Alternative Elliptic Curve Representations
draft-ietf-lwig-curve-representations-02

Abstract

This document specifies how to represent Montgomery curves and (twisted) Edwards curves as curves in short-Weierstrass form and illustrates how this can be used to carry out elliptic curve computations using existing implementations of, e.g., ECDSA and ECDH using NIST prime curves.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
Table of Contents

1. Fostering Code Reuse with New Elliptic Curves .......... 3
2. Specification of Wei25519 ................................ 4
3. Use of Representation Switches .......................... 4
4. Examples .................................................. 5
   4.1. Implementation of X25519 ........................... 5
   4.2. Implementation of Ed25519 .......................... 6
   4.3. Specification of ECDSA25519 ......................... 6
   4.4. Other Uses ........................................... 6
5. Caveats .................................................... 7
6. Security Considerations ................................... 8
7. Privacy Considerations ..................................... 8
8. IANA Considerations ....................................... 8
9. Acknowledgements .......................................... 9
10. References ................................................ 9
   10.1. Normative References ................................. 9
   10.2. Informative References .............................. 10
Appendix A. Some (non-Binary) Elliptic Curves ............... 10
   A.1. Curves in short-Weierstrass Form ................... 10
   A.2. Montgomery Curves ................................... 11
   A.3. Twisted Edwards Curves .............................. 11
Appendix B. Elliptic Curve Nomenclature and Finite Fields ... 11
   B.1. Elliptic Curve Nomenclature ........................ 11
   B.2. Finite Fields ......................................... 13
Appendix C. Elliptic Curve Group Operations ................... 14
   C.1. Group Law for Weierstrass Curves ................... 14
   C.2. Group Law for Montgomery Curves .................... 15
   C.3. Group Law for Twisted Edwards Curves ............... 15
Appendix D. Relationship Between Curve Models ................ 16
   D.1. Mapping between Twisted Edwards Curves and Montgomery Curves .................................. 16
   D.2. Mapping between Montgomery Curves and Weierstrass Curves .................................... 17
   D.3. Mapping between Twisted Edwards Curves and Weierstrass Curves .................................. 18
Appendix E. Curve25519 and Cousins .......................... 18
   E.1. Curve Definition and Alternative Representations .... 18
   E.2. Switching between Alternative Representations .... 19
   E.3. Domain Parameters ..................................... 20
Appendix F. Further Mappings ................................. 22
It is well-known that elliptic curves can be represented using different curve models. Recently, IETF standardized elliptic curves that are claimed to have better performance and improved robustness against "real world" attacks than curves represented in the traditional "short" Weierstrass model. This document specifies an alternative representation of points of Curve25519, a so-called Montgomery curve, and of points of Edwards25519, a so-called twisted Edwards curve.
Edwards curve, which are both specified in [RFC7748], as points of a specific so-called "short" Weierstrass curve, called Wei25519. We also define how to efficiently switch between these different representations.

Use of Wei25519 allows easy definition of new signature schemes and key agreement schemes already specified for traditional NIST prime curves, thereby allowing easy integration with existing specifications, such as NIST SP 800-56a [SP-800-56a], FIPS Pub 186-4 [FIPS-186-4], and ANSI X9.62-2005 [ANSI-X9.62], and fostering code reuse on platforms that already implement some of these schemes using elliptic curve arithmetic for curves in "short" Weierstrass form (see Appendix C.1).

2. Specification of Wei25519

For the specification of Wei25519 and its relationship to Curve25519 and Edwards25519, see Appendix E. For further details and background information on elliptic curves, we refer to the other appendices.

The use of Wei25519 allows reuse of existing generic code that implements short-Weierstrass curves, such as the NIST curve P-256, to also implement the CFRG curves Curve25519 and Edwards25519. We also cater to reusing of existing code where some domain parameters may have been hardcoded, thereby widening the scope of applicability. To this end, we specify the short-Weierstrass curves Wei25519.2 and Wei25519.-3, with hardcoded domain parameter a=2 and a=-3 (mod p), respectively; see Appendix G. (Here, p is the characteristic of the field over which these curves are defined.)

3. Use of Representation Switches

The curves Curve25519, Edwards25519, and Wei25519, as specified in Appendix E.3, are all isomorphic, with the transformations of Appendix E.2. These transformations map the specified base point of each of these curves to the specified base point of each of the other curves. Consequently, a public-key pair \((k, R := k \cdot G)\) for any one of these curves corresponds, via these isomorphic mappings, to the public-key pair \((k, R' := k \cdot G')\) for each of these other curves (where \(G\) and \(G'\) are the corresponding base points of these curves). This observation extends to the case where one also considers curve Wei25519.2 (which has hardcoded domain parameter \(a=2\), as specified in Appendix G.3, since it is isomorphic to Wei25519, with the transformation of Appendix G.2, and, thereby, also isomorphic to Curve25519 and Edwards25519.

The curve Wei25519.-3 (which has hardcoded domain parameter \(a=-3 \pmod p\)) is not isomorphic to the curve Wei25519, but is related in a
slightly weaker sense: the curve Wei25519 is isogenous to the curve Wei25519.-3, where the mapping of Appendix G.2 is an isogeny of degree $l=47$ that maps the specified base point $G$ of Wei25519 to the specified base point $G'$ of Wei25519.-3 and where the so-called dual isogeny (which maps Wei25519.-3 to Wei25519) has the same degree $l=47$, but does not map $G'$ to $G$, but to a fixed multiple thereof, where this multiple is $l=47$. Consequently, a public-key pair $(k, R:=k*G)$ for Wei25519 corresponds to the public-key pair $(k, R':=k*G')$ for Wei25519.-3 (via the $l$-isogeny), whereas the public-key pair $(k, R':=k*G')$ corresponds to the public-key pair $(l*k, l*R=l*k*G)$ of Wei25519 (via the dual isogeny). (Note the extra scalar $l=47$ here.)

Alternative curve representations can, therefore, be used in any cryptographic scheme that involves computations on public-private key pairs, where implementations may carry out computations on the corresponding object for the isomorphic or isogenous curve and convert the results back to the original curve (where, in case this involves an $l$-isogeny, one has to take into account the factor $l$). This includes use with elliptic-curve based signature schemes and key agreement and key transport schemes.

4. Examples

4.1. Implementation of X25519

RFC 7748 [RFC7748] specifies the use of X25519, a co-factor Diffie-Hellman key agreement scheme, with instantiation by the Montgomery curve Curve25519. This key agreement scheme was already specified in Section 6.1.2.2 of NIST SP 800-56a [SP-800-56a] for elliptic curves in short Weierstrass form. Hence, one can implement X25519 using existing NIST routines by (1) representing a point of the Montgomery curve Curve25519 as a point of the Weierstrass curve Wei25519; (2) instantiating the co-factor Diffie-Hellman key agreement scheme of the NIST specification with the resulting point and Wei25519 domain parameters; (3) representing the key resulting from this scheme (which is a point of the curve Wei25519 in Weierstrass form) as a point of the Montgomery curve Curve25519. The representation change can be implemented via a simple wrapper and involves a single modular addition (see Appendix D.2). Using this method has the additional advantage that one can reuse the public-private key pair routines, domain parameter validation, and other checks that are already part of the NIST specifications. Note: at this point, it is unclear whether this implies that a FIPS-accredited module implementing co-factor Diffie-Hellman for, e.g., P-256 would also extend this accreditation to X25519.
4.2. Implementation of Ed25519

RFC 8032 [RFC8032] specifies Ed25519, a "full" Schnorr signature scheme, with instantiation by the twisted Edwards curve Edwards25519. One can implement the computation of the ephemeral key pair for Ed25519 using an existing Montgomery curve implementation by (1) generating a public-private key pair \((k, R' := kG')\) for Curve25519; (2) representing this public-private key as the pair \((k, R := kG)\) for Ed25519. As before, the representation change can be implemented via a simple wrapper. Note that the Montgomery ladder specified in Section 5 of RFC7748 [RFC7748] does not provide sufficient information to reconstruct \(R' := (u, v)\) (since it does not compute the \(v\)-coordinate of \(R'\)). However, this deficiency can be remedied by using a slightly modified version of the Montgomery ladder that includes reconstruction of the \(v\)-coordinate of \(R' := kG'\) at the end of hereof (which uses the \(v\)-coordinate of the base point of Curve25519 as well). For details, see Appendix C.1.

4.3. Specification of ECDSA25519

FIPS Pub 186-4 [FIPS-186-4] specifies the signature scheme ECDSA and can be instantiated not just with the NIST prime curves, but also with other Weierstrass curves (that satisfy additional cryptographic criteria). In particular, one can instantiate this scheme with the Weierstrass curve Wei25519 and the hash function SHA-256, where an implementation may generate a public-private key pair for Wei25519 by (1) internally carrying out these computations on the Montgomery curve Curve25519, the twisted Edwards curve Edwards25519, or even the Weierstrass curve Wei25519.-3 (with hardcoded \(a = -3\) domain parameter); (2) representing the result as a key pair for the curve Wei25519.

Note that, in either case, one can implement these schemes with the same representation conventions as used with existing NIST specifications, including bit/byte-ordering, compression functions, and the-like. This allows generic implementations of ECDSA with the hash function SHA-256 and with the NIST curve P-256 or with the curve Wei25519 specified in this draft to use the same implementation (instantiated with, respectively, the NIST P-256 elliptic curve domain parameters or with the domain parameters of curve Wei25519 specified in Appendix E).

4.4. Other Uses

Any existing specification of cryptographic schemes using elliptic curves in Weierstrass form and that allows introduction of a new elliptic curve (here: Wei25519) is amenable to similar constructs, thus spawning "offspring" protocols, simply by instantiating these using the new curve in "short" Weierstrass form, thereby allowing code and/or specifications reuse and, for implementations that so...
desire, carrying out curve computations "under the hood" on Montgomery curve and twisted Edwards curve cousins hereof (where these exist). This would simply require definition of a new object identifier for any such envisioned "offspring" protocol. This could significantly simplify standardization of schemes and help keeping the resource and maintenance cost of implementations supporting algorithm agility [RFC7696] at bay.

