Decaf: Eliminating cofactors through point compression

Mike Hamburg

Rambus Cryptography Research
mhamburg@cryptography.com

This is a minor revision of a CRYPTO 2015 paper, ©2015 IACR

Abstract. We propose a new unified point compression format for Edwards, Twisted Edwards and Montgomery curves over large-characteristic fields, which effectively divides the curve's cofactor by 4 at very little cost to performance. This allows cofactor-4 curves to efficiently implement prime-order groups.

1 Introduction

"Let $\mathbb{G}$ be a group of prime order q." This defines the requirements for the main group in many cryptographic systems [1,9,16,18,19], most often with the intention that $\mathbb{G}$ will be the group of points on an elliptic curve. However, practical implementations usually do not quite deliver a group of prime order q, at least not without significant caveats. Implementations of prime-order curves usually have incomplete or variable-time addition formulas. For example, OpenSSL 1.0.1f, LibTomCrypt 1.17, PolarSSL 1.3.9 and Crypto++ 5.6.2 all use a branch to decide whether the inputs to their point-addition functions are equal, so that they can call the doubling function instead. Some of these libraries also have branches to detect cases where two points add to the identity point, or where one of them is the identity point. Even if this does not introduce timing variations in ECDH or ECDSA signing, it may introduce timing variations in other systems. Furthermore, there is a special case when encoding and decoding the identity point, which is at infinity in the Weierstrass model.

This problem can be mitigated by using complete addition laws. While such laws exist for prime-order curves [6,11], they are faster and much simpler for other elliptic curves such as (twisted) Edwards curves [14,5,4], Hessian curves [15], Jacobi quartics [8] or Jacobi intersections [26,20]. These curves have a *cofactor*, denoted h, where the order of the curve is $h \cdot q$ for some large prime q. The cofactor h is always divisible by 3 for Hessian curves, and by 4 for the other models.

1.1 Pitfalls of a cofactor

Many authors consider the advantages of a non-prime-order group, such the points on an Edwards curve, to outweigh the disadvantages. But the disadvantages are not negligible. There are several pitfalls which appear specifically for $h > 1$:

Small-subgroup attacks. Here an attacker sends a point whose order divides h, and a hapless user multiplies it by some scalar and uses the result. This will either result in a point known to the attacker (if the scalar is known to be divisible by h), or worse it may give the attacker information about the scalar. If the scalar is a private key, then leaking a few bits is a minor problem, though it is devastating to password-authenticated key exchange (PAKE) protocols [13].

Leaking a few bits of a scalar is a much more serious problem if the scalar is arithmetically related to a private key. Menezes and Ustaoglu used scalar leaks through the cofactor in their attack [27] on MQV [25] and HMQV [24]. HMQV was designed to avoid this weakness, but not successfully.

A related attack is to replace a point P with $P + T$, where T lies in a small subgroup. If the user multiplies by a scalar s, they will get $sP + sT$ instead of sP, where the difference sT gives away the low-order bits of s. Therefore, it isn't always enough to reject points in the small subgroup.

The usual defense against these attacks is to multiply certain points by h, and possibly to abort the protocol if the result is the identity. But one must decide which points to take these steps on, and the extra factor of h can complicate the arithmetic. In a prime-order group, this attack is easier to mitigate: at most, one must check for the identity point in the proper places.

Non-injective behavior. Multiplication by a scalar is a 1-to-1 function if the scalar is relatively prime to the group order. In a prime-order group, this is any scalar in $[1, q-1]$, and is true of a random scalar with high probability. The same is not true in a composite-order group. This means that adding a small-order element to e.g. a public key can produce the same result, possibly resulting in identity misbinding. This can be mitigated by making scalars relatively prime to h — exactly the opposite of techniques which clear the cofactor.

Covert channels. Non-injective behavior may make it easier to exfiltrate data through the cofactor component, even in protocols where behavior is otherwise deterministic.

Implementation-defined behavior. Some systems, such as Ed25519 [7], do not specify behavior when the inputs have a nonzero h-torsion component. In particular, Ed25519 signature verification can be different for batched vs singleton signatures, or between different implementations. This can cause disruption in protocols where all parties must agree on whether a signature is valid, such as a blockchain or Byzantine agreement. In other cases, it may make it easier to fingerprint implementations.

Nontrivial modifications. If a system or protocol is specified and proved secure on a prime-order group, then both the system and the proof may need to be changed for a group with a cofactor $h > 1$. Usually the modification is small. Often it is enough simply to multiply the outputs by h. However, only an expert will be able to tell exactly what modification is required. Cryptographic proofs are difficult, and this may represent enough work to prevent adoption.

1.2 Our contribution

The cofactor pitfalls can be avoided by using a related group of prime order q. The most obvious choice is the order-q subgroup of the elliptic curve. But validating membership in that subgroup is slow and complex, requiring either an extra scalar multiplication, checking for roots of division polynomials, or inverting multiple isogenies.

We propose two new ways to build a group of prime order q based on an Edwards or twisted Edwards curve $\mathcal{E}$ of order $4q$, thus eliminating the cofactor. In the first proposal of this paper, the group is $\mathcal{E}/\mathcal{E}[4]$. That is, two points on the curve $\mathcal{E}$ are considered equal if their difference has small order (dividing 4). This requires three changes:

- The function which checks equality of group elements must be modified.
- The function which encodes points before sending them must encode these equal points as equal sequences of bits.
- The function which decodes points must be designed to accept only the possible outputs of the encoding function.

Our second, improved proposal uses a different group $\psi(\mathcal{E})$, which in the usual case will be $2\mathcal{E}/\mathcal{E}[2]$. That is, only the even ($2q$-order) points of $\mathcal{E}$ are used, and two points are considered the same if they differ by a point of order 2. This requires the same three changes.

The difficult parts are the encoding and decoding routines, which are the main contributions of this paper. We describe the encoding algorithm as "compression" because its output is an element of the underlying field $\mathbb{F}$ rather than the usual two elements. In fact, it will be a "non-negative" element of $\mathbb{F}$, which allows us to save an additional bit.

It is important to note that internally, the points used in the first proposal can be any points on $\mathcal{E}$, and in the second proposal they can be any even points on $\mathcal{E}$. Points which differ by a point of order 4 (resp. 2) are considered equal, and will be encoded to binary strings in the same way. This is similar to using projective coordinates: two values in memory may be considered same point and encode to the same binary string, even though the X, Y and Z coordinates are different. This is how using a prime-order $\psi(\mathcal{E})$ instead of $\mathcal{E}$ mitigates small-subgroup attacks. Points of small order can appear internally, but they are considered equal to the identity element. Likewise $P + T$ can appear internally with T in a small-order subgroup, but it is considered equal to P and is encoded in the same way.

