SwiftEC: Shallue–van de Woestijne Indifferentiable Function To Elliptic Curves

Faster Indifferentiable Hashing to Most Elliptic Curves

Jorge Chávez-Saab[1,2], Francisco Rodríguez-Henríquez[1,2], and Mehdi Tibouchi[3]

[1] Computer Science Department, Cinvestav IPN, Mexico City, Mexico
[2] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, United Arab Emirates
[3] NTT Corporation, Tokyo, Japan

Abstract. Hashing arbitrary values to points on an elliptic curve is a required step in many cryptographic constructions, and a number of techniques have been proposed to do so over the years. One of the first ones was due to Shallue and van de Woestijne (ANTS-VII), and it had the interesting property of applying to essentially all elliptic curves over finite fields. It did not, however, have the desirable property of being *indifferentiable from a random oracle* when composed with a random oracle to the base field.

Various approaches have since been considered to overcome this limitation, starting with the foundational work of Brier et al. (CRYPTO 2011). For example, if $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ is the Shallue–van de Woestijne (SW) map and $\mathfrak{h}_1, \mathfrak{h}_2$ are *two* independent random oracles to $\mathbb{F}_q$, we now know that $m \mapsto f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_2(m)\big)$ is indifferentiable from a random oracle. Unfortunately, this approach has the drawback of being twice as expensive to compute than the straightforward, but not indifferentiable, $m \mapsto f\big(\mathfrak{h}_1(m)\big)$. Most other solutions so far have had the same issue: they are at least as costly as two base field exponentiations, whereas plain encoding maps like f cost only one exponentiation. Recently, Koshelev (DCC 2022) provided the first construction of indifferentiable hashing at the cost of one exponentiation, but only for a very specific class of curves (some of those with j-invariant 0), and using techniques that are unlikely to apply more broadly.

In this work, we revisit this long-standing open problem, and observe that the SW map actually fits in a one-parameter family $(f_u)_{u\in\mathbb{F}_q}$ of encodings, such that for independent random oracles $\mathfrak{h}_1, \mathfrak{h}_2$ to $\mathbb{F}_q$, $F\colon m \mapsto f_{\mathfrak{h}_2(m)}\big(\mathfrak{h}_1(m)\big)$ is indifferentiable. Moreover, on a very large class of curves (essentially those that are either of odd order or of order divisible by 4), the one-parameter family admits a rational parametrization, which let us compute F at almost the same cost as small f, and finally achieve indifferentiable hashing to most curves with a single exponentiation.

Our new approach also yields an improved variant of the Elligator Squared technique of Tibouchi (FC 2014) that represents points of arbitrary elliptic curves as close-to-uniform random strings.

Keywords: Elliptic curve cryptography · Hashing to curves · Indifferentiability · Elligator · Algebraic geometry

1 Introduction

Indifferentiable hashing to elliptic curves. Numerous cryptographic primitives and protocols constructed over elliptic curve groups involve *hashing* to an elliptic curve: they assume the existence of a public function $\mathfrak{H}$ mapping arbitrary bit strings to elliptic curve points / group elements. Moreover, the function $\mathfrak{H}$ is supposed to behave "like a random oracle". Such a functionality is required for example for many password-authenticated key exchange protocols, identity-based encryption schemes, short signature schemes, verifiable random functions, oblivious PRFs and more. It is therefore important to understand how it can be efficiently instantiated in practice, and moreover with constant-time implementations, since the data that is hashed to the curve is often sensitive and can thus be compromised by timing side-channel attacks. This problem is in fact currently the subject of an IETF standardization effort within the Crypto Forum Research Group [FHSS+22].

It became an active research topic about a decade ago, particularly after the work of Brier et al. [BCI+10], which applied Maurer et al.'s *indifferentiability* framework [MRH04] to properly formalize what it meant for $\mathfrak{H}$ to "behave like a random oracle", and proposed several constructions satisfying the required properties. The design paradigm that emerged at the time as the main approach to hashing to elliptic curve groups combines so-called *encoding functions* to the elliptic curve, which are algebraic (or piecewise algebraic) maps from the base field to the group of points on the curve, with random oracles to the base field and other sets that are "easy to hash to", as well as simple arithmetic operations on the curve.