5. Caveats

The examples above illustrate how specifying the Weierstrass curve Wei25519 (or any curve in short-Weierstrass format, for that matter) may facilitate reuse of existing code and may simplify standards development. However, the following caveats apply:

1. Wire format. The transformations between alternative curve representations can be implemented at negligible relative incremental cost if the curve points are represented as affine points. If a point is represented in compressed format, conversion usually requires a costly point decompression step. This is the case in [RFC7748], where the inputs to the co-factor Diffie-Hellman scheme X25519, as well as its output, are represented in u-coordinate-only format. This is also the case in [RFC8032], where the EdDSA signature includes the ephemeral signing key represented in compressed format (see Appendix I for details);

2. Representation conventions. While elliptic curve computations are carried-out in a field GF(q) and, thereby, involve large integer arithmetic, these integers are represented as bit- and byte-strings. Here, [RFC8032] uses least-significant-byte (LSB)/least-significant-bit (lsb) conventions, whereas [RFC7748] uses LSB/most-significant-bit (msb) conventions, and where most other cryptographic specifications, including NIST SP800-56a [SP-800-56a], FIPS Pub 186-4 [FIPS-186-4], and ANSI X9.62-2005 [ANSI-X9.62] use MSB/msb conventions. Since each pair of conventions is different (see Appendix J for details), this does necessitate bit/byte representation conversions;

3. Domain parameters. All traditional NIST curves are Weierstrass curves with domain parameter a=-3, while all Brainpool curves [RFC5639] are isomorphic to a Weierstrass curve of this form. Thus, one can expect there to be existing Weierstrass implementations with a hardcoded a=-3 domain parameter ("Jacobian-friendly"). For those implementations, including the curve Wei25519 as a potential vehicle for offering support for the CFRG curves Curve25519 and Edwards25519 is not possible, since not of the required form. Instead, one has to implement
Wei25519.-3 and include code that implements the isogeny and dual isogeny from and to Wei25519. This isogeny has degree \( l = 47 \) and requires roughly 9kB of storage for isogeny and dual-isogeny computations (see the tables in Appendix H). Note that storage would have reduced to a single 64-byte table if only the curve would have been generated so as to be isomorphic to a Weierstrass curve with hardcoded \( a = -3 \) parameter (this corresponds to \( l = 1 \)). Note: an example of such a curve is the Montgomery curve \( M_{A,B} \) over \( GF(p) \) with \( p = 2^{255}-19 \), \( B = 1 \), and \( A = -1410290 \) or (if one wants the base point to still have \( u \)-coordinate \( u = 9 \)) \( A = -3960846 \). In either case, the resulting curve has the same cryptographic properties as Curve25519 and the same performance (since \( A \) is a 3-byte integer as is the case with the domain parameter \( A = 486662 \) used with Curve25519), while being "Jacobian-friendly" by design.

6. Security Considerations

The different representations of elliptic curve points discussed in this document are all obtained using a publicly known transformation, which is either an isomorphism or a low-degree isogeny. It is well-known that an isomorphism maps elliptic curve points to equivalent mathematical objects and that the complexity of cryptographic problems (such as the discrete logarithm problem) of curves related via a low-degree isogeny are tightly related. Thus, the use of these techniques does not negatively impact cryptographic security.

As to implementation security, reusing existing high-quality code or generic implementations that have been carefully designed to withstand implementation attacks for one curve model may allow a more economical way of development and maintenance than providing this same functionality for each curve model separately (if multiple curve models need to be supported) and, otherwise, may allow a more gradual migration path, where one may initially use existing and accredited chipsets that cater to the pre-dominant curve model used in practice for over 15 years.

7. Privacy Considerations

The transformations between different curve models described in this document are publicly known and, therefore, do not affect privacy provisions.

8. IANA Considerations

An object identifier is requested for Wei25519 and ECDSA25519, using the representation conventions in this document.
9. Acknowledgements

Thanks to Nikolas Rosener for discussions surrounding implementation details of the techniques described in this document and to Phillip Hallam-Baker for triggering inclusion of verbiage on the use of Montgomery ladders with recovery of the y-coordinate. Thanks to Stanislav Smyshlyaev for his careful review.

10. References

10.1. Normative References


[SP-800-56a]
NIST SP 800-56a, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Log Cryptography, Revision 2", US Department of Commerce/National Institute of Standards and Technology, Gaithersburg, MD, June 2013.

10.2. Informative References


[tEd-Formulas]

Appendix A. Some (non-Binary) Elliptic Curves

A.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 defining equation \( Y^2 = X^3 + a*X + b \), where \( a \) and \( b \) are elements of GF(q) and where \( 4*a^3 + 27*b^2 \) is nonzero. 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 (the so-called affine points), together with the special point \( O \) (the so-called "point at infinity"). This set forms a group
under addition, via the so-called "secant-and-tangent" rule, where
the point at infinity serves as the identity element. See
Appendix C.1 for details of the group operation.

A.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^3 + A*u^2 + u, where A and B are elements of GF(q)
and where A is unequal to (+/-)2 and where B is nonzero. 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 (the so-
called affine points), together with the special point O (the so-
called "point at infinity"). This set forms a group under addition,
via the so-called "secant-and-tangent" rule, where the point at
infinity serves as the identity element. See Appendix C.2 for
details of the group operation.

A.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 distinct
nonzero elements of GF(q). 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 (the so-called affine points). It can be shown
that this set forms a group under addition if a is a square in GF(q),
whereas d is not, where the point O:=(0, 1) serves as the identity
element. (Note that the identity element satisfies the defining
equation.) See Appendix C.3 for details of the group operation.

An Edwards curve is a twisted Edwards curve with a=1.

Appendix B. Elliptic Curve Nomenclature and Finite Fields

B.1. Elliptic Curve Nomenclature

Each curve defined in Appendix A forms a commutative group under
addition (denoted by '+'). In Appendix C we specify the group laws,
which depend on the curve model in question. For completeness, we
here include some common elliptic curve nomenclature and basic
properties (primarily so as to keep this document self-contained).
These notions are mainly used in Appendix E and Appendix G and not
essential for our exposition. This section can be skipped at first
reading.

Any point P of a curve E is a generator of the cyclic subgroup
(P):={k*P \mid k = 0, 1, 2,...} of the curve. (Here, k*P denotes the
sum of k copies of P, where 0*P is the identity element O of the
curve.) If (P) has cardinality l, then l is called the order of P.
The order of curve E is the cardinality of the set of its points,
commonly denoted by |E|. A curve is cyclic if it is generated by some
point of this curve. All curves of prime order are cyclic, while all
curves of order h*n, where n is a large prime number and where h is a
small number (the so-called co-factor), have a large cyclic subgroup
of prime order n. In this case, a generator of order n is called a
base point, commonly denoted by G. A point of order dividing h is
said to be in the small subgroup. For curves of prime order, this
small subgroup is the singleton set, consisting of only the identity
element O. If a point is not in the small subgroup, it has order at
least n.

If R is a point of the curve that is also contained in (P), there is
a unique integer k in the interval [0, l-1] so that R=k*P, where l is
the order of P. 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
of the curve, if such a number exists.

If P is a fixed base point G of the curve, the pair (k, R:=k*G) is
called a public-private key pair, the integer k the private key, and
the point R the corresponding public key. The private key k can be
represented as an integer in the interval [0,n-1], where G has order
n.

In this document, a quadratic twist of a curve E defined over a field
GF(q) is a curve E’ related to E, with cardinality |E’|,
where |E|+|E’|=2*(q+1). If E is a curve in one of the curve models
specified in this document, a quadratic twist of this curve can be
expressed using the same curve model, although (naturally) with its
own curve parameters. Two curves E and E’ defined over a field GF(q)
are said to be isogenous if these have the same order and are said to
be isomorphic if these have the same group structure. Note that
isomorphic curves have necessarily the same order and are, thus, a
special type of isogenous curves. Further details are out of scope.

Weierstrass curves can have prime order, whereas Montgomery curves
and twisted Edwards curves always have an order that is a multiple
of four (and, thereby, a small subgroup of cardinality four).

An ordered pair (x, y) whose coordinates are elements of GF(q) can be
associated with any ordered triple of the form [x*z: y*z: z], where z
is a nonzero element of GF(q), and can be uniquely recovered from
such a representation. The latter representation is commonly called
a representation in projective coordinates.
The group laws in Appendix C are mostly expressed in terms of affine points, but can also be expressed in terms of the representation of these points in projective coordinates, thereby allowing clearing of denominators. The group laws may also involve non-affine points (such as the point at infinity O of a Weierstrass curve or of a Montgomery curve). Those can also be represented in projective coordinates. Further details are out of scope.

### B.2. Finite Fields

The field $\text{GF}(q)$, where $q$ is an odd prime power, is defined as follows.

If $p$ is a prime number, the field $\text{GF}(p)$ consists of the integers in the interval $[0,p-1]$ and two binary operations on this set: addition and multiplication modulo $p$.

If $q=p^m$ and $m>0$, the field $\text{GF}(q)$ is defined in terms of an irreducible polynomial $f(z)$ in $z$ of degree $m$ with coefficients in $\text{GF}(p)$ (i.e., $f(z)$ cannot be written as the product of two polynomials in $z$ of lower degree with coefficients in $\text{GF}(p)$): in this case, $\text{GF}(q)$ consists of the polynomials in $z$ of degree smaller than $m$ with coefficients in $\text{GF}(p)$ and two binary operations on this set: polynomial addition and polynomial multiplication modulo the irreducible polynomial $f(z)$. By definition, each element $x$ of $\text{GF}(q)$ is a polynomial in $z$ of degree smaller than $m$ and can, therefore, be uniquely represented as a vector $(x_{m-1}, x_{m-2}, \ldots, x_1, x_0)$ of length $m$ with coefficients in $\text{GF}(p)$, where $x_i$ is the coefficient of $z^i$ of polynomial $x$. Note that this representation depends on the irreducible polynomial $f(z)$ of the field $\text{GF}(p^m)$ in question (which is often fixed in practice). Note that $\text{GF}(q)$ contains the prime field $\text{GF}(p)$ as a subset. If $m=1$, we always pick $f(z):=z$, so that the definitions of $\text{GF}(p)$ and $\text{GF}(p^1)$ above coincide. If $m>1$, then $\text{GF}(q)$ is called a (nontrivial) extension field over $\text{GF}(p)$. The number $p$ is called the characteristic of $\text{GF}(q)$.