With the combination of the complete Edwards group law and our point encoding, protocols can gain the simplicity, security and speed benefits of (twisted) Edwards curves without any cofactor-related difficulty. The cost is a small increase in code complexity in the point encoding and decoding functions. On balance, we believe that our encoding can make the design of the entire system simpler. In terms of overhead, our encoding and decoding perform as well as existing point compression and decompression algorithms.

Designers often use untwisted Edwards, twisted Edwards or Montgomery curves. Montgomery curves give simple Diffie-Hellman protocols, and twisted Edwards curves give a speed boost but have incomplete formulas in fields of order 3 (mod 4). Our second proposal adds flexibility for curve choice. The same wire format can be used for a Montgomery curve as for its 4-isogenous Edwards and twisted Edwards curves. Furthermore, for twisted Edwards curves of cofactor 4, the subgroup we use avoids the incomplete cases in the addition laws.

Our group $\psi(\mathcal{E})$ and encoding algorithm can be used on curves with cofactor greater than 4. It still divides the cofactor by 4, so $\psi(\mathcal{E})$ will not have prime order. Additionally, $\psi(\mathcal{E})$ is not of the form $2\mathcal{E}/\mathcal{E}[2]$ if $\mathcal{E}$ has full 2-torsion.

We call this technique "Decaf" after the procedure which divides the effect of coffee by 4. We have built reference and optimized implementations of Decaf, and have posted them online at `http://sourceforge.net/p/ed448goldilocks/code/ci/decaf/tree/`. Our code carries out essentially all the operations described in this paper and appendices on the curve `Ed448-Goldilocks`, reducing the cofactor from 4 to 1.

2 Definitions and notation

Finite field. Let $\mathbb{F}$ be a finite field whose characteristic is neither 2 nor 3.

Even elements. An element g of an Abelian group $\mathbb{G}$ is said to be *even* if $g = 2h$ for some $h \in \mathbb{G}$. The even elements form a subgroup denoted $2\mathbb{G}$.

Torsion elements. An element g of a group $\mathbb{G}$ is a k-torsion element if $k \cdot g = 0_{\mathbb{G}}$. The k-torsion elements of an Abelian group form a subgroup usually denoted $\mathbb{G}[k]$. The k-torsion subgroup of an elliptic curve over a finite field has order dividing k^2; in particular, the 2-torsion subgroup has size 1, 2 or 4.

Projective space. Denote by $\mathbb{P}^n(\mathbb{F})$ the n-dimensional projective space over $\mathbb{F}$. Its elements are written as ratios $(X : Y : Z : \ldots)$, usually in upper-case. As a traditional short-cut, we usually write the elements of $\mathbb{P}^2(\mathbb{F})$ as a lower-case tuple (x, y) equivalent to $(x : y : 1)$, with the understanding that the equations involving these points may have be extended to cover "points at infinity" of the form $(X : Y : 0)$.

Twisted Edwards curves. Twisted Edwards curves have two parameters, a and d. They are specified as

$$\mathcal{E}_{a,d} := \left\{(x, y) \in \mathbb{P}^2(\mathbb{F}) : a \cdot x^2 + y^2 = 1 + d \cdot x^2 \cdot y^2\right\}$$

Another form, extended homogeneous coordinates [22], is used for high performance and simpler formulas:

$$\mathcal{E}_{a,d} := \left\{(X : Y : Z : T) \in \mathbb{P}^3(\mathbb{F}) : XY = ZT \text{ and } a \cdot X^2 + Y^2 = Z^2 + d \cdot T^2\right\}$$

We will use "untwisted" to mean $a = 1$. "Twisted" is the general case, which we sometimes narrow to $a = -1$. The identity point of any Edwards curve is $(0, 1) = (0 : 1 : 1 : 0)$.

An Edwards curve is called "complete" if d and ad are nonsquare in $\mathbb{F}$, which also implies that a is square. A complete Edwards curve has no points at infinity, and supports fast addition formulas which are complete in that they compute the correct answer for any two input points [5].

Montgomery curves. A Montgomery curve has two parameters, called A and B. It has the form

$$\mathcal{M}_{B,A} := \left\{(u, v) \in \mathbb{P}^2(\mathbb{F}) : Bv^2 = u \cdot (u^2 + Au + 1)\right\}$$

The identity point of this curve is a point at infinity, namely $(0 : 1 : 0)$. The curve is "untwisted" if $B = 1$. Over 3 (mod 4) fields, any twisted Montgomery curve can be put into a form with $B = 1$, but over 1 (mod 4) fields, this is not true. In particular, $B \neq 1$ is potentially useful to handle the twist of Curve25519, which has cofactor 4.

Jacobi quartic curves. A Jacobi quartic curve has two parameter, called A and e, and is defined by

$$\mathcal{J}_{e,A} := \left\{(s, t) \in \mathbb{P}^2(\mathbb{F}) : t^2 = es^4 + 2As^2 + 1\right\}$$

with an identity point at $(0, 1)$. The curve is "untwisted" if $e = 1$. We will only consider curves with $e = a^2$ in this paper; such curves always have full 2-torsion.

The curve parameters. As a corollary of Ahmadi and Granger's work [2], for any $a, d \in \mathbb{F}\backslash\{0, 1\}$, the following curves are isogenous:

$$\mathcal{E}_{a,d}; \quad \mathcal{E}_{-a,d-a}; \quad \mathcal{M}_{a,2-4d/a}; \quad \mathcal{J}_{a^2,a-2d}$$

Specifically, the Edwards, twisted Edwards and Montgomery curves are all 2-isogenous to the Jacobi quartic, and thus 4-isogenous to each other. We will write the 2-isogenies explicitly in Sections 4.1 and 5. Since our point encoding works on this family of isogenous curves, we will consider these specific curves parameterized by a and d.

We will write $\mathcal{E}$ as a shorthand for $\mathcal{E}_{a,d}$, $\mathcal{J}$ for $\mathcal{J}_{a^2,a-2d}$, and $\mathcal{M}$ for $\mathcal{M}_{a,2-4d/a}$.

Coset. In an Abelian group $\mathbb{G}$, the coset of a subgroup $H \subset \mathbb{G}$ with respect to an element $g \in \mathbb{G}$ is $H + g := \{h + g : h \in H\}$.

Non-negative field elements. Let $p > 2$ be prime. Define a residue $x \in \mathbb{F} = \mathbb{Z}/p\mathbb{Z}$ to be "non-negative" if the least absolute residue for x is in $[0, (p - 1)/2]$, and "negative" otherwise. This definition can be generalized (easily but non-canonically) to extension fields. Define $|x|$ to be x or $-x$, whichever is non-negative. Define $\sqrt{x}$ to be an arbitrary square root of x, not necessarily the non-negative one.