More precisely, consider for instance[4] the problem of hashing to the subgroup $\mathbb{G}$ of cofactor h in $E(\mathbb{F}_q)$, where E is an elliptic curve defined over the finite field $\mathbb{F}_q$ and such that $E(\mathbb{F}_q)$ is cyclic of order n with generator P. Then Brier et al. [BCI+10] showed that the following construction:

$$\mathfrak{H}_{\text{slow}}(m) = [h] \cdot \Big(f\big(\mathfrak{h}_1(m)\big) + [\mathfrak{h}_2(m)]P \Big) \tag{1}$$

is *indifferentiable from a random oracle* when $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are modeled as independent random oracles to $\mathbb{F}_q$ and $\mathbb{Z}/n\mathbb{Z}$ respectively (which are easy to realize, heuristically, using bitstring-valued hash functions) and $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ is a mapping (the encoding function) satisfying mild conditions. This means that whenever[5] a cryptographic scheme or protocol is proved secure in the random oracle model with respect to a $\mathbb{G}$-valued random oracle $\mathfrak{H}$, that random oracle can be instantiated securely with the construction $\mathfrak{H}_{\text{slow}}$.

As we have mentioned, the construction above requires a suitable *encoding function* $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$. A number of candidates were known at the time for

[4] The general case of a non-cyclic $E(\mathbb{F}_q)$ can be treated similarly. We refer to Brier et al. [BCI+10] for details.

[5] Technically, this holds in the case of *single-stage* security games, as clarified by Ristenpart et al. [RSS11]. This limitation is rarely of concern in our context.

various classes of elliptic curves, such as those of Shallue and van de Woestijne [SvdW06], Ulas [Ula07] or Icart [Ica09], and many more have been proposed since [KLR10,FT10,Far11,FT12,FJT13,BHKL
All of them can be computed in constant time at the cost of one full size exponentiation in $\mathbb{F}_q$ (typically a square root or cube root computation), which dominates the complexity, plus a few other less costly operations in the field, like multiplications, inversions and Jacobi symbol computations.

In contrast, the second term of $\mathfrak{H}_{\text{slow}}$ is a full-size *scalar multiplication* over the curve, which typically exceeds the computationally cost of a field exponentiation by a factor of 10 or more depending on base field size and curve arithmetic. This makes $\mathfrak{H}_{\text{slow}}$ a fairly inefficient construction.

To alleviate this issue, Brier et al. also proved that the following construction is also indifferentiable from a random oracle:

$$\mathfrak{H}_{\text{square}}(m) = [h] \cdot \Big(f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_1(m)\big) \Big) \tag{2}$$

when $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are modeled as independent random oracles to $\mathbb{F}_q$, and when f is specifically Icart's function. The result was later extended by Farashahi et al. [FFS+13], who showed that basically all of the known encoding functions f could also be plugged into that construction. This provides indifferentiable hashing to arbitrary elliptic curves at the cost of essentially *two* base fields exponentiations.

On the other hand, in certain primitives and protocols proved secure with respect to a $\mathbb{G}$-valued random oracle $\mathfrak{H}$, one can show that $\mathfrak{H}$ can be securely instantiated using the following simpler construction:

$$\mathfrak{H}_{\text{non-unif}}(m) = [h] \cdot f\big(\mathfrak{h}(m)\big) \tag{3}$$

where $\mathfrak{h}$ is modeled as a random oracle to $\mathbb{F}_q$. This construction is not nearly as well-behaved as (2). In fact, f usually only reaches a fraction of the points on $E(\mathbb{F}_q)$, and induces a non-uniform distribution over its image, so that $\mathfrak{H}_{\text{non-unif}}$ can typically be efficiently distinguished from a random oracle, and in particular it is *not* indifferentiable in the sense discussed so far. Nevertheless, certain primitives and protocols do not require the full strength of indifferentiability, and $\mathfrak{H}_{\text{non-unif}}$ is sometimes sufficient to let their security proofs go through.

A rough idea of why this happens is that, in a random oracle proof of security, the simulator generally want to program the random oracle by setting the hash of some message m to a value Q, but that point Q itself can usually be anything depending on some randomness. So assuming that $h = 1$, the simulator might typically want to set $\mathfrak{H}(m)$ to $Q = [r] \cdot P$ for some random r, say. Now if $\mathfrak{H}$ is defined in the protocol using a construction like (3), the simulator would pick a random r and set $\mathfrak{h}(m)$ to one of the preimages $u \in f^{-1}(P)$ if $P \in f(\mathbb{F}_q)$. If however P is not in the image of f, the simulator would pick another random r and try again.