A field element $y$ is called a square in $\text{GF}(q)$ if it can be expressed as $y=x^2$ for some $x$ in $\text{GF}(q)$; it is called a non-square in $\text{GF}(q)$ otherwise. If $y$ is a square in $\text{GF}(q)$, we denote by $\sqrt{y}$ one of its square roots (the other one being $-\sqrt{y}$). For methods for computing square roots and inverses in $\text{GF}(q)$ — if these exist — see Appendix L.1 and Appendix L.2, respectively.

**NOTE:** The curves in Appendix E and Appendix G are all defined over a prime field $\text{GF}(p)$, thereby reducing all operations to simple modular integer arithmetic. Strictly speaking we could, therefore, have refrained from introducing extension fields. Nevertheless, we included the more general exposition, so as to accommodate potential
introduction of new curves that are defined over a (nontrivial) extension field at some point in the future. This includes curves proposed for post-quantum isogeny-based schemes, which are defined over a quadratic extension field (i.e., where \( q = p^2 \)), and elliptic curves used with pairing-based cryptography. The exposition in either case is almost the same and now automatically yields, e.g., data conversion routines for any finite field object (see Appendix J). Readers not interested in this, could simply view all fields as prime fields.

Appendix C. Elliptic Curve Group Operations

C.1. Group Law for Weierstrass Curves

For each point \( P \) of the Weierstrass curve \( W_{a,b} \), the point at infinity \( O \) serves as identity element, i.e., \( P + O = O + P = P \).

For each affine point \( P = (X, Y) \) of the Weierstrass curve \( W_{a,b} \), the point \(-P\) is the point \( (X, -Y) \) and one has \( P + (-P) = O \).

Let \( P_1 = (X_1, Y_1) \) and \( P_2 = (X_2, Y_2) \) be distinct affine points of the Weierstrass curve \( W_{a,b} \) and let \( Q = P_1 + P_2 \), where \( Q \) is not the identity element. Then \( Q = (x, y) \), where

\[
X + X_1 + X_2 = \lambda^2 \quad \text{and} \quad Y + Y_1 = \lambda (X_1 - X),
\]

where

\[
\lambda := \frac{Y_2 - Y_1}{(X_2 - X_1)}.
\]

Let \( P = (X_1, Y_1) \) be an affine point of the Weierstrass curve \( W_{a,b} \) and let \( Q = 2P \), where \( Q \) is not the identity element. Then \( Q = (x, y) \), where

\[
X + 2X_1 = \lambda^2 \quad \text{and} \quad Y + Y_1 = \lambda (X_1 - X),
\]

where

\[
\lambda := \frac{(3X_1^2 + a)}{(2Y_1)}.
\]

From the group laws above it follows that if \( P = (X, Y) \), \( P_1 = kP = (X_1, Y_1) \), and \( P_2 = (k+1)P = (X_2, Y_2) \) are distinct affine points of the Weierstrass curve \( W_{a,b} \) and if \( Y \) is nonzero, then the \( Y \)-coordinate of \( P_1 \) can be expressed in terms of the \( X \)-coordinates of \( P \), \( P_1 \), and \( P_2 \), and the \( Y \)-coordinate of \( P \), as

\[
Y_1 = \frac{(X \cdot X_1 + a) \cdot (X + X_1) + 2 \cdot b - X_2 \cdot (X - X_1)^2}{(2Y_1)}.
\]

This property allows recovery of the \( Y \)-coordinate of a point \( P_1 = kP \) that is computed via the so-called Montgomery ladder, where \( P \) is an affine point with nonzero \( Y \)-coordinate (i.e., it does not have order two). Further details are out of scope.
C.2. Group Law for Montgomery Curves

For each point $P$ of the Montgomery curve $M_{(A,B)}$, the point at infinity $O$ serves as identity element, i.e., $P + O = O + P = P$.

For each affine point $P:=(u, v)$ of the Montgomery curve $M_{(A,B)}$, the point $-P$ is the point $(u, -v)$ and one has $P + (-P) = O$.

Let $P1:=(u1, v1)$ and $P2:=(u2, v2)$ be distinct affine points of the Montgomery curve $M_{(A,B)}$ and let $Q:=P1 + P2$, where $Q$ is not the identity element. Then $Q:=(u, v)$, where

$$u + u1 + u2 = B*\lambda^2 - A$$

and

$$v + v1 = \lambda*(u1 - u),$$

where

$$\lambda:=(v2 - v1)/(u2 - u1).$$

Let $P:=(u1, v1)$ be an affine point of the Montgomery curve $M_{(A,B)}$ and let $Q:=2*P$, where $Q$ is not the identity element. Then $Q:=(u, v)$, where

$$u + 2*u1 = B*\lambda^2 - A$$

and

$$v + v1 = \lambda*(u1 - u),$$

where

$$\lambda:=(3*u1^2 + 2*A*u1+1)/(2*B*v1).$$

From the group laws above it follows that if $P=(u, v)$, $P1=k*P=(u1, v1)$, and $P2=(k+1)*P=(u2, v2)$ are distinct affine points of the Montgomery curve $M_{(A,B)}$ and if $v$ is nonzero, then the $v$-coordinate of $P1$ can be expressed in terms of the $u$-coordinates of $P$, $P1$, and $P2$, and the $v$-coordinate of $P$, as

$$v1=((u*u1+1)*(u+u1+2*A)-2*A-u2*(u-u1)^2)/(2*B*v).$$

This property allows recovery of the $v$-coordinate of a point $P1=k*P$ that is computed via the so-called Montgomery ladder, where $P$ is an affine point with nonzero $v$-coordinate (i.e., it does not have order one or two). Further details are out of scope.

C.3. Group Law for Twisted Edwards Curves

Note: The group laws below hold for twisted Edwards curves $E_{(a,d)}$ where $a$ is a square in GF(q), whereas $d$ is not. In this case, the addition formulae below are defined for each pair of points, without exceptions. Generalizations of this group law to other twisted Edwards curves are out of scope.

For each point $P$ of the twisted Edwards curve $E_{(a,d)}$, the point $O:=(0,1)$ serves as identity element, i.e., $P + O = O + P = P$. 
For each point \( P := (x, y) \) of the twisted Edwards curve \( E_{a,d} \), the point \(-P\) is the point \((-x, y)\) and one has \( P + (-P) = O \).

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

\[
x = (x_1y_2 + x_2y_1)/(1 + d*x_1*x_2*y_1*y_2) \quad \text{and} \quad y = (y_1y_2 - a*x_1*x_2)/(1 - d*x_1*x_2*y_1*y_2).
\]

Let \( P := (x_1, y_1) \) be a point of the twisted Edwards curve \( E_{a,d} \) and let \( Q := 2*P \). Then \( Q := (x, y) \), where

\[
x = (2*x_1*y_1)/(1 + d*x_1^2*y_1^2) \quad \text{and} \quad y = (y_1^2 - a*x_1^2)/(1 - d*x_1^2*y_1^2).
\]

Note that one can use the formulae for point addition for point doubling, taking inverses, and adding the identity element as well (i.e., the point addition formulae are uniform and complete (subject to our Note above)).

From the group laws above (subject to our Note above) it follows that if \( P := (x, y) \), \( P_1 = k*P := (x_1, y_1) \), and \( P_2 = (k+1)*P := (x_2, y_2) \) are affine points of the twisted Edwards curve \( E_{a,d} \) and if \( x \) is nonzero, then the \( x \)-coordinate of \( P_1 \) can be expressed in terms of the \( y \)-coordinates of \( P \), \( P_1 \), and \( P_2 \), and the \( x \)-coordinate of \( P \), as

\[
x_1 = (y*y_1 - y_2)/(x*(a-d*y*y_1*y_2)).
\]

This property allows recovery of the \( x \)-coordinate of a point \( P_1 = k*P \) that is computed via the so-called Montgomery ladder, where \( P \) is an affine point with nonzero \( x \)-coordinate (i.e., it does not have order one or two). Further details are out of scope.

Appendix D. Relationship Between Curve Models

The non-binary curves specified in Appendix A are expressed in different curve models, viz. as curves in short-Weierstrass form, as Montgomery curves, or as twisted Edwards curves. These curve models are related, as follows.

D.1. Mapping between Twisted Edwards Curves and Montgomery Curves

One can map points of the Montgomery curve \( M_{A,B} \) to points of the twisted Edwards curve \( E_{a,d} \), where \( a := (A+2)/B \) and \( d := (A-2)/B \) and, conversely, map points of the twisted Edwards curve \( E_{a,d} \) to points of the Montgomery curve \( M_{A,B} \), where \( A := 2(a+d)/(a-d) \) and where
B: \equiv 4/(a-d). For twisted Edwards curves we consider (i.e., those where a is a square in GF(q), whereas d is not), this defines a one-to-one correspondence, which - in fact - is an isomorphism between M\{A,B\} and E\{a,d\}, thereby showing that, e.g., the discrete logarithm problem in either curve model is equally hard.

For the Montgomery curves and twisted Edwards curves we consider, the mapping from M\{A,B\} to E\{a,d\} is defined by mapping the point at infinity O and the point (0, 0) of order two of M\{A,B\} to, respectively, the point (0, 1) and the point (0, -1) of order two of E\{a,d\}, while mapping each other point (u, v) of M\{A,B\} to the point (x, y) := (u/v, (u-1)/(u+1)) of 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 of E\{a,d\} to, respectively, the point at infinity O and the point (0, 0) of order two of M\{A,B\}, while each other point (x, y) of E\{a,d\} is mapped to the point (u, v) := ((1+y)/(1-y), (1+y)/(1-y)*x) of 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, thereby potentially allowing code reuse.

D.2. Mapping between Montgomery Curves and Weierstrass Curves

One can map points of the Montgomery curve M\{A,B\} to points of the Weierstrass curve W\{a,b\}, where a := (3-A^2)/(3*B^2) and b := (2*A^3-9*A)/(27*B^3). This defines a one-to-one correspondence, which - in fact - is an isomorphism between M\{A,B\} and W\{a,b\}, thereby showing that, e.g., the discrete logarithm problem in either curve model is equally hard.

The mapping from M\{A,B\} to W\{a,b\} is defined by mapping the point at infinity O of M\{A,B\} to the point at infinity O of W\{a,b\}, while mapping each other point (u, v) of M\{A,B\} to the point (X, Y) := (u/B+A/(3*B), v/B) of W\{a,b\}. Note that not all Weierstrass curves can be injectively 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, such as is the case with the so-called "NIST curves", this inverse mapping is not defined.