We chose this definition of non-negative because it is easy to evaluate, and it works over every odd-characteristic field. Alternative choices would be to distinguish by the low bit, or for fields 3 (mod 4), by the Legendre symbol. We avoided the Legendre symbol because it restricts field choices and is somewhat expensive to compute.

Encoding. For sets S and T, and encoding from S to T is an efficient function $\text{enc} : S \to T$ with efficient left-inverse $\text{dec} : T \to S \uplus \{\bot\}$, which fails by returning $\bot$ on every element of $T \setminus \text{enc}[S]$. We are interested in an encoding from an elliptic curve $\mathcal{E}$ over the field $\mathbb{F}$ to a binary set $\{0,1\}^n$ for some fixed n. We assume that the implementer has already chosen an encoding from $\mathbb{F}$ to binary. Since encodings can be composed and distributed over products, it suffices to encode to a set such as $\mathbb{F}$, $\mathbb{F}^2$ or $\mathbb{F} \times \{0,1\}$ which has a natural encoding to binary.

Compression. Since most elliptic curve forms are defined as subsets of $\mathbb{P}^2(\mathbb{F})$, they admit a straightforward encoding to $\mathbb{F}^2$ (and thence to binary) with a finite number of special cases corresponding to points at infinity. We call an encoding "point compression" or simply "compression" if its codomain is smaller than $\mathbb{F}^2$ when naturally encoded to binary. Most of the encoding algorithms in this paper map to the set $\mathbb{F}$ or to its non-negative elements, and so are point compression functions. The set of non-negative elements of $\mathbb{F}$ generally requires one fewer bit to encode than $\mathbb{F}$ itself.

3 An Edwards-only solution

There is a simple way to remove the a cofactor of 4 from an untwisted Edwards curve. A complete Edwards curve $\mathcal{E}_{a,d}$ has a 4-torsion subgroup of size exactly 4, whose coset with respect to $P = (x, y)$ is

$$\mathcal{E}[4] + P = \left\{(x,y); (y/\sqrt{a}, -x\sqrt{a}); (-x,-y); (-y/\sqrt{a}, x\sqrt{a})\right\}$$

Of this coset, there is exactly one representative point such that y and xy are both non-negative, and x is nonzero.[1] We can define the encoding of P to be the y-value of this representative. Note that the representation of the identity point is $(0, -1)$, so the identity point encodes to $0 \in \mathbb{F}$.

Similar solutions apply to incomplete Edwards curves. For curves whose 4-torsion group is Z_4, there is exactly one representative with y and y/x both finite and non-negative. For curves with full 2-torsion, there is exactly one representative with x finite and both y and $(y^2 + ax^2)/xy$ non-negative.

The usual addition formulas for incomplete Edwards curves produce the wrong answer $(0/0)$ for operations involving points at infinity, but are otherwise complete. Therefore, if the decoding operation chooses a coset representative in a subgroup that contains no points at infinity (e.g. in the prime-order subgroup),

[1] When $a = 1$, there is also exactly one representative where x and y are both non-negative, and x is nonzero, which could make for a simpler encoding.

then it is safe to use these curves. However, there is not an obvious way to make this section's decoding formulas restrict to a subgroup.

Furthermore, this format is not compatible with the fast, simple Montgomery ladder on Montgomery curves. We will remedy these problems using a slightly more complex encoding.

4 A solution from the Jacobi quartic

On the Jacobi quartic $\mathcal{J}_{a^2,a-2d}$, the coset of the 2-torsion group with respect to $P = (s,t)$ is exactly

$$\mathcal{J}[2] + P = \left\{(s,t); (-s,-t); (1/as, -t/as^2); (-1/as, t/as^2)\right\}$$

So a similar solution applies on $\mathcal{J}$ modulo its 2-torsion: we can encode a point P by the s-coordinate of the coset representative (s,t), where s is non-negative and finite, and t/s is non-negative or infinite[2]. Call this encoding $\text{enc}_{\mathcal{J}}(P)$, and call the corresponding decoding algorithm $\text{dec}_{\mathcal{J}}$. Note that the identity point encodes to $0 \in \mathbb{F}$.

4.1 From the Jacobi quartic to Edwards curves

The curves $\mathcal{E}_{a,d}$ and $\mathcal{J}_{a^2,a-2d}$ are isogenous by the map

$$\phi_a(s,t) = \left(\frac{2s}{1+as^2}, \frac{1-as^2}{t}\right) \text{ with dual } \bar{\phi}_a(x,y) = \left(\frac{x}{y}, \frac{2-y^2-ax^2}{y^2}\right)$$

Note that swapping (a,d) with $(-a, d-a)$ results in the same curve $\mathcal{J}_{a^2,a-2d}$, and gives an isogeny ϕ_{-a} to the curve $\mathcal{E}_{-a,d-a}$.

We will need the following lemma, whose trivial proof is omitted:

Lemma 1. *Let ϕ be a homomorphism from an abelian group $\mathbb{G}$ to another abelian group $\mathbb{H}$, and let $\mathbb{G}'$ be a subgroup of $\mathbb{G}$. Then ϕ acts as a well-defined homomorphism from $\mathbb{G}/\mathbb{G}'$ to $\phi[\mathbb{G}]/\phi[\mathbb{G}']$ which is a subgroup of $\mathbb{H}/\phi[\mathbb{G}']$. Furthermore, if $\ker\phi \subseteq \mathbb{G}'$, then ϕ acts as an isomorphism between these groups.*

Since the isogeny ϕ_a is a group homomorphism whose kernel is in $\mathcal{J}[2]$, we can extend the encoding on $\mathcal{J}/\mathcal{J}[2]$ to an encoding on $\phi_a[\mathcal{J}]/\phi_a[\mathcal{J}[2]]$:

$$\text{enc}(P) := \text{enc}_{\mathcal{J}}(\phi_a^{-1}(P)) \text{ with } \text{dec}(b) := \phi_a(\text{dec}_{\mathcal{J}}(b))$$

The lemma shows that both encoding and decoding are well-defined. In particular, P has two preimages under ϕ_a, but they represent the same element of $\mathcal{J}/\mathcal{J}[2]$ and have the same encoding under $\text{enc}_{\mathcal{J}}$.

Let $\psi(\mathcal{E})$ denote the group $\phi_a[\mathcal{J}]/\phi_a[\mathcal{J}[2]]$. If the 4-torsion group of $\mathcal{E}$ is cyclic, then $\psi(\mathcal{E})$ is more simply expressed as $2\mathcal{E}/\mathcal{E}[2]$.

[2] Checking the sign of s/t works about as well as t/s in our formulas; the choice of t/s is more or less arbitrary.