Therefore, construction (3), while less general and well-behaved than (2), is sometimes good enough for security at half the computational cost. This is a substantial difference in terms of efficiency that practitioners may be sensitive to, so much so that *both* of these constructions are in fact proposed in the current

IETF draft [FHSS+22]. Construction (3), however, comes with the caveats that applications using it "SHOULD carefully analyze the security implications of nonuniformity", and that "cryptographic protocols whose security analysis relies on a random oracle that outputs points with a uniform distribution MUST NOT" use it. This results in the somewhat unfortunate situation that implementers have to choose between two approaches for implementing hashing to elliptic curves: one which is secure in all cases but slower, and one which is faster but requires a careful analysis to ascertain that it does not fully compromise the security of the scheme.

The quest for fast indifferentiable hashing. Ideally, one would prefer to have the best of both worlds: indifferentiable hashing at the cost of a single exponentiation in the base field instead of two. Obtaining this for general elliptic curves is a long-standing open problem.

In special cases, solutions exist: this is particularly the case for supersingular curves of j-invariant 0 and 1728, for which it has long been known [BF01,FT10] that an "almost bijective" encoding function f exists; it is then easy to check that plugging that f into construction (3) does achieve indifferentiability. Unfortunately, those types of supersingular curves, which were popular to reach the 80-bit security level in pairing applications in the early 2000s, are no longer used today due to exceedingly large parameters at higher security levels. Moreover, there are strong reasons to believe that almost bijective encodings cannot exist for general elliptic curves [Tib14b].

Progress towards adressing the general open problem was made by Tibouchi and Kim [TK17], who extended the statistical results of Farashahi et al., and established in particular that, asymptotically, it was possible to achieve indifferentiable hashing at a cost of less than two exponentiations by tweaking construction (1) with a random oracle $\mathfrak{h}_2$ mapping to a short interval. That result is mostly of theoretical significance, however, since it requires very large base fields to provide meaningful error bounds.

Recently, Koshelev [Kos22] made a practically significant advance, by showing that indifferentiable hashing at the cost of a single exponentiation was possible for certain *ordinary* curves of j-invariant 0 over suitable base fields. This is still a negligible fraction of all elliptic curves, but it is practically relevant since it includes pairing-friendly curves like some of the BLS12 used today. Koshelev's approach is also the first one considered in the last decade or so that substantially departs from the framework of constructions (1)–(3) above. While those earlier techniques reduce the problem of indifferentiable hashing to the encoding function $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$, which is defined over a one-dimensional domain, Koshelev bases its construction on a map $F\colon \mathbb{F}_q^2 \to E(\mathbb{F}_q)$ with a two-dimensional range. Looking back at Brier et al.'s original proof for the indifferentiability of construction (2) using Icart's encoding function, this is fairly natural (since that proof was constructed around a two-dimensional argument), but it is an important shift in perspective.

In this paper, we use a similar idea (albeit very different techniques) to settle the open problem for a large class of elliptic curves: for essentially all curves over fields $\mathbb{F}_q$ with $q \equiv 1 \pmod 3$ with either odd order or order divisible by 4 (this includes almost all elliptic curves in current use), we are able to construct a new indifferentiable hashing, which we call SwiftEC, at the cost of a single exponentiation in the base field.

Representing points as uniform random strings. A very different question, but which has been tackled using similar techniques, was introduced in Bernstein et al.'s *Elligator* paper [BHKL13]: how can one represent a uniform point on $E(\mathbb{F}_q)$ in a public way as a close to uniform random bit string. The stated goal was to achieve a form of steganography for censorship circumvention. Indeed, network traffic containing points on a certain elliptic curve (e.g. public keys for encryption or signature) represented in usual ways (either as full coordinates (x, y), in compressed form $(x, \operatorname{sgn} y)$ or in x-only form) can be easily distinguished from random, which may lead to automated traffic interruption or targeted surveillance.

As a countermeasure, Bernstein et al. suggested to use an encoding function $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ with the property that it maps an interval $I \subset \mathbb{F}_q$ of length $\approx q/2$ *injectively* into $E(\mathbb{F}_q)$. Then, any point in $f(I)$ can be represented by its unique preimage under f in I. In particular, if q is close to a power of two, this readily gives a simple representation of random elements in $f(I) \subset E(\mathbb{F}_q)$ as uniform random bit strings (and when q is far from a power of two, it suffices to represent elements of I as uniform random bit strings, which can be easily done by expanding the representation and introducing randomness).