If the Weierstrass curve W\{a,b\} has a point (alpha, 0) of order two and c := a+3*(alpha)^2 is a square in GF(q), one can map points of this curve to points of the Montgomery curve M\{A,B\}, where A := 3*alpha/gamma and B := 1/gamma and where gamma is any square root of c. In this case, the mapping from W\{a,b\} to M\{A,B\} is defined by mapping
the point at infinity $O$ of $W_{a,b}$ to the point at infinity $O$ of $M_{A,B}$, while mapping each other point $(X,Y)$ of $W_{a,b}$ to the point $(u,v):=((X-\alpha)/\gamma,Y/\gamma)$ of $M_{A,B}$. As before, this defines a one-to-one correspondence, which — in fact — is an isomorphism between $W_{a,b}$ and $M_{A,B}$. It is easy to see that the mapping from $W_{a,b}$ to $M_{A,B}$ and that from $M_{A,B}$ to $W_{a,b}$ (if defined) are each other’s inverse.

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, thereby potentially allowing code reuse.

Note that implementations for elliptic curves with short-Weierstrass form that hard-code the domain parameter $a$ to $a=-3$ (which value is known to allow more efficient implementations) cannot always be used this way, since the curve $W_{a,b}$ resulting from an isomorphic mapping cannot always be expressed as a Weierstrass curve with $a=-3$ via a coordinate transformation. For more details, see Appendix F.

D.3. Mapping between Twisted Edwards Curves and Weierstrass Curves

One can map points of the twisted Edwards curve $E_{a,d}$ to points of the Weierstrass curve $W_{a,b}$, via function composition, where one uses the isomorphic mapping between twisted Edwards curve and Montgomery curves of Appendix D.1 and the one between Montgomery and Weierstrass curves of Appendix D.2. Obviously, one can use function composition (now using the respective inverses — if these exist) to realize the inverse of this mapping.

Appendix E. Curve25519 and Cousins

E.1. Curve Definition and Alternative Representations

The elliptic curve Curve25519 is the Montgomery curve $M_{A,B}$ defined over the prime field $GF(p)$, with $p:=2^{255}-19$, where $A:=486662$ and $B:=1$. 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. For this curve, the base point is the point $(G_\mu, G_\nu)$, where $G_\mu=9$ and where $G_\nu$ is an odd integer in the interval $[0, p-1]$.

This curve has the same group structure as (is "isomorphic" to) the twisted Edwards curve $E_{a,d}$ defined over $GF(p)$, with $a$ as base point the point $(G_\xi, G_\eta)$, where parameters are as specified in Appendix E.3. This curve is denoted as Edwards25519. For this
curve, the parameter $a$ is a square in $\text{GF}(p)$, whereas $d$ is not, so the group laws of Appendix C.3 apply.

The curve is also isomorphic to the elliptic curve $W_{\{a,b\}}$ in short-Weierstrass form defined over $\text{GF}(p)$, with as base point the point $(GX, GY)$, where parameters are as specified in Appendix E.3. This curve is denoted as Wei25519.

### E.2. Switching between Alternative Representations

Each affine point $(u, v)$ of Curve25519 corresponds to the point $(X, Y):=(u + A/3, v)$ of Wei25519, while the point at infinity of Curve25519 corresponds to the point at infinity of Wei25519. (Here, we used the mappings of Appendix D.2.) Under this mapping, the base point $(Gu, Gv)$ of Curve25519 corresponds to the base point $(GX, GY)$ of Wei25519. The inverse mapping maps the affine point $(X, Y)$ of Wei25519 to $(u, v):=(X - A/3, Y)$ of Curve25519, while mapping the point at infinity of Wei25519 to the point at infinity of Curve25519. Note that this mapping involves a simple shift of the first coordinate and can be implemented via integer-only arithmetic as a shift of $(p+A)/3$ for the isomorphic mapping and a shift of $-(p+A)/3$ for its inverse, where $\delta=(p+A)/3$ is the element of $\text{GF}(p)$ defined by

$$
\delta 192986815395526992372618308347813179754997444273427339909597
334652188435537
$$

$$
(=0x2aaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaaaa aaaaaaa)
$$

(Note that, depending on the implementation details of the field arithmetic, one may have to shift the result by $+p$ or $-p$ if this integer is not in the interval $[0,p-1]$.)

The curve Edwards25519 is isomorphic to the curve Curve25519, where the base point $(Gu, Gv)$ of Curve25519 corresponds to the base point $(Gx,Gy)$ of Edwards25519 and where 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 of Edwards25519 and where each other point $(u, v)$ of Curve25519 corresponds to the point $(c*u/v, (u-1)/(u+1))$ of Edwards25519, where $c$ is the element of $\text{GF}(p)$ defined by

$$
c \sqrt{-(A+2)/B}
$$

$$
51042569399160536130206135233146329284152202253034631822681833788
666877215207
$$
Here, we used the mapping of Appendix D.1 and normalized this using the mapping of Appendix F.1 (where the element s of that appendix is set to c above). The inverse mapping from Edwards25519 to Curve25519 is defined by mapping the point (0, 1) and the point (0, -1) of order two of Edwards25519 to, respectively, the point at infinity and the point (0,0) of order two of Curve25519 and having each other point (x, y) of Edwards25519 correspond to the point ((1 + y)/(1 - y), c*(1 + y)/((1-y)*x)) of Curve25519.

The curve Edwards25519 is isomorphic to the Weierstrass curve Wei25519, where the base point (Gx, Gy) of Edwards25519 corresponds to the base point (GX,GY) of Wei25519 and where the identity element (0,1) and the point (0,-1) of order two of Edwards25519 correspond to, respectively, the point at infinity O and the point (A/3, 0) of order two of Wei25519 and where each other point (x, y) of Edwards25519 corresponds to the point (X, Y):=((1+y)/(1-y)+A/3, c*(1+y)/((1-y)*x)) of Wei25519, where c was defined before. (Here, we used the mapping of Appendix D.3.) The inverse mapping from Wei25519 to Edwards25519 is defined by mapping the point at infinity O and the point (A/3, 0) of order two of Wei25519 to, respectively, the identity element (0,1) and the point (0,-1) of order two of Edwards25519 and having each other point (X, Y) of Wei25519 correspond to the point (c*(3*X-A)/(3*Y), (3*X-A-3)/(3*X-A+3)) of Edwards25519.

Note that these mappings can be easily realized if points are represented in projective coordinates, using a few field multiplications only, thus allowing switching between alternative curve representations with negligible relative incremental cost.

E.3. Domain Parameters

The parameters of the Montgomery curve and the corresponding isomorphic curves in twisted Edwards curve and short-Weierstrass form are as indicated below. Here, the domain parameters of the Montgomery curve Curve25519 and of the twisted Edwards curve Edwards25519 are as specified in [RFC7748]; the domain parameters of Wei25519 are "new".

General parameters (for all curve models):

\[ p \equiv 2^{255} - 19 \]

\[ (=0x7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff) \]
Montgomery curve-specific parameters (for Curve25519):

A 486662
B 1
Gu 9 (=0x9)
Gv 14781619447589544791020593568409986887264606134616475288964881837
755586237401

Twisted Edwards curve-specific parameters (for Edwards25519):

a -1 (-0x01)
d -121665/121666 = -(A-2)/(A+2)

Gx 15112221349535400772501151409588531511454012693041857206046113283
949847762202

Gy 4/5
Appendix F. Further Mappings

The non-binary curves specified in Appendix A are expressed in different curve models, viz. as curves in short-Weierstrass form, as Montgomery curves, or as twisted Edwards curves. In Appendix D we already described relationships between these various curve models. Further mappings exist between elliptic curves within the same curve model. These can be exploited to force some of the domain parameters to specific values that allow for a more efficient implementation of the addition formulae.

F.1. Isomorphic Mapping between Twisted Edwards Curves

One can map points of the twisted Edwards curve $E_{\{a,d\}}$ to points of the twisted Edwards curve $E_{\{a',d'\}}$, where $a := a' * s^2$ and $d := d' * s^2$ for some nonzero element $s$ of $GF(q)$. This defines a one-to-one
correspondence, which - in fact - is an isomorphism between $E_{a,d}$ and $E_{a',d'}$.

The mapping from $E_{a,d}$ to $E_{a',d'}$ is defined by mapping the point $(x,y)$ of $E_{a,d}$ to the point $(x', y') := (s*x, y)$ of $E_{a',d'}$. The inverse mapping from $E_{a',d'}$ to $E_{a,d}$ is defined by mapping the point $(x', y')$ of $E_{a',d'}$ to the point $(x, y) := (x'/s, y')$ of $E_{a,d}$.

Implementations may take advantage of this mapping to carry out elliptic curve group operations originally defined for a twisted Edwards curve with generic domain parameters $a$ and $d$ on a corresponding isomorphic twisted Edwards curve with domain parameters $a'$ and $d'$ that have a more special form, which are known to allow for more efficient implementations of addition laws. In particular, it is known that such efficiency improvements exist if $a' := -1$ (see [tEd-Formulas]).

F.2. Isomorphic Mapping between Montgomery Curves

One can map points of the Montgomery curve $M_{A,B}$ to points of the Montgomery curve $M_{A',B'}$, where $A := A'$ and $B := B' * s^2$ for some nonzero element $s$ of GF(q). This defines a one-to-one correspondence, which - in fact - is an isomorphism between $M_{A,B}$ and $M_{A',B'}$.

The mapping from $M_{A,B}$ to $M_{A',B'}$ is defined by mapping the point at infinity $O$ of $M_{A,B}$ to the point at infinity $O$ of $M_{A',B'}$, while mapping each other point $(u,v)$ of $M_{A,B}$ to the point $(u', v') := (u, s*v)$ of $M_{A',B'}$. The inverse mapping from $M_{A',B'}$ to $M_{A,B}$ is defined by mapping the point at infinity $O$ of $M_{A',B'}$ to the point at infinity $O$ of $M_{A,B}$, while mapping each other point $(u', v')$ of $M_{A',B'}$ to the point $(u,v) := (u', v'/s)$ of $M_{A,B}$.

One can also map points of the Montgomery curve $M_{A,B}$ to points of the Montgomery curve $M_{A',B'}$, where $A' := -A$ and $B' := -B$. This defines a one-to-one correspondence, which - in fact - is an isomorphism between $M_{A,B}$ and $M_{A',B'}$.