4.2 Encoding

When encoding from $\psi(\mathcal{E})$, we are given a point $P = (x, y)$ in the image of ϕ_{aq} on $\mathcal{E}$. We need to efficiently compute s where $(s, t) = \phi_a^{-1}(x, y)$. We know that

$$\begin{aligned} x &= 2s/(1 + as^2) \\ \text{so } s &= (1 \pm \sqrt{1 - ax^2})/ax \end{aligned}$$

Also,

$$\begin{aligned} y &= (1 - as^2)/t \\ \text{so } t/s &= (1 - as^2)/sy \\ &= \mp 2\sqrt{1 - ax^2}/xy \end{aligned}$$

It turns out to be particularly straightforward to compute this encoding from the popular extended homogeneous coordinates. Explicit formulas are given in Appendix A.1.

4.3 Decoding

To decode, we are given s and must compute

$$(x, y) = \left(\frac{2s}{1 + as^2}, \; \frac{1 - as^2}{\sqrt{a^2 s^4 + (2a - 4d)s^2 + 1}} \right)$$

with the square root t taken so that t/s is non-negative. This requires the "inverse square root trick" to compute $1/s$ and t at the same time, with care to avoid division by 0. The exact formulas are given in Appendix A.2. The input must be rejected if s is negative or if it is not a field element (eg. if it is the binary encoding of a number $\geq p$), or if the square root doesn't exist.

It is simplest to decode to projective form, so that the denominators need not be cleared. It is also relatively easy to decode to affine form by batching a computation of $1/(1 + as^2)$ with the square root. Decoded points always have a well-defined affine form on curves with cofactor exactly 4, because those curves have no points at infinity in the image of ϕ_a.

4.4 Completeness

Importantly, if the cofactor of $\mathcal{J}$ is exactly 4, then the image $\psi(\mathcal{E})$ contains no points at infinity. An easy way to see this is that if $\phi_a(s, t)$ were at infinity, then $\bar{\phi}_a(\phi_a(s, t))$ would be either at infinity or at $(0, -1)$. In either case, it would be a nontrivial 2-torsion point [21]. But it cannot be a 2-torsion point, because $\bar{\phi}_a \circ \phi_a$ is the doubling map on $\mathcal{J}$ (by definition of an isogeny), and its image is exactly the subgroup of order q.

4.5 Equality

Ordinarily, testing for equality in a quotient group $\mathbb{G}/\mathbb{H}$ requires testing whether $P = Q + H$ for each $H \in \mathbb{H}$. But if the cofactor is exactly 4, then equality testing is actually easier on $\psi(\mathcal{E})$ than on $\mathcal{E}$. In this case, two points $(X_1 : Y_1 : Z_1 : T_1)$ and $(X_2 : Y_2 : Z_2 : T_2)$ are equal if and only if

$$X_1 \cdot Y_2 = X_2 \cdot Y_1$$

This is because X/Y is the s-coordinate of the image Q of $\bar{\phi}_a(X : Y : Z : T)$ on $\mathcal{J}$. The only other point with that s-coordinate has a nontrivial 2-torsion component (it is $(0, -1) - Q$), but the image $(\bar{\phi}_a \circ \phi_a)[\mathcal{J}]$ is the prime-order subgroup $\mathcal{J}[q]$.

In particular, for a curve of cofactor exactly 4, a point $(X : Y : Z : T)$ is equal to the identity precisely when $X = 0$.

4.6 Security

Using Decaf gives the security benefits of a prime-order group without weakening well-studied cryptographic assumptions. In particular:

- The discrete logarithm problem is equivalent on $\mathcal{E}, \mathcal{J}$ and $\psi(\mathcal{E})$. The same is true for computational Diffie-Hellman, gap DH, static DH, strong DH, and should hold for similar computation problems.
- If the Decaf group $\psi(\mathcal{E})$ has prime order q, then the DDH problem is equivalent on $\mathcal{E}[q], \mathcal{J}[q]$ and $\psi(\mathcal{E})$. The same is true for decision linear, and should hold for similar decision problems. These decision problems are easy on groups with a small cofactor, such as $\mathcal{E}$ itself.

The straightforward proofs of these reductions are omitted.

4.7 Batch encoding

On a server which needs to generate signatures and/or ephemeral keys at prodigious rates, it may be advantageous to batch the point encoding algorithm.

The encoding algorithm listed above cannot be batched easily because of the inverse square root computation. However, the square root can be avoided if we wish to compress $2P$ instead of P, that is, if P is computed as $(k/2 \mod q) \cdot B$ instead of $k \cdot B$. In this case, we can simply evaluate the dual 2-isogeny $\bar{\phi}$ from $\mathcal{E}$ to $\mathcal{J}$:

- Compute $1/(xy)$ and $t/s = (2 - y^2 - ax^2)/xy$.
- If t/s is non-negative, then output $|s| = |x/y| = |x^2/xy|$.
- Otherwise output $|1/s| = |y/x| = |y^2/xy|$.

The computation of $1/(xy)$ can be batched over multiple points using Montgomery's trick.

Encoding	enc cost	dec cost	clear h	order	factor	size
(x, y)	$1I + 2M$	$3M$	$12M$	$4q \to q$	4	$2\lceil \lg \lvert\mathbb{F}\rvert \rceil$
$(x, \operatorname{sign} y)$	$1I + 2M$	$1I_2 + 3M$	$12M$	$4q \to q$	4	$\lceil \lg \lvert\mathbb{F}\rvert \rceil + 1$
$(x, \operatorname{sign} y)$, check	$1I + 2M$	$> 2I_2 + L$	0	q	1	$\lceil \lg \lvert\mathbb{F}\rvert \rceil + 1$
First proposal	$1I + 2M$	$1I_2 + 3M$	0	$4q$	1	$\lceil \lg \lvert\mathbb{F}\rvert \rceil - 1$
Second proposal	$1I_2 + 7M$	$1I_2 + 10M$	0	$2q$	1	$\lceil \lg \lvert\mathbb{F}\rvert \rceil - 1$
Batchable	$1I + 6M$	$1I_2 + 10M$	0	$2q$	2	$\lceil \lg \lvert\mathbb{F}\rvert \rceil - 1$

Fig. 1. Cost of encoding and decoding algorithms. M = multiply, I = inversion, I_2 = inverse square root, L = Legendre symbol. Squarings are treated as $0.8M$ and multiplies by constants as $0.2M$, but columns are rounded to the nearest M.

4.8 Performance

Overall, Decaf's performance is very similar to a traditional point compression scheme. Encoding and decoding take one field exponentiation each.

A comparison to existing point encoding algorithms is shown in Figure 1. It shows:

- The encoding and decoding costs.
- The cost to clear the cofactor if one remains.
- The order of the resulting points on the curve, with $4q \to q$ meaning a cofactor that will most likely be cleared.
- The extra factor induced by encoding and cofactor clearing.
- The size in bits of the encoding's codomain.