This approach has two drawbacks. First, suitable encodings f that are injective over a large interval are hard to construct, and only known for limited families of elliptic curves [Far11,FJT13,BHKL13], all of order divisible by 3 or 4 (and hence not including curves of prime order, for example). Second, one needs to address the issue of points falling outside $f(I)$. Since the goal is to represent *random* points on $E(\mathbb{F}_q)$ as bit strings, the assumption is that in the cryptographic protocol under consideration, the point to represent is obtained by some sort of random process, and it is possible to use rejection sampling until reaching $f(I)$. Since the image size covers roughly half of all points on the curve, this will require about two iterations on average, often an acceptable cost. However, if the process generating the point is expensive, rejecting may be less than ideal.

Tibouchi's Elligator Squared paper [Tib14a] addressed these shortcomings by, in essence, applied construction (2) above "in reverse". One of the key properties that makes construction (2) an indifferentiable hash function is the fact that, for an encoding function $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$, the following map:

$$\begin{aligned} f^{\otimes 2}\colon \mathbb{F}_q^2 &\to E(\mathbb{F}_q) \\ (u, v) &\mapsto f(u) + f(v) \end{aligned} \tag{4}$$

induces a close-to-uniform distribution on its image. In particular, a uniformly random preimage of a uniformly random point in $E(\mathbb{F}_q)$ is close to uniform

in $\mathbb{F}_q^2$. This provides a simple solution to the point representation problem that works for general elliptic curves and can represent all points, avoiding the need for rejection sampling inside the protocol to reach a particular subset of the curve. However, representation size is about twice as large as Elligator (a drawback partially addressed in subsequent work [TK17]) and the representation function, computing uniformly random preimages under $f^{\otimes 2}$, is also somewhat more complicated and costly than that of Elligator.

Basically, to compute a random preimage of $P \in E(\mathbb{F}_q)$, one picks a uniform $v \in \mathbb{F}_q$ and computes u as a preimage of $P - f(v)$. However, rejection sampling is necessary to ensure the uniformity of the distribution, which requires multiple iterations, each of them evaluating the function f (at a cost of a field exponentiation each).

In this paper, as a by-product of our new SwiftEC construction, we also obtain ElligatorSwift, a much faster variant of Elligator Squared over all the curves over which SwiftEC is defined. The idea is that fully computing the underlying encoding in the forward direction becomes unnecessary, saving many field exponentiations in the process.

Contributions and technical overview. The starting point of our work is to revisit the first construction of an encoding function to general elliptic curves, originally due to Shallue and van de Woestijne [SvdW06]. We observe that that construction actually had a number of interesting properties that have not been considered so far, and that we manage to build upon with suitable additional analysis. To describe them, we need to first recall a few facts about the Shallue–van de Woestijne encoding itself.

Given an elliptic curve $E\colon y^2 = g(x)$ ($g(x) = x^3 + ax + b$) over a finite field $\mathbb{F}_q$ of characteristic ≥ 5, Shallue and van de Woestijne construct a certain algebraic surface S in the affine space over $\mathbb{F}_q$ together with three rational functions x_1, x_2, x_3 such that the product $g(x_1)g(x_2)g(x_3)$ is a square. This means in particular that, when evaluated at any point P of $S(\mathbb{F}_q)$ (outside of the locus of poles), at least one of $x_1(P)$, $x_2(P)$ or $x_3(P)$ must be the x-coordinate of a point in $E(\mathbb{F}_q)$. Indeed, the product $g\big(x_1(P)\big)g\big(x_2(P)\big)g\big(x_3(P)\big)$ is a square in $\mathbb{F}_q$, and since the product of three nonsquares in $\mathbb{F}_q$ is a nonsquare, at least one of the factors must be square, yielding the x-coordinate of a point in $E(\mathbb{F}_q)$. Based on that, we can define an encoding function from $S(\mathbb{F}_q)$ to $E(\mathbb{F}_q)$ simply by mapping a point P to one of the points of x-coordinate $x_i(P)$ that works (selecting the index i and the sign of the y-coordinate in a predetermined way).