In this case, the mapping from $M_{A,B}$ to $M_{A',B'}$ is defined by mapping the point at infinity $O$ of $M_{A,B}$ to the point at infinity $O$ of $M_{A',B'}$, while mapping each other point $(u,v)$ of $M_{A,B}$ to the point $(u', v') := (-u, v)$ of $M_{A',B'}$. The inverse mapping from $M_{A',B'}$ to $M_{A,B}$ is defined by mapping the point at infinity $O$ of $M_{A',B'}$ to the point at infinity $O$ of $M_{A,B}$, while mapping each other point $(u', v')$ of $M_{A',B'}$ to the point $(u,v) := (-u', v')$ of $M_{A,B}$.
Implementations may take advantage of this mapping to carry out
elliptic curve group operations originally defined for a Montgomery
curve with generic domain parameters A and B on a corresponding
isomorphic Montgomery curve with domain parameters A' and B' that
have a more special form, which is known to allow for more efficient
implementations of addition laws. In particular, it is known that
such efficiency improvements exist if B' assumes a small absolute
value, such as B' := (+/-)1. (see [Ladder]).

F.3. Isomorphic Mapping between Weierstrass Curves

One can map points of the Weierstrass curve $W_{a,b}$ to points of the
Weierstrass curve $W_{a',b'}$, where $a':=a*s^4$ and $b':=b*s^6$ for some
nonzero element $s$ of GF(q). This defines a one-to-one
correspondence, which - in fact - is an isomorphism between $W_{a,b}$
and $W_{a',b'}$.

The mapping from $W_{a,b}$ to $W_{a',b'}$ is defined by mapping the point
at infinity $O$ of $W_{a,b}$ to the point at infinity $O$ of $W_{a',b'}$,
while mapping each other point $(X,Y)$ of $W_{a,b}$ to the point
$(X',Y') := (X*s^2, Y*s^3)$ of $W_{a',b'}$. The inverse mapping from
$W_{a',b'}$ to $W_{a,b}$ is defined by mapping the point at infinity $O$ of
$W_{a',b'}$ to the point at infinity $O$ of $W_{a,b}$, while mapping each
other point $(X',Y')$ of $W_{a',b'}$ to the point $(X,Y) := (X'/s^2,Y'/s^3)$
of $W_{a,b}$.

Implementations may take advantage of this mapping to carry out
elliptic curve group operations originally defined for a Weierstrass
curve with generic domain parameters $a$ and $b$ on a corresponding
isomorphic Weierstrass curve with domain parameter $a'$ and $b'$ that
have a more special form, which is known to allow for more efficient
implementations of addition laws, and translating the result back to
the original curve. In particular, it is known that such efficiency
improvements exist if $a'=-3$ (mod p), where p is the characteristic of
GF(q), and one uses so-called Jacobian coordinates with a particular
projective version of the addition laws of Appendix C.1. While not
all Weierstrass curves can be put into this form, all traditional
NIST curves have domain parameter $a=-3$, while all Brainpool curves
[RFC5639] are isomorphic to a Weierstrass curve of this form.

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 $W_{a,b}$ cannot always be expressed in
terms of a Weierstrass curve with $a'=-3$ via a coordinate
transformation: this only holds if $a'/a$ is a fourth power in GF(q)
(see Section 3.1.5 of [GECC]). However, even in this case, one can
still express the curve $W_{a,b}$ as a Weierstrass curve with a small
domain parameter value \( a' \), thereby still allowing a more efficient implementation than with a general domain parameter value \( a \).

F.4. Isogenous Mapping between Weierstrass Curves

One can still map points of the Weierstrass curve \( W_{a,b} \) to points of the Weierstrass curve \( W_{a',b'} \), where \( a' := -3 \pmod{p} \) and where \( p \) is the characteristic of \( GF(q) \), even if \( a'/a \) is not a fourth power in \( GF(q) \). In that case, this mapping cannot be an isomorphism (see Appendix F.3). Instead, the mapping is a so-called isogeny (or homomorphism). Since most elliptic curve operations process points of prime order or use so-called "co-factor multiplication", in practice the resulting mapping has similar properties as an isomorphism. In particular, one can still take advantage of this mapping to carry out elliptic curve group operations originally defined for a Weierstrass curve with domain parameter \( a \) unequal to \(-3 \pmod{p} \) on a corresponding isogenous Weierstrass curve with domain parameter \( a' = -3 \pmod{p} \) and translating the result back to the original curve.

In this case, the mapping from \( W_{a,b} \) to \( W_{a',b'} \) is defined by mapping the point at infinity \( O \) of \( W_{a,b} \) to the point at infinity \( O \) of \( W_{a',b'} \), while mapping each other point \((X,Y)\) of \( W_{a,b} \) to the point \((X',Y') := (u(X)/w(X)^2, Y*v(X)/w(X)^3) \) of \( W_{a',b'} \). Here, \( u(X), v(X), \) and \( w(X) \) are polynomials in \( X \) that depend on the isogeny in question. The inverse mapping from \( W_{a',b'} \) to \( W_{a,b} \) is again an isogeny and defined by mapping the point at infinity \( O \) of \( W_{a',b'} \) to the point at infinity \( O \) of \( W_{a,b} \), while mapping each other point \((X', Y')\) of \( W_{a',b'} \) to the point \((X,Y) := (u'(X')/w'(X')^2, Y'*v'(X')/w'(X')^3) \) of \( W_{a,b} \), where -- again -- \( u'(X'), v'(X'), \) and \( w'(X') \) are polynomials in \( X' \) that depend on the isogeny in question. These mappings have the property that their composition is not the identity mapping (as was the case with the isomorphic mappings discussed in Appendix F.3), but rather a fixed multiple hereof: if this multiple is \( l \) then the isogeny is called an isogeny of degree \( l \) (or \( l \)-isogeny) and \( u, v, \) and \( w \) (and, similarly, \( u', v', \) and \( w' \)) are polynomials of degrees \( l, 3*(l-1)/2, \) and \( (l-1)/2 \), respectively. Note that an isomorphism is simply an isogeny of degree \( l=1 \). Details of how to determine isogenies are outside scope of this document (for this, contact the author of this document).

Implementations may take advantage of this mapping to carry out elliptic curve group operations originally defined for a Weierstrass curve with a generic domain parameter \( a \) on a corresponding isogenous Weierstrass curve with domain parameter \( a' = -3 \pmod{p} \), where one can use so-called Jacobian coordinates with a particular projective version of the addition laws of Appendix C.1. Since all traditional
NIST curves have domain parameter $a=-3$, while all Brainpool curves [RFC5639] are isomorphic to a Weierstrass curve of this form, this allows taking advantage of existing implementations for these curves that may have a hardcoded $a=-3 \pmod p$ domain parameter, provided one switches back and forth to this curve form using the isogenous mapping in question.

Note that isogenous mappings can be easily realized using representations in projective coordinates and involves roughly $3*l$ finite field multiplications, thus allowing switching between alternative representations at relatively low incremental cost compared to that of elliptic curve scalar multiplications (provided the isogeny has low degree $l$). Note, however, that this does require storage of the polynomial coefficients of the isogeny and dual isogeny involved. This illustrates that low-degree isogenies are to be preferred, since an $l$-isogeny (usually) requires storing roughly $6*l$ elements of GF($q$). While there are many isogenies, we therefore only consider those with the desired property with lowest possible degree.

Appendix G. Further Cousins of Curve25519

G.1. Further Alternative Representations

The Weierstrass curve Wei25519 is isomorphic to the Weierstrass curve Wei25519.2 defined over GF($p$), with as base point the pair (G2X,G2Y), and isogenous to the Weierstrass curve Wei25519.-3 defined over GF($p$), with as base point the pair (G3X, G3Y), where parameters are as specified in Appendix G.3 and where the related mappings are as specified in Appendix G.2.

G.2. Further Switching

Each affine point $(X, Y)$ of Wei25519 corresponds to the point $(X', Y'):= (X*s^2, Y*s^3)$ of Wei25519.2, where $s$ is the element of GF($p$) defined by

$$s = 20343593038935618591794247374137143598394058341193943326473831977$$
$$39407761440$$

$$= 0x047f6814 6d568b44 7e4552ea a5ed633d 02d62964 a2b0a120$$
$$5e7941e9 375de020,$$

while the point at infinity of Wei25519 corresponds to the point at infinity of Wei25519.2. (Here, we used the mapping of Appendix F.3.) Under this mapping, the base point (GX, GY) of Wei25519 corresponds to the base point (G2X,G2Y) of Wei25519.2. The inverse mapping maps the affine point $(X', Y')$ of Wei25519.2 to $(X,Y):=(X'/s^2,Y'/s^3)$ of
Wei25519, while mapping the point at infinity O of Wei25519.2 to the point at infinity O of Wei25519. Note that this mapping (and its inverse) involves a modulo multipication of both coordinates with fixed constants s^2 and s^3 (respectively, 1/s^2 and 1/s^3), which can be precomputed.

Each affine point (X, Y) of Wei25519 corresponds to the point (X', Y'):=(X1*t^2,Y1*t^3) of Wei25519.-3, where (X1,Y1)=(u(X)/w(X)^2,Y*v(X)/w(X)^3), where u, v, and w are the polynomials with coefficients in GF(p) as defined in Appendix H.1 and where t is the element of GF(p) defined by

t   357281333982891756495869386056605426886916156991696662967154525084
      644181596229
    (=0x4efd6829 88ff8526 e189f712 5999550c e9ef729b ed1a7015
       73b1bab8 8bfc845),

while the point at infinity of Wei25519 corresponds to the point at infinity of Wei25519.-3. (Here, we used the isogenous mapping of Appendix F.4.) Under this isogenous mapping, the base point (GX,GY) of Wei25519 corresponds to the base point (G3X,G3Y) of Wei25519.-3. The dual isogeny maps the affine point (X', Y') of Wei25519.-3 to the affine point (X, Y):=(u'(X1)/w'(X1)^2,Y1*v'(x1)/w'(X1)^3) of Wei25519, where (X1,Y1)=(X'/t^2,Y'/t^3) and where u', v', and w' are the polynomials with coefficients in GF(p) as defined in Appendix H.2, while mapping the point at infinity O of Wei25519.-3 to the point at infinity O of Wei25519. Under this dual isogenous mapping, the base point (G3X, G3Y) of Wei25519.-3 corresponds to a multiple of the base point (GX, GY) of Wei25519, where this multiple is l=47 (the degree of the isogeny; see the description in Appendix F.3). Note that this isogenous map (and its dual) primarily involves the evaluation of three fixed polynomials involving the x-coordinate, which takes roughly 140 modular multiplications (or less than 5-10% relative incremental cost compared to the cost of an elliptic curve scalar multiplication).