If inversion I is implemented using Fermat's little theorem, it is likely to be slightly more expensive than an inverse square root I_2. In practice, implementations that need both I and I_2 with $|\mathbb{F}| \equiv 3 \pmod 4$ often implement inversion as $x/(\pm\sqrt{x^2})^2$, costing $M + 2S$ more, and this is usually close to optimal anyway.[3]

The (x, y) method is uncompressed, and $(x, \operatorname{sign} y)$ is classically compressed. These methods do not remove the cofactor, so many protocols will remove it at the cost of two doublings $\approx 12M$. This changes the order of the internal points from $4q$ to q. The third row is compression with order checking. The order checking can be accomplished by inverting a 2-isogeny twice: the first inversion requires an inverse square root, but the second requires only checking that the root exists, i.e. computing a Legendre symbol.

The first proposal (Section 3) is a quotient group on an untwisted Edwards curve. It is slightly more expensive on a twisted Edwards curve, and is dangerous for such curves when $|\mathbb{F}| \equiv 3 \pmod 4$ because the internal points can have order 4. The second proposal (Section 4) avoids this problem, and gives an encoding

[3] We tested a simple dynamic program which computes 3-register powering ladders, and it gave this technique for NIST P-192, P-256, P-384, P-521, and also for $2^{414} - 17$ (Curve41417) and $2^{448} - 2^{224} - 1$ (Ed448-Goldilocks).

compatible with several curve models, but at the cost of about 8 extra field multiplications and correspondingly higher complexity.

A downside of methods which include an inverse square root I_2 is that they cannot use an EGCD-based inversion method. They also cannot be batched using Montgomery's batch inversion trick, which accomplishes N inversions using one inversion and $3(N-1)$ multiplications. The batchable encoding method (Section 4.7) replaces the inverse square root in encoding with an inversion but multiplies by an extra factor of 2.

It is seen that our methods cost less in total than point compression plus clearing the cofactor. Even for operations which do not need to clear the cofactor (eg. key generation), the overhead from our encoding is relatively small. Fast key generation operations cost on the order of 3M per bit of the curve's order, so the difference in encoding costs is well under 1% for cryptographically useful curves.

5 Compatibility with Montgomery curves

The Montgomery ladder on Montgomery curves is a very simple and fast way to implement scalar multiplication for Diffie-Hellman (DH) key exchange. In its simplest form, the ladder discards sign information, making it inherently incompatible with any point encoding format that conveys sign information. Furthermore, it does not distinguish between the curve and its quadratic twist, necessitating the use of twist-safe curves [3]. However, we would like to interoperate with the Montgomery ladder with minimal changes. For example, if a protocol uses a $(u, \operatorname{sign} v)$ format, then the ladder can be modified to compute the sign, or the protocol can be changed to discard the sign bit for DH outputs.

We will show how to use Decaf with the Montgomery ladder on the curve

$$\mathcal{M}_{a,2-4d/a} : av^2 = u \cdot (u^2 + (2-4d/a) \cdot u + 1)$$

where conveniently the value of $(A+2)/4$ is $1 - d/a$. The curve $\mathcal{M}_{a,2-4d/a}$ is isogenous to $\mathcal{J}_{a^2,a-2d}$ by the maps

$$\phi(s,t) = \left(\frac{1}{as^2}, -\frac{t}{as^3}\right) \quad \text{and} \quad \bar{\phi}(u,v) = \left(\frac{1-u^2}{2av}, \frac{a(u+1)^4 + 8du(u^2+1)}{4a^2v^2}\right)$$

More simply, $\phi(s,t) = (as^2, ts) + T_2$, where the 2-torsion point T_2 can be ignored due to the quotient. This means that Montgomery ladder implementations can take input in Decaf format, simply by starting the ladder at $u = as^2$.

When the ladder finishes, it is possible to efficiently encode the output point in the Decaf point format, including the correct sign information for v. However, recovering the sign information is complicated. Furthermore, it is possible to reject elements on the twist rather than on the curve, which the usual Montgomery ladder does not do, and it is possible to do all of this with only one field exponentiation (an inverse square root). This means that the Montgomery ladder will behave exactly the same as a standard decoding, scalar multiplication and encoding. We give the full details of how to do this in Appendix B. Some of the formulas in that section may be of independent interest.

It is also possible (and complicated) to do these things with existing point formats such as $(u, \operatorname{sign} v)$, but almost no implementations do. Instead, since the Montgomery ladder is used almost exclusively for Diffie-Hellman, most implementations clear the cofactor and output only u, losing the information about v. This leaves the Montgomery ladder code very simple. It is also easy to do this with the Decaf encoding, by clearing the cofactor[4] and outputting $|1/\sqrt{au}|$. The implementation should abort on $u = 0$ and $u = \infty$, which lie in a small subgroup. This will also reject points on the twist, because even points on the twist have either $u = 0, u = \infty$ or au nonsquare.

An Edwards or twisted Edwards implementation can interoperate with this simpler behavior simply by computing $|s| = |x/y|$, instead of encoding any sign information.

6 Hashing to the curve

Some protocols require a map from $\mathbb{F}$ to a curve [10,19,23], either to build a hash function which is either indifferentiable from a random oracle, or at least suitable for encoding computational Diffie-Hellman (CDH) challenges in the random oracle model. We could do this by using Elligator 1 or 2 on $\mathcal{E}$ or $\mathcal{M}$. Since Decaf only operates subgroups on these curves, we would need to double the output of the map to make sure it is in the subgroup.

However, there is a better solution. We can instead use Elligator 2 on the Jacobi quartic $\mathcal{J}$, since it has a point of order 2. Then we can translate this point to the Edwards and Montgomery curves using the isogeny. That way, the groups and maps implemented by these curves are all compatible. The formulas for Elligator 2 are found in Appendix C. It is important to note that Elligator 2 provides a 1:1 map to a group of order $h \cdot q$, not of order $(h/4) \cdot q$. Therefore, the map be up to 4:1 once the isogeny and quotient are applied.

This map is suitable for deriving CDH challenges from a random oracle. That is, it is still suitable for use in derivatives of BLS [10], SPAKE2 [1][5], SPEKE [23] and possibly Dragonfly [19][6]. These protocols do not require a random oracle map to $\mathbb{G}$. They only require a map from strings to the curve which is at most k-to-1 for small k, hits at least a $1/\ell$ fraction of the points for small ℓ, and whose inverse is efficiently sampleable. When a full random oracle map to $\mathbb{G}$ is

[4] Clearing the cofactor takes one doubling on a cofactor-4 curve, because of the isogeny. Another option would be to quotient out the cofactor, by choosing the lexicographically greater of $|\sqrt{u/a}|$ and $|1/\sqrt{au}|$. But this is more complex and doesn't reject points on the twist.

