis to Normal Basis Conversion

Let α be an element of the field GF(2^m) with bit-string representation p with respect to a given polynomial basis and bit-string representation n with respect to a given normal basis. The bit strings p and n are related via

$$p\,\Gamma = n,$$

where Γ is an ($m \times m$) matrix with entries in GF(2). The matrix Γ, which only depends on the bases, can be easily computed given its second-to-last row. For each conversion, that second-to-last row is given below.

<u>Degree 233:</u>

```
0x0be 19b89595 28bbc490 038f4bc4 da8bdfc1 ca36bb05 853fd0ed 0ae200ce
```

<u>Degree 283:</u>

```
0x3347f17 521fdabc 62ec1551 acf156fb 0bceb855 f174d4c1 7807511c 9f745382
 add53bc3
```

Degree 409:

```
0x0eb00f2 ea95fd6c 64024e7f 0b68b81f 5ff8a467 acc2b4c3 b9372843 6265c7ff
 a06d896c ae3a7e31 e295ec30 3eb9f769 de78bef5
```

Degree 571:

```
0x7940ffa ef996513 4d59dcbf e5bf239b e4fe4b41 05959c5d 4d942ffd 46ea35f3
 e3cdb0e1 04a2aa01 cef30a3a 49478011 196bfb43 c55091b6 1174d7c0 8d0cdd61
 3bf6748a bad972a4
```

If r is the second-to-last row of Γ and represents the element β of GF(2^m) with respect to the normal basis, then the rows of Γ, from top to bottom, are the bit-string representations of the elements

$$\beta^{m-1}, \beta^{m-2}, \ldots, \beta^2, \beta, 1$$

with respect to this normal basis. (Note that the element 1 is represented by the all-1 bit string.)

Alternatively, the matrix is the inverse of the matrix described in Appendix G.3.3.

More details of these computations can be found in Annex A.7 of the IEEE Standard 1363-2000 standard [IEEE 1363].

G.3.3 Normal Basis to Polynomial Basis Conversion

Let α be an element of the field GF(2^m) with bit-string representation n with respect to a given normal basis and bit-string representation p with respect to a given polynomial basis. The bit strings p and n are related via

$$n\,\Delta = p,$$

where Δ is an $(m \times m)$ matrix with entries in GF(2). The matrix Δ, which depends only on the bases, can be easily computed given its top row. For each conversion, that top row is given below.

Degree 233:

```
0x149 9e398ac5 d79e3685 59b35ca4 9bb7305d a6c0390b cf9e2300 253203c9
```

Degree 283:

```
0x31e0ed7 91c3282d c5624a72 0818049d 053e8c7a b8663792 bc1d792e ba9867fc
 7b317a99
```

Degree 409:

```
0x0dfa06b e206aa97 b7a41fff b9b0c55f 8f048062 fbe8381b 4248adf9 2912ccc8
 e3f91a24 e1cfb395 0532b988 971c2304 2e85708d
```

<u>Degree 571:</u>

```
0x452186b bf5840a0 bcf8c9f0 2a54efa0 4e813b43 c3d41496 06c4d27b 487bf107
 393c8907 f79d9778 beb35ee8 7467d328 8274caeb da6ce05a eb4ca5cf 3c3044bd
 4372232f 2c1a27c4
```

If r is the top row of Δ and represents the element β of $GF(2^m)$, then the rows of Δ, from top to bottom, are the bit strings representing the elements

$$\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}$$

with respect to the polynomial basis. Alternatively, the matrix is the inverse of the matrix described in Appendix G.3.2.

More details of these computations can be found in Annex A.7 of the IEEE Std 1363-2000 standard.

Appendix H – Other Allowed Elliptic Curves

H.1 Brainpool Curves

This standard also allows the curves specified in *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation* [RFC 5639], which support a security strength of 112 bits or higher. In particular, this includes brainpoolP224r1, brainpoolP256r1, brainpoolP320r1, brainpoolP384r1, and brainpoolP512r1. These curves were pseudorandomly generated and are allowed to be used for interoperability reasons.