G.3. Further Domain Parameters

The parameters of the Weierstrass curve with a=2 that is isomorphic with Wei25519 and the parameters of the Weierstrass curve with a=-3 that is isogenous with Wei25519 are as indicated below. Both domain parameter sets can be exploited directly to derive more efficient point addition formulae, should an implementation facilitate this.

General parameters: same as for Wei25519 (see Appendix E.3)
Weierstrass curve-specific parameters (for Wei25519.2, i.e., with a=2):

a  2 (=0x02)

b  12102640281269758552371076649779977768474709596484288167752775713
    178787220689
    (=0x1ac1da05 b55bc146 33bd39e4 7f94302e f19843dc f669916f
     6a5dfd01 65538cd1)

G2X 10770553138368400518417020196796161136792368198326337823149502681
     097436401658
    (=0x17cfeac3 78aed661 318e8634 582275b6 d9ad4def 072ea193
     5ee3c4e8 7a940ffa)

G2Y 54430575861508405653098668984457528616807103332502577521161439773
     88639873869
    (=0x0c08a952 c55dfad6 2c4f13f1 a8f68dca dc5c331d 297a37b6
     f0d7fdcc 51e16b4d)

Weierstrass curve-specific parameters (for Wei25519.-3, i.e., with a=-3):

a  -3
    (=0x7fffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
     ffffffffffffffff ffffffffffffffff)

b  29689592517550930188872794512874050362622433571298029721775200646
    451501277098
    (=0x41a3b6bf c668778e be2954a4 b1df36d1 485eceef1 ea614295
     796e1022 40891faa)

G3X 53837179229940872434942723257480777370451127212339198133697207846
     219400243292
    (=0x7706c37b 5a84128a 3884a5d7 1811f1b5 5da3230f fb17a8ab
     0b32e48d 31a6685c)

G3Y 69548073091100184414402055529279970392514867422855141773070804184
     60388229929
    (=0x0f60480c 7a5c0e11 40340adc 79d6a2bf 0cb57ad0 49d025dc
     38d80c77 985f0329)
Appendix H. Isogeny Details

The isogeny and dual isogeny are both isogenies with degree $l=47$. Both are specified by a triple of polynomials $u$, $v$, and $w$ (resp. $u'$, $v'$, and $w'$) of degree 47, 69, and 23, respectively, with coefficients in GF($p$). The coefficients of each of these polynomials are specified in Appendix H.1 (for the isogeny) and in Appendix H.2 (for the dual isogeny). For each polynomial in variable $x$, the coefficients are tabulated as sequence of coefficients of $x^0$, $x^1$, $x^2$, ..., in hexadecimal format.

H.1. Isogeny Parameters

H.1.1. Coefficients of $u(x)$

0 0x670ed14828b6f1791ceb3a9cc0edfe127dee8729c5a72ddf77bb1abaebbbba1e8
1 0x1135ca8bd5383cb3545402c8bce2ced14b45c29b241e4751b035f27524a9f932
2 0x3223806ff5f669c430efdf74df8389f058d180e2fcca5cdef3eacecdd2c34771
3 0x3188f3cf3f17a819c228517f6cd9814466c8c8eae2efcc47a29bfc1c364266
4 0x2541305c958c5a326f44efad2bec2b8e87abe840fadb08f2d994cd382fd8ce42
5 0x6e6f9c5792f3ff4f97f860f44a9c469cec42bd711526b733e10915be5b2bd8c6
6 0x3e9ad2e5f594b9ce6b06d4565891d28a1be8790000b396ef0bf59215d6cabfde
7 0x27848895d236403bb161347d19c913e7df5f372732a823ed807ee1d30266be
8 0x42f9d171ea8dc2f4a14ea46cc0ee54967175ecfe83a975137b753cb127c35060
9 0x128e40efa2d3ccb51567e73e9e19e731eac45700fa13ce5781cbe5ddd95648
10 0x450e5086c065430b496d8952dd2d5f2c5102bc27074d4d1e98bfa47413e045
11 0x487ef93ada70dfd44a4db8c41542e33d1aa32237bd3ca359b3ce1c59585f253d
12 0x33d2092700f26b1d2db96efb36cc2fa0a949be1307f49689022eab1892b01b875
13 0x4732b5996a20ebe4d5c5e2375d3b6c4b700c681bd9904343a14a055ef0ecd48
14 0x64d9e8272b9f5c6ad3470db54328386f42b18cb1c592cc6caf7893141b2107
15 0x52bbacdf1f85c61ef7ead8da27260fa2821f796167ed449b283036508ac5c5
16 0x320447ed91210985e2c401cfe1a93db1379424cf748f92fd61ab5cc356bc89a2
H.1.2. Coefficients of \( v(x) \)

0 0xf9f5eb7134e6f8daf30c45afa58d7bfc6d4e3ccbb5de87b562fd77403972b2
1 0x36c2dcd9e88f0d2d517a15fc453a098bbb5a50eb6e8da906fae184ae1a13f7
2 0xb40078302c24fa394a834880d5bf46732ca1b4894172fb7f775821276f558b3
3 0x53dd8e2234573f7f3f7df11e90a7bdd7b75d807f9712f521d4f18af59a5f26
4 0x6d4d7bb08de9061988a8c6ff3beb10e933d4d2fbb8872d256a38c74c8c2ceda
5 0x71bfe5831b30e28cd0fbe1e9916ab2291c6beacc5a0f8e2c9165c632e61dd2f5
6 0x7c524f4d17ff2ee88463da012fc12a5b677fb5b0ab59f4bbf162d76be1c89c
7 0x758183d5e07878d3364e3fd4c863a5dc1fe723f48c4ab4273fc034f5454d59a4
8 0x1eb41ef2479444ecdcccbb200f64bde53f434a02b6c3f485d32f14da6a7700e1
9 0x1490f3851f016cc3cf8a1e3c16a53317253d232ed425297531b560d70770315c
10 0x9bc43131964e46d905c3899c9d465c3abbd26eab9371c0e429b36d4b86469c
11 0x5f27c173d94c82d83483f08c88daa0bcf5af8f436a47262050f240e9be3b
12 0x1d20010ec741aa393cd19f0133b35f067adab0d105babe75fe45c8ba732ceb
13 0x1b3c669ae498b86be2f0c946a9ff6c48e44740d7d9804146915747c30c25996a
14 0x24c6090f79ec13e3ae454d8f0f9e0c30a893810595f79602f2ba013b3c10db
15 0x4650c5b5648c6c43ac75a2042048c699e44437929268661726e7182a31b1532f
H.1.3. Coefficients of \( w(x) \)

\[
\begin{array}{ll}
0 & 0x3da24d42421264f30939ff00203880f2b017eb3feccf8933ae61e18df8c8ba116 \\
1 & 0x457f20bc393cdc9a66848ce174e2fa41d77e6dbae05a317a1f6e3ae87860f8 \\
2 & 0x7f608a2285c480d5c9592c435431fae94695beef79d770bb6d029c1d10a53295 \\
3 & 0x3832acc520a485100a0a1695792465142a5572bed1b2e50e1f8f662ac7289bb \\
4 & 0x2dfb10559e31b328eb34beedd5e537c3f4d7b9bebfb0749f75d6d068662fbaa \\
5 & 0x25396820381d04159f655dd41c74303ded05d54a775e2f5800659adda28 \\
6 & 0x6fa070a70ca2bc6d4d0795fb28d4990b2cc80cd72d48b603a8ac8c8268bef6a6 \\
7 & 0x27f488578357388b20fbc7503328e1d10de02b082b3c7b8ceb33c29fea7a0d2 \\
8 & 0x15776851a7cabcfe84c63211806915c0c15c75068a47021968c743846076e6 \\
9 & 0x101565b08a9af015c172f19b48a4df254c4fbd85f27d153edc9131d45e8f \\
10 & 0x196b0fffb9f3229feaadac0d7459b905caab6b83f905ee813ee8449f8a62c \\
11 & 0x1f55784691719f765f04ae9051ec95d5deb42ae45405a9d7833855a695a94 \\
12 & 0x628858f79cca86305739d084d365d5a9e56e51a4485d253ae3f2e4379fa8aff \\
13 & 0x4a842dcd934a80d1e61dab3622a8c4d390da1592de56d1c14c4d3f72dd01a5 \\
14 & 0xf3bfc9c1b7a1125f94766a4097d0f1018963bc11cb7bc07a1d94d5e282477 \\
15 & 0x1c4bd70488c4882846500691fa7543b7ef6944469c3e3b4707ea2c99383e53c \\
16 & 0x2d7017e47b24b89b0528932c4ade43f09091b91db0072e6ebdc5e777cb215e35
\end{array}
\]
H.2. Dual Isogeny Parameters

H.2.1. Coefficients of u'(x)

0 0xf0edd5b84a20aaac8f1419efdd02a5cca77b21e4cf2a78c49b5127d98bc5882
1 0x7115e60d44a58630417df33dd45b8a546fa00b79f0a3b2bdcf49694bade87c0a
2 0xb3f3a6f3c445c7dc1f9112127514e88c32ff3f367ba0edad4d75b7e7b94b56
3 0x1eb31bb333d7048b87f2b3d4ec76d69035927b41c30274368649c87c5e1ab30
4 0x552c886c2044153e280832264066ccee2a7da1127dc9720e2a380e9d37049ac64
5 0x4504f27098db2ef5840b74e42445298755d9493141f5417c02f04d47797dda
6 0x82c242cceb19698a4fa305aafef64e051c04ae8b52cb68d89ee8522e628
7 0x480473406add76cf1d77661b3ff506c038d9cdd5ad6e1ea41969430bb767d23
8 0x25f47bb506fba80c79d1763365fa9076d4c4cb6644f73ed37918074397e88588
9 0x10f13ed36eab593fa20817f6bb70cac292e18d300498f6642e35cbdf772f0855
10 0x7d28329d695f3b305620f83a58df1531e89a43c7b3151d16f3b60a8246c36ade
11 0x2c5ec8c42b16dc6049bb2c7b4ffe9d65d7209e886badbd5f865dec35e4ab4a
12 0x7f4f33cd50255537e6cde15a4a327a5790c37e081802654b56c95643454e133
13 0x7d30431a121d9240c761998cf83d228237e80c3ef5c7191ec9617208e0ab8c0c
14 0x4d2a7d6609610c1deed56425a4615b92f70a507e1079b2681d96a2b874cf0630
H.2.2. Coefficients of $v'(x)$