[5] Replacing SPAKE2's $H(\text{password}) \cdot (M, N)$ with an Elligator-like map results in a protocol with the same properties but no static-CDH assumption.

[6] Dragonfly lacks a security proof, so we cannot actually be sure that any such map is suitable. But given the similarity to SPEKE this seems likely.

required, Brier et al.'s result [12] shows that mapping two independently chosen field elements and adding them is sufficient.

It is still possible to use Elligator 2 as a partial steganographic encoding for public keys, as in EKE. One may invert the isogeny to obtain a point on $\mathcal{J}$, randomize its 2-torsion components, and apply the inverse map defined by Elligator 2. Unfortunately, this requires an extra randomization step and an extra inverse-square-root operation compared to the original Elligator 2.

7 Removing larger cofactors

Quotient groups can be used to remove larger cofactors. However, it is not immediately clear which point should be the distinguished one. Simply using the least lexicographic encoding would work. It would take linear time in h to scan every point in the coset, but still only one inversion because of Montgomery's batch inversion trick. Such a linear-time scan would be required both to encode and to decode a point.

However, in the case of complete Edwards curves $\mathcal{E}$ with $\mathcal{E}[8]$ isomorphic to Z_8, there is a simpler way to distinguish one point from a coset. In this case, $\mathcal{J}$ has a small subgroup isomorphic to $Z_2 \times Z_4$. The Z_2-component is the kernel of the isogeny, so that the Z_4 component maps to the 4-torsion subgroup of $\mathcal{E}$. To choose a distinguished point in a coset $S = \mathcal{E}[8] + P$ it suffices to choose one of the two points (x, y) with xy non-negative and $y \neq 0$. This chooses a point modulo $\mathcal{E}[2]$, and then one can run the encoding for $\mathcal{E}/\mathcal{E}[2]$ to produce the final encoding. It is possible to do this with only one inverse square root. To decode the point, one must decode it to affine form instead of projective, and check that xy is non-negative.

This allows Decaf to operate on Ed25519 or the isogenous curve $\mathcal{E}_{1,-121655}$. Since over a 1-mod-4 field, $\mathcal{E}_{a,d}$ and $\mathcal{E}_{-a,-d}$ are isomorphic by the map $x \to x\sqrt{-1}$, arithmetic on either curve can still take advantage of the faster complete twisted Edwards formulas.

8 Future work

We do not believe that Decaf is the last word in cofactor-reducing compression algorithms. Additionally, an improved encoding scheme with simpler formulas would make this technique more compelling.

9 Conclusion

We have shown a straightforward way to implement a prime-order group $\mathbb{G}$ using Edwards, twisted Edwards, Jacobi quartic and Montgomery curves. All four curve shapes implement the same group and so are compatible, except that as usual it is complicated to make the Montgomery ladder retain sign information. Our technique is otherwise similar in complexity and performance to traditional point compression techniques, though it may improve performance by making faster curves safe. Furthermore, we have shown how to implement an Elligator-like map from $\mathbb{F}$ to $\mathbb{G}$, which is also compatible with all 4 models.

10 Acknowledgements

The author thanks Mark Marson, Steven Galbraith and the anonymous reviewers for their editorial suggestions.

References

1. Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. *Topics in cryptology–CT-RSA 2005*, pages 191–208, 2005.
2. Omran Ahmadi and Robert Granger. On isogeny classes of edwards curves over finite fields. Cryptology ePrint Archive, Report 2011/135, 2011. `http://eprint.iacr.org/2011/135`.
3. Daniel Bernstein. Curve25519: new Diffie-Hellman speed records. *Public Key Cryptography-PKC 2006*, pages 207–228, 2006.
4. Daniel Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. *Progress in Cryptology–AFRICACRYPT 2008*, pages 389–405, 2008.
5. Daniel Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In *Advances in cryptology–ASIACRYPT 2007*, pages 29–50. Springer, 2007.
6. Daniel Bernstein and Tanja Lange. Complete addition laws for elliptic curves, 2009. `http://cr.yp.to/talks/2009.04.17/slides.pdf`.
7. D.J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.Y. Yang. High-speed high-security signatures. *Cryptographic Hardware and Embedded Systems, CHES 2011*, 2011.
8. Olivier Billet and Marc Joye. The jacobi model of an elliptic curve and side-channel analysis. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 34–42. Springer, 2003.
9. Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *Advances in Cryptology–CRYPTO 2008*, pages 108–125. Springer, 2008.
10. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology?ASIACRYPT 2001*, pages 514–532. Springer, 2001.
11. Joppe W. Bos, Craig Costello, Patrick Longa, and Michael Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. Cryptology ePrint Archive, Report 2014/130, 2014. `http://eprint.iacr.org/`.
12. Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In *Advances in Cryptology–CRYPTO 2010*, pages 237–254. Springer, 2010.
13. Dylan Clarke and Feng Hao. Cryptanalysis of the dragonfly key exchange protocol. *IET Information Security*, 8(6):283–289, 2014.
14. H.M. Edwards. A normal form for elliptic curves. *Bulletin-American Mathematical Society*, 44(3):393, 2007.
15. Reza R Farashahi and Marc Joye. Efficient arithmetic on hessian curves. In *Public Key Cryptography–PKC 2010*, pages 243–260. Springer, 2010.

16. Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the diffie-hellman problems. *Journal of Cryptology*, 20(4):493–514, 2007.
17. Mike Hamburg. Fast and compact elliptic-curve cryptography. Cryptology ePrint Archive, Report 2012/309, 2012. `http://eprint.iacr.org/2012/309`.
18. Feng Hao and Peter Ryan. J-pake: Authenticated key exchange without pki. Cryptology ePrint Archive, Report 2010/190, 2010. `http://eprint.iacr.org/`.
19. Dan Harkins. Dragonfly: A pake scheme, 2012. `http://www.ietf.org/proceedings/83/slides/slides-83-cfrg-0.pdf`.
20. Hüseyin Hışıl, Kenneth Wong, Gary Carter, and Ed Dawson. Faster group operations on elliptic curves. Cryptology ePrint Archive, Report 2007/441, 2007. `http://eprint.iacr.org/`.
21. Huseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Jacobi quartic curves revisited. Cryptology ePrint Archive, Report 2009/312, 2009. `http://eprint.iacr.org/2009/312`.
22. Hüsseyin Hışıl, Kenneth Wong, Gary Carter, and Ed Dawson. Twisted edwards curves revisited. *Advances in Cryptology–ASIACRYPT 2008*, pages 326–343, 2008.
23. David P Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, 1996.
24. Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *Advances in Cryptology–CRYPTO 2005*, pages 546–566. Springer, 2005.
25. Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.
26. P-Y Liardet and Nigel P Smart. Preventing spa/dpa in ecc systems using the jacobi form. In *Cryptographic Hardware and Embedded Systems?CHES 2001*, pages 391–401. Springer, 2001.
27. Alfred Menezes and Berkant Ustaoglu. On the importance of public-key validation in the mqv and hmqv key agreement protocols. In *Progress in Cryptology-INDOCRYPT 2006*, pages 133–147. Springer, 2006.
28. Peter Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.