0 0x43c9b67cc5b16e167b55f190db61e44d48d813a7112910f10e3f3d8da85d61d3
1 0x72046db07e0e7882ff3f0f38b54b45ca84153be47a7fd1dd8f6402e17c47966f
2 0x1593d97b65a070b6b3f879fe3dc4d1ef03c0e781c997111d5c1748f956f1ff0c
3 0x54e5fec076b8779338432bdc5a449e36823a0a7c905fd37f232330b026a143a0
4 0x46328dd9bc336e0873abd453db472468393333f8bf2010c6ac283933216e98038
5 0x25d0c64de1dfe1c6d5ff5f2d98ab637d8b39bfcf0d88e8d7eb03ce
6 0x3a175c46b2cd8e2b313ddee2d5f3097b78114a6295f283cf58a33844b088b3b34
7 0x5cf4e6f745bddd61181a71b4db31dc4c30c84957f63df163bee5e4667a8d38
8 0x630971c39b723ea51cfd870478331d60396b31f39a593ebd9b1eb543875283
9 0x7ea8f895dcd85fc6c2b58793789bd9246e62fa7a8c7116936876f4d8df869b
10 0x503818acb535bcaacaf8ad44a83c213a9e83af7c937dc9b3e5b6efedc0a7428c
11 0xe815373920ec3cbf3f8cae20d4389d367dc4398e0169124af90edc3e6d42b8
12 0x7e4b23e1e0b739087f77910cc635a92a3dc184a791400cbceae056c19c853815
13 0x45322201db4b5e0a643229e07c0ab7c36e4274745689be2c19cfa8a702129d
H.2.3. Coefficients of w'(x)

0 0x6bd7f1fc5dd51b7d832848c180f019bcbdb101d4b3435230a79cc4f95c35e15e
1 0x17413bb3ee505184a504e14419b8d7c8517a0d268f65b0d7f5b0ba68d6166dd0
2 0x47f4471beed06e5e2b6d5569c20e30346bdba2921d9676603c58e55431572f90
3 0x2af7eaafdf04f6910a5b01cde0c77dca09487f1cd1116b38db34563e7b0b41eb
4 0x57f0a593459732eeff1d2e2f7085bf9adf5348479ba56f7afde17c4a0d3d3477b
5 0x4da04e912f145c8d1e5957e0a9e44cca83e74345b38583b70840bdfdbd0288ed
6 0x7cc9c3a51a3767d9d37c6652c349adc09bfe477d99f249a2a7bc803c1cf39ed
7 0x425d7e58b8adf87eefb445b424ba308ee7880228921651959a7eab548180ad49
8 0x48156db5c99248234c09f43fedf509005943d35f5d7422621617467b06d314f
9 0xd837dbbd1af32d04e2699cb026399c1928472aa17f0a1d3af2d4bc9923456a
10 0x5b8006e0f924e67c1f207464a9d025758c078b43ddc0e9a9e9993641e5650be
11 0x29c91284e5d14939a69bc848908bd9f18346c259b9bd40f3ed65182f3a2f39
12 0x25550b0f3bceef18a6bf4a46c45bf1b92f22a76da466bdf19d07398c80b0f946
13 0x495d289b1db16229d7d4630cb65d52500256547401f121a9b09fb8e82cf01953
14 0x718c8610ea7048a370eabfd9888c633ee31dd70f8bcc58361962bb08619963e
Appendix I. Point Compression

Point compression allows a shorter representation of affine points of an elliptic curve by exploiting algebraic relationships between the coordinate values based on the defining equation of the curve in question. Point decompression refers to the reverse process, where one tries and recover the affine point from its compressed representation and information on the domain parameters of the curve. Consequently, point compression followed by point decompression is the identity map.

The description below makes use of an auxiliary function (the parity function), which we first define for prime fields GF(p) and then extend to all fields GF(q), where q is an odd prime power. We assume each finite field to be unambiguously defined.

Let y be a nonzero element of GF(q). If q:=p is an odd prime number, y and p-y can be uniquely represented as integers in the interval [1,p-1] and have odd sum p. Consequently, one can distinguish y from -y via the parity of this representation, i.e., via par(y):=y (mod 2). If q:=p^m, where p is an odd prime number and where m>0, both y and -y can be uniquely represented as vectors of length m, with coefficients in GF(p) (see Appendix B.2). In this case, the leftmost nonzero coordinate values of y and -y are in the same position and have representations in [1,p-1] with different parity. As a result, one can distinguish y from -y via the parity of the representation of this coordinate value. This extends the definition of the parity function to any odd-size field GF(q), where one defines par(0):=0.
I.1. Point Compression for Weierstrass Curves

If \( P := (X, Y) \) is an affine point of the Weierstrass curve \( W_{a,b} \) defined over the field \( \text{GF}(q) \), then so is \(-P := (X, -Y)\). Since the defining equation \( Y^2 = X^3 + aX + b \) has at most two solutions with fixed \( X \)-value, one can represent \( P \) by its \( X \)-coordinate and one bit of information that allows one to distinguish \( P \) from \(-P\), i.e., one can represent \( P \) as the ordered pair \( \text{compr}(P) := (X, \text{par}(Y)) \). If \( P \) is a point of order two, one can uniquely represent \( P \) by its \( X \)-coordinate alone, since \( Y = 0 \) and has fixed parity. Conversely, given the ordered pair \((X, t)\), where \( X \) is an element of \( \text{GF}(q) \) and where \( t = 0 \) or \( t = 1 \), and the domain parameters of the curve, one can use the defining equation of the curve to try and determine candidate values for the \( Y \)-coordinate given \( X \), by solving the quadratic equation \( Y^2 = \alpha \), where \( \alpha := X^3 + aX + b \). If \( \alpha \) is not a square in \( \text{GF}(q) \), this equation does not have a solution in \( \text{GF}(q) \) and the ordered pair \((X, t)\) does not correspond to a point of this curve. Otherwise, there are two solutions, viz. \( Y = \sqrt{\alpha} \) and \(-Y\). If \( \alpha \) is a nonzero element of \( \text{GF}(q) \), one can uniquely recover the \( Y \)-coordinate for which \( \text{par}(Y) := t \) and, thereby, the point \( P := (X, Y) \). This is also the case if \( \alpha = 0 \) and \( t = 0 \), in which case \( Y = 0 \) and the point \( P \) has order two. However, if \( \alpha = 0 \) and \( t = 1 \), the ordered pair \((X, t)\) does not correspond to the outcome of the point compression function.

I.2. Point Compression for Montgomery Curves

If \( P := (u, v) \) is an affine point of the Montgomery curve \( M_{A,B} \) defined over the field \( \text{GF}(q) \), then so is \(-P := (u, -v)\). Since the defining equation \( Bv^2 = u^3 + Au^2 + u \) has at most two solutions with fixed \( u \)-value, one can represent \( P \) by its \( u \)-coordinate and one bit of information that allows one to distinguish \( P \) from \(-P\), i.e., one can represent \( P \) as the ordered pair \( \text{compr}(P) := (u, \text{par}(v)) \). If \( P \) is a point of order two, one can uniquely represent \( P \) by its \( u \)-coordinate alone, since \( v = 0 \) and has fixed parity. Conversely, given the ordered pair \((u, t)\), where \( u \) is an element of \( \text{GF}(q) \) and where \( t = 0 \) or \( t = 1 \), and the domain parameters of the curve, one can use the defining equation of the curve to try and determine candidate values for the \( v \)-coordinate given \( u \), by solving the quadratic equation \( v^2 = \alpha \), where \( \alpha := (u^3 + Au^2 + u)/B \). If \( \alpha \) is not a square in \( \text{GF}(q) \), this equation does not have a solution in \( \text{GF}(q) \) and the ordered pair \((u, t)\) does not correspond to a point of this curve. Otherwise, there are two solutions, viz. \( v = \sqrt{\alpha} \) and \(-v\). If \( \alpha \) is a nonzero element of \( \text{GF}(q) \), one can uniquely recover the \( v \)-coordinate for which \( \text{par}(v) := t \) and, thereby, the affine point \( P := (u, v) \). This is also the case if \( \alpha = 0 \) and \( t = 0 \), in which case \( v = 0 \) and the point \( P \) has order two. However, if \( \alpha = 0 \) and \( t = 1 \), the ordered pair \((u, t)\) does not correspond to the outcome of the point compression function.
I.3. Point Compression for Twisted Edwards Curves

If \( P := (x, y) \) is an affine point of the twisted Edwards curve \( E_{\{a,d\}} \) defined over the field \( GF(q) \), then so is \( -P := (-x, y) \). Since the defining equation \( a x^2 + y^2 = 1 + d x^2 y^2 \) has at most two solutions with fixed \( y \)-value, one can represent \( P \) by its \( y \)-coordinate and one bit of information that allows one to distinguish \( P \) from \( -P \), i.e., one can represent \( P \) as the ordered pair \( \text{compr}(P) := (\text{par}(x), y) \). If \( P \) is a point of order one or two, one can uniquely represent \( P \) by its \( y \)-coordinate alone, since \( x = 0 \) and has fixed parity. Conversely, given the ordered pair \((t, y)\), where \( y \) is an element of \( GF(q) \) and where \( t = 0 \) or \( t = 1 \), and the domain parameters of the curve, one can use the defining equation of the curve to try and determine candidate values for the \( x \)-coordinate given \( y \), by solving the quadratic equation \( x^2 = \alpha \), where \( \alpha = (1 - y^2) / (a - d y^2) \). If \( \alpha \) is not a square in \( GF(q) \), this equation does not have a solution in \( GF(q) \) and the ordered pair \((t, y)\) does not correspond to a point of this curve. Otherwise, there are two solutions, viz. \( x = \sqrt{\alpha} \) and \(-x\). If \( \alpha \) is a nonzero element of \( GF(q) \), one can uniquely recover the \( x \)-coordinate for which \( \text{par}(x) = t \) and, thereby, the affine point \( P := (x, y) \). This is also the case if \( \alpha = 0 \) and \( t = 0 \), in which case \( x = 0 \) and the point \( P \) has order one or two. However, if \( \alpha = 0 \) and \( t = 1 \), the ordered pair \((t, y)\) does not correspond to the outcome of the point compression function.

Appendix J. Data Conversions

The string over some alphabet \( S \) consisting of the symbols \( x_{l-1}, x_{l-2}, \ldots, x_1, x_0 \) (each in \( S \), in this order, is denoted by \( \text{str}(x_{l-1}, x_{l-2}, \ldots, x_1, x_0) \). The length of this string (over \( S \)) is the number of symbols it contains (here: \( l \)). The empty string is the (unique) string of length \( l = 0 \).

An octet is an integer in the interval \([0, 256)\). An octet string is a string, where the alphabet is the set of all octets. A binary string (or bit string, for short) is a string, where the alphabet is the set \( \{0, 1\} \). Note that the length of a string is defined in terms of the underlying alphabet.

J.1. Conversion between Bit Strings and Integers

There is a 1-1 correspondence between bit strings of length \( l \) and the integers in the interval \([0, 2^l)\), where the bit string \( X := \text{str}(x_{l-1}, x_{l-2}, \ldots, x_1, x_0) \) corresponds to the integer \( x := x_{l-1} 2^{l-1} + x_{l-2} 2^{l-2} + \ldots + x_1 2 + x_0 \). (If \( l = 0 \), the empty bit string corresponds to the integer zero.) Note that while the mapping from bit strings to integers is uniquely defined, the inverse mapping from integers to bit strings is not, since any
non-negative integer smaller than $2^t$ can be represented as a bit string of length at least $t$ (due to leading zero coefficients in base 2 representation). The latter representation is called tight if the bit string representation has minimal length.

J.2. Conversion between Octet Strings and Integers (OS2I, I2OS)

There is a 1-1 correspondence between octet strings of length $l$ and the integers in the interval $[0, 256^l)$, where the octet string $X:=\text{str}(X_{l-1}, X_{l-2}, \ldots, X_1, X_0)$ corresponds to the integer $x:=X_{l-1}*256^{l-1} + X_{l-2}*256^{l-2} + \ldots + X_1*256 + X_0*1$. (If $l=0$, the empty string corresponds to the integer zero.) Note that while the mapping from octet strings to integers is uniquely defined, the inverse mapping from integers to octet strings is not, since any non-negative integer smaller than $256^t$ can be represented as an octet string of length at least $t$ (due to leading zero coefficients in base 256 representation). The latter representation is called tight if the octet string representation has minimal length. This defines the mapping OS2I from octet strings to integers and the mapping I2OS($x,l$) from non-negative integers smaller than $256^l$ to octet strings of length $l$.

J.3. Conversion between Octet Strings and Bit Strings (BS2OS, OS2BS)

There is a 1-1 correspondence between octet strings of length $l$ and and bit strings of length $8*l$, where the octet string $X:=\text{str}(X_{l-1}, X_{l-2}, \ldots, X_1, X_0)$ corresponds to the right-concatenation of the 8-bit strings $x_{l-1}, x_{l-2}, \ldots, x_1, x_0$, where each octet $X_i$ corresponds to the 8-bit string $x_i$ according to the mapping of Appendix J.1 above. Note that the mapping from octet strings to bit strings is uniquely defined and so is the inverse mapping from bit strings to octet strings, if one prepends each bit string with the smallest number of 0 bits so as to result in a bit string of length divisible by eight (i.e., one uses pre-padding). This defines the mapping OS2BS from octet strings to bit strings and the corresponding mapping BS2OS from bit strings to octet strings.

J.4. Conversion between Field Elements and Octet Strings (FE2OS, OS2FE)

There is a 1-1 correspondence between elements of a fixed finite field $GF(q)$, where $q=p^m$ and $m>0$, and vectors of length $m$, with coefficients in $GF(p)$, where each element $x$ of $GF(q)$ is a vector $(x_{m-1}, x_{m-2}, \ldots, x_1, x_0)$ according to the conventions of Appendix B.2. In this case, this field element can be uniquely represented by the right-concatenation of the octet strings $X_{m-1}, X_{m-2}, \ldots, X_1, X_0$, where each octet string $X_i$ corresponds to the integer $x_i$ in the interval $[0,p-1]$ according to the mapping of Appendix J.2 above. Note that both the mapping from field elements to
to octet strings and the inverse mapping are only uniquely defined if each octet string $X_i$ has the same fixed size (e.g., the smallest integer $l$ so that $256^l \geq p$) and if all integers are reduced modulo $p$. If so, the latter representation is called tight if $l$ is minimal so that $256^l \geq p$. This defines the mapping FE2OS($x$, $l$) from field elements to octet strings and the mapping OS2FE($X$, $l$) from octet strings to field elements, where the underlying field is implicit and assumed to be known from context. In this case, the octet string has length $1 \cdot m$.

J.5. Ordering Conventions

One can consider various representation functions, depending on bit-ordering and octet-ordering conventions.

The description below makes use of an auxiliary function (the reversion function), which we define both for bit strings and octet strings. For a bit string [octet string] $X$:=str($x_{l-1}$, $x_{l-2}$, ..., $x_1$, $x_0$), its reverse is the bit string [octet string] $X'$:=rev($X$):=str($x_0$, $x_1$, ..., $x_{l-2}$, $x_{l-1}$).

We now describe representations in most-significant-bit first (msb) or least-significant-bit first (lsb) order and those in most-significant-byte first (MSB) or least-significant-byte first (LSB) order.

One distinguishes the following octet-string representations of integers and field elements:

1. MSB, msb: represent field elements and integers as above, yielding the octet string str($X_{l-1}$, $X_{l-2}$, ..., $X_1$, $X_0$).

2. MSB, lsb: reverse the bit-order of each octet, viewed as 8-bit string, yielding the octet string str((rev($X_{l-1}$)), rev($X_{l-2}$)), ..., rev($X_1$), rev($X_0$)).

3. LSB, lsb: reverse the octet string and bit-order of each octet, yielding the octet string str(rev($X_0$)), rev($X_1$), ..., rev($X_{l-2}$), rev($X_{l-1}$)).

4. LSB, msb: reverse the octet string, yielding the octet string str($X_0$, $X_1$, ..., $X_{l-2}$, $X_{l-1}$).

Thus, the 2-octet string "07e3" represents the integer 2019 (=0x07e3) in MSB/msb order, the integer 57,543 (0xe0c7) in MSB/lsb order, the integer 51,168 (0xc7e0) in LSB/lsb order, and the integer 58,119 (=0xe307) in LSB/msb order.
Note that, with the above data conversions, there is still some ambiguity as to how to represent an integer or a field element as a bit string or octet string (due to leading zeros). However, tight representations (as defined above) are non-ambiguous.

Appendix K. Representations for Curve25519 Family Members

K.1. Wei25519

The representation of integers, field elements, affine points, and compressed points for the curve Wei25519 are as indicated below. Representations are relative to the prime field GF(p), where p = 2^255-19 is one of the general domain parameters of Appendix E.3.

Each field element z of GF(p) is represented as the octet string FE2OS(z), where one uses one the MSB/msb conventions and tight representation, as specified in Appendix J. In particular, each element of GF(p) is represented as a 32-byte octet string, which - when viewed as a bit string - has the leftmost bit position set to 0.

Each affine point (X, Y) of Wei25519 is represented as the right concatenation of the 32-byte octet representations for the x- and y-coordinate of this point according to the conventions above, i.e., it is represented as the 64-byte octet string str(FE2OS(X), FE2OS(Y)).

For each compressed point (X, t) of Wei25519, the parity bit t (which is an element of the field GF(2)), is represented as a 1-bit bit string, whereas the x-coordinate X (which is an element of GF(p)), is represented as a 32-byte octet string FE2OS(X). The result is "squeezed", by superimposing the 1-bit representation of t on the leftmost (unused) bit-position of the 32-byte octet representation of X.

Each integer in the interval [0,n-1] is viewed as an element of the prime field GF(n) and represented using MSB/msb conventions and a tight representation. In particular, each element of GF(n) is represented as a 32-byte octet string, which - when viewed as a bit string - has the leftmost three bit positions set to 0.

Appendix L. Auxiliary Functions

L.1. Square Roots in GF(q)

Square roots are easy to compute in GF(q) if q = 3 (mod 4) (see Appendix L.1.1) or if q = 5 (mod 8) (see Appendix L.1.2). Details on how to compute square roots for other values of q are out of scope.
If square roots are easy to compute in GF(q), then so are these in GF(q^2).

L.1.1. Square Roots in GF(q), where q = 3 (mod 4)

If y is a nonzero element of GF(q) and z := y^{(q-3)/4}, then y is a square in GF(q) only if y*z^2=1. If y*z^2=1, z is a square root of 1/y and y*z is a square root of y in GF(q).

L.1.2. Square Roots in GF(q), where q = 5 (mod 8)

If y is a nonzero element of GF(q) and z := y^{(z-5)/8}, then y is a square in GF(q) only if y^2*z^4=1.

a. If y*z^2=+1, z is a square root of 1/y and y*z is a square root of y in GF(q);

b. If y*z^2=-1, i*z is a square root of 1/y and i*y*z is a square root of y.

Here, i is an element of GF(q) for which i^2=-1 (e.g., i := 2^{(q-1)/4}). This field element can be precomputed.

L.2. Inversion

If y is an integer and gcd(y,n)=1, one can efficiently compute 1/y (mod n) via the extended Euclidean Algorithm (see Section 2.2.5 of [GECC]). One can use this algorithm as well to compute the inverse of a nonzero element y of a prime field GF(p), since gcd(y,p)=1.

The inverse of a nonzero element y of GF(q) can be computed as

1/y := y^{(q-2)} (since y^{(q-1)}=1).

Further details are out of scope. If inverses are easy to compute in GF(q), then so are these in GF(q^2).

The inverses of two nonzero elements y1 and y2 of GF(q) can be computed by first computing the inverse z of y1*y2 and by subsequently computing y2*z := 1/y1 and y1*z := 1/y2.

Author’s Address

Rene Struik
Struik Security Consultancy

Email: rstruik.ext@gmail.com