A Explicit formulas for encoding and decoding

A.1 Encoding

We wish to compute

$$s = (1 \pm \sqrt{1-ax^2})/ax \quad \text{and} \quad t/s = \mp 2\sqrt{1-ax^2}/xy$$

We know from the curve equation that

$$(1-ax^2)\cdot(1-y^2) \quad = \quad 1+ax^2y^2-(y^2+ax^2) \quad = \quad (a-d)x^2y^2$$

so that

$$\sqrt{1-ax^2}/xy = \pm\sqrt{(a-d)/(1-y^2)}$$

Observe that in extended homogeneous coordinates,

$$1/x^2 = (a-dy^2)/(1-y^2) = (aZ^2-dY^2)/(Z^2-Y^2)$$

and therefore

$$1/x = (aZX - dYT)/(Z^2 - Y^2)$$

This leads to a relatively simple encoding formula given an inverse-square-root algorithm:

$$\begin{aligned}
&r \leftarrow 1/\sqrt{(a-d)\cdot(Z+Y)\cdot(Z-Y)}\\
&u \leftarrow (a-d)\cdot r\\
&r \leftarrow -r \text{ if } -2\cdot u\cdot Z \text{ is negative}\\
&s \leftarrow |u\cdot(r\cdot(aZ\cdot X - d\cdot Y\cdot T)+Y)/a|
\end{aligned}$$

In theory, this formula has an exceptional case because it divides by 0 when $y = \pm 1$. But if the inverse square root function returns $r = 0$ in this case, the correct answer $s = 0$ will emerge.

A.2 Decoding

To decode to $\mathcal{E}_{a,d}$, we must recover

$$(x, y) = \left(\frac{2s}{1+as^2}, \frac{1-as^2}{\sqrt{a^2s^4+(2a-4d)s^2+1}}\right)$$

where $\sqrt{a^2s^4+(2a-4d)s^2+1}/s$ is non-negative. We can compute this as:

$$\begin{aligned}
&\text{Reject unless } s = |s|\\
X &\leftarrow 2\cdot s\\
Z &\leftarrow 1 + a\cdot s^2\\
u &\leftarrow Z^2 - 4d\cdot s^2\\
v &\leftarrow \begin{cases} 1/\sqrt{u\cdot s^2} & \text{if } u\cdot s^2 \text{ is square and nonzero}\\ 0 & \text{if } u\cdot s^2 = 0\\ \text{[reject]} & \text{if } u\cdot s^2 \text{ is not square}\end{cases}\\
v &\leftarrow -v \text{ if } u\cdot v \text{ is negative}\\
w &\leftarrow v\cdot s\cdot(2-Z)\\
w &\leftarrow w+1 \text{ if } s = 0\\
Y &\leftarrow w\cdot Z\\
T &\leftarrow w\cdot X\\
P &\leftarrow (X:Y:Z:T)
\end{aligned}$$

The special case for $s = 0$ is required to decode $s = 0$ to the identity point instead of the point $(0, 0)$, which isn't on the curve.

B The modified Montgomery ladder

We use a modified version of the traditional Montgomery ladder on Montgomery curves [28]. This uses our knowledge of s_0 such that the initial state $u_0 = as_0^2$. It converts

$$(s_0, U_1 : Z_1, U_2 : Z_2) \text{ where } (s_0 : v_0 : 1) + (U_1 : V_1 : Z_1) - (U_2 : V_2 : Z_2) = 0_{\mathcal{M}}$$

for some unknown v_0, V_1, V_2, to

$$(s_0, U_3 : Z_3, U_4 : Z_4)$$

where

$$(U_4 : V_4 : Z_4) = 2(U_2 : V_2 : Z_2) \text{ and } (U_3 : V_3 : Z_3) = (U_1 : V_1 : Z_1) + (U_2 : V_2 : Z_2)$$

again with the V components unknown. Note that on $\mathcal{M}_{a,2-4d/a}$, the coefficient $(A+2)/4 = 1 - d/a$. The ladder works as follows:

$$\begin{aligned}
E &\leftarrow U_2 + Z_2 \\
F &\leftarrow U_2 - Z_2 \\
G &\leftarrow U_1 + Z_1 \\
H &\leftarrow U_1 - Z_1 \\
K &\leftarrow E^2 - F^2 \\
U_3 &\leftarrow (EH + FG)^2 \\
Z_3 &\leftarrow a \cdot (s_0 \cdot (EH - FG))^2 \\
U_4 &\leftarrow E^2 \cdot F^2 \\
Z_4 &\leftarrow K \cdot (F^2 + (1 - d/a) \cdot K)
\end{aligned}$$

The computation of the common subexpressions EH, FG, E^2 and F^2 has been removed for brevity. The salient feature of this ladder is that it computes as intermediate results S_3 and W_3 which are square roots of X_3 and Z_3/a respectively. Our implementation allocates is temporary variables in a way that prevents these intermediates from being overwritten until the next iteration of the ladder. This means that they will be available to the encoding algorithm.

B.1 Encoding

At the end of the Montgomery ladder, we need to encode the point stored in $U_4 : Z_4$. Recall that a point on $\mathcal{M}_{a,2-4d/a}$ is the image of an isogeny from $\mathcal{J}_{a^2,a-2d}$:

$$\phi((s,t)) = (as^2, st) + T_2 \text{ with inverse } \phi^{-1}((u,v) + T_2) = (\sqrt{u/a}, v/\sqrt{u/a})$$

Because the encoding is the same modulo 2-torsion components, we can ignore T_2 completely. The principal difficulty of encoding is to simultaneously determine

$\sqrt{u_2}$ and its inverse from the ladder state, and whether $t_2/s_2 = av_2/u_2$ is positive given that $t_0/s_0 = av_0/u_0$ is known to be positive. Fortunately, the invariants of the ladder make this possible. From [17], Appendices A.2 and A.3, we know that if

$$(u_0, v_0) + (u_1, v_1) + (u_2, v_2) = 0 \text{ on } \mathcal{M}$$

that is, if these points lie on a line, then

$$2av_1v_2 = (u_1u_2 + 1)(u_1 + u_2) - u_0(u_1 - u_2)^2 + 2Au_1u_2 \tag{1}$$

and symmetrically, and also

$$4(u_0 + u_1 + u_2 + A)(u_0u_1u_2) = (1 - u_0u_1 - u_1u_2 - u_0u_2)^2 \tag{2}$$

Adding (2) to $2u_0 \cdot (1)$, we have

$$4au_0v_1v_2 = (1 - u_1u_2 - u_0u_2 + u_0u_1) \cdot (1 - u_1u_2 + u_0u_2 - u_0u_1) \tag{3}$$

and symmetrically. Multiplying and dividing symmetric copies of this equation, we obtain

$$4au_0u_1v_2^2/u_2 = (1 - u_1u_2 - u_0u_2 + u_0u_1)^2$$

so that

$$\frac{v_2}{u_2} = \frac{1 - u_1u_2 - u_0u_2 + u_0u_1}{\pm 2\sqrt{au_0u_1u_2}}$$

This equation cannot determine the sign of the square root, but (3) shows it to be consistent for all three points on the line.

This means that it is enough to compute $\pm 1/\sqrt{au_0u_1u_2}$. This will allow us to determine av_0/u_0 to adjust the sign of the square root. It will allow us to check whether av_2/u_2 is negative, in which case we should output $1/\sqrt{au_2}$ instead of $\sqrt{u_2/a}$. Furthermore, the input point s_0 is $\sqrt{u_0/a}$, and the modified Montgomery ladder state contains either $\sqrt{au_1}$ or $\sqrt{au_2}$, depending on the last bit of the ladder. This allows us to compute $\sqrt{u_2/a}$ or its inverse from the ladder state and $1/\sqrt{au_0u_1u_2}$ with no additional field exponents. In the actual computation, u_1 and u_2 are given in projective form, but this does not greatly complicate matters because the equations are nearly homogeneous.

Special cases If u_2 is zero or infinite, then it is a 2-torsion point and the output is zero. Likewise if $u_0 = 0$, then the base point is the identity and again the output will always be zero. If u_1 is zero or infinite, then the output is either the initial point or its inverse, depending on whether the last step in the ladder swapped u_1 and u_2. So the output should be either s_0 or $|1/as_0|$.

Twist rejection The above procedure will reject points which are on the twist of $\mathcal{M}$ instead of on $\mathcal{M}$ itself, because $au_0u_1u_2$ will not be square for such points. However, there is one sticking point: if the secret key is 0 or ± 1 modulo the twist's group order q', then u_1 or u_2 may be 0 even if $u_0 \neq 0$. This would make

$au_0u_1u_2 = 0$ a square number. This quirk is unlikely to apply in a real system, but it is still worth avoiding.

To fix this issue, we can test whether $au_0u_1u_2 = 0$; if so, the special case above is used to determine the sign, so the output of the square root doesn't matter. Therefore, the square root can be changed to $\sqrt{u_0^2 + Au_0 + 1}$ to determine whether the input point was on the curve or not.

C Elligator 2

The Jacobi quartic $\mathcal{J}_{a^2,a-2d}$ is birationally equivalent to the Legendre curve

$$ay^2 = x(x-1)(x-d/a)$$

by the maps

$$(s,t) = \left(\frac{ax-d}{a^2y}, \frac{ax^2-2dx+d}{ax(x-1)}\right); \quad (x,y) = \left(\frac{as^2+t+1}{2as^2}, \frac{as^2+t+1-2ds^2}{2a^2s^3}\right)$$

Since the Legendre curve has 3 points of order 2, we have a choice of which to use as the point of order 2 in Elligator 2. It turns out to be best for symmetry if we use $(d/a, 0)$.

The Elligator 2 map E to $\mathcal{J}_{a^2,a-2d}$ starts with a fixed quadratic nonresidue n and an input $r_0 \in \mathbb{F}$, and begins by computing a probable nonresidue $r := nr_0^2$. It then returns

$$(s,t) = \left(+\sqrt{\frac{(r+1)(a-2d)}{(dr+a-d)(dr-ar-d)}}, \quad \frac{-(r-1)(a-2d)^2}{(dr+a-d)(dr-ar-d)} - 1\right)$$

or

$$(s,t) = \left(-\sqrt{\frac{r(r+1)(a-2d)}{(dr+a-d)(dr-ar-d)}}, \quad \frac{r(r-1)(a-2d)^2}{(dr+a-d)(dr-ar-d)} - 1\right)$$

for whichever case the square root is defined, prioritizing the second case if $r = 0$ and both are square. Here we take $+\sqrt{\cdot}$ to mean the non-negative square root, and $-\sqrt{\cdot}$ to mean its negation. This formulation seems to have two advantages over other ways to formulate Elligator 2 for this curve. First, the two cases are very similar, requiring only very small adjustments for the square vs. nonsquare cases. Second, the formula is invariant under the parameter involution $(a,d) \leftrightarrow (-a, d-a)$ which preserves $\mathcal{J}$ but swaps the untwisted and twisted Edwards curves. This means that if one implementor uses (a,d) and another chooses $(-a, d-a)$, then even their Elligator 2 implementations will remain compatible.

The map to $\mathcal{J}$ can easily be computed as follows:

$$\begin{aligned}
r &\leftarrow nr_0^2 \\
D &\leftarrow (dr + a - d) \cdot (dr - ar - d) \\
N &\leftarrow (r + 1) \cdot (a - 2d) \\
c, e &\leftarrow \begin{cases} +1, \quad\;\; 1/\sqrt{ND} & \text{if } ND \text{ is square} \\ -1, \, nr_0/\sqrt{nND} & \text{otherwise} \end{cases} \\
s &\leftarrow c \cdot |N \cdot e| \\
t &\leftarrow -c \cdot N \cdot (r - 1) \cdot ((a - 2d) \cdot e)^2 - 1
\end{aligned}$$

Note that if D or N is 0, the result should be some 2-torsion point ether at $(0, \pm 1)$ or at infinity. In fact, a naïve inverse square root algorithm will return $e = 0$ in this case, resulting in the point $(0, -1)$ which is indeed a 2-torsion point. Since we are quotienting out the 2-torsion group, this result is satisfactory.

To map to the Edwards or Montgomery curves, one simply applies the isogeny from $\mathcal{J}$.

To invert the Elligator 2 map, let $c = \operatorname{sign} s$ and note that

$$nr_0^2 = r = \frac{(2d - a)s^2 + c(t + 1)}{(2d - a)s^2 - c(t + 1)}$$

If $\sqrt{r/n}$ exists, then it is the inverse; otherwise, there is no inverse. Since the group in question isn't $\mathcal{J}$ but rather $\mathcal{J}/\mathcal{J}[2]$, to ensure a random sample $E^{-1}(P)$, one must add a random 2-torsion element to P.