The second step of the construction is to note that the specific surface S under consideration can in fact be seen as a one-parameter family of conics over $\mathbb{F}_q$. Based on that, Shallue and van de Woestijne fix the value of the parameter, obtain a single non-degenerate conic over $\mathbb{F}_q$, and use the fact that such a conic always admits a rational parametrization to obtain a map $\mathbb{F}_q \to S(\mathbb{F}_q)$ to the chosen conic. Composing with the previous map finally gives an encoding $\mathbb{F}_q \to E(\mathbb{F}_q)$ as desired, which can be used in constructions (1)–(3) above for hashing,

and in the Elligator Squared framework: this is what is usually known as the Shallue–van de Woestijne encoding.

Our contributions rely on two novel observations regarding that original construction:

- first, for a large class of elliptic curves E which we characterize in details, the surface S regarded as a family of conics actually admits a global, two-parameter parametrization over $\mathbb{F}_q$. This means that one can effectively construct a rational map $\mathbb{F}_q^2 \to S(\mathbb{F}_q)$ that is essentially a bijection. This result is obtained using techniques due to van Hoeij and Cremona [vHC06] classifying conics over function fields;
- second, unlike each of the maps defined by individual conics, the map from $S(\mathbb{F}_q)$ as a whole to the set $X_{E,\mathbb{F}_q}$ of elements of $\mathbb{F}_q$ which are x-coordinates on $E(\mathbb{F}_q)$ is *admissible*: it satisfies the sufficient conditions of Brier et al. [BCI+10] to construct indifferentiable hashing. The most important of those conditions is regularity: the image of a uniform point in $S(\mathbb{F}_q)$ is close to uniform in $X_{E,\mathbb{F}_q}$. We are able to establish that property by giving a precise description of the preimage of an $x \in X_{E,\mathbb{F}_q}$: it consists of the union of one algebraic curve drawn on S (the set of points P such that $x_1(P) = x$, say) and two halves of two other curves (the subset of the curves given by $x_2(P) = x$ and $x_3(P) = x$ respectively, with the condition that $g(x_1(P))$ is a nonsquare). By counting points on those curves and curve subsets, we are able to establish the required statistical properties, and deduce that $S(\mathbb{F}_q) \to X_{E,\mathbb{F}_q}$ is admissible.

Combining those two observations, we obtain, for a large, explicit class of elliptic curves E (including almost all curves in practical use), an admissible encoding $\mathbb{F}_q^2 \to X_{E,\mathbb{F}_q}$. Adding a sign bit to choose the y-coordinate on E yields an admissible encoding $F\colon \mathbb{F}_q^2 \times \{0,1\} \to E(\mathbb{F}_q)$ as well, which can be computed at the cost of a single exponentiation in $\mathbb{F}_q$ (namely, the square root computation needed to derive the y-coordinate). This has the two consequences mentioned above, over the elliptic curves E of interest:

- given a hash function $\mathfrak{h}$ modeled as a random oracle with values in $\mathbb{F}_q^2 \times \{0,1\}$ (which is easy to heuristically instantiate), the map $m \mapsto F\big(\mathfrak{h}(m)\big)$ is indifferentiable from a random oracle, and can be computed at the cost of a single exponentiation. This is the SwiftEC construction;
- given a uniform point on the curve, we can efficiently sample a uniform preimage of it under F, and this becomes a close-to-uniformly distributed element of $\mathbb{F}_q^2 \times \{0,1\}$. Since such an element is easy to represent as a uniform bit string, we thus obtain an Elligator Square-like representation technique which is much faster than Elligator Square itself, as it requires far fewer field exponentiations on average. This is the ElligatorSwift construction.

In addition, we also get indifferentiable hashing to the set $X_{E,\mathbb{F}_q}$ without any field exponentiation at all. This even faster construction, XSwiftEC, is particularly interesting in context where x-only arithmetic is feasible, such as for example BLS signatures [BLS01].

2 Preliminaries

2.1 Quadratic Residuosity

Throughout this paper, $\mathbb{F}_q$ denotes the finite field with q elements. We only consider finite fields of characteristic $\neq 2, 3$. The *quadratic character* $\chi_2 \colon \mathbb{F}_q \to \{-1, 0, 1\}$ is the map that sends 0 to 0, nonzero squares to 1 and nonzero nonsquares to -1. It is well-defined, multiplicative, and extends the unique nontrivial multicative group morphism $\mathbb{F}_q^\times \to \{-1, 1\}$. A related map is IsSquare, which sends all squares to 1 and nonsquares to 0.

When q is prime, the quadratic character coincides with the Legendre symbol, and can be computed efficiently by repeated applications of quadratic reciprocity. This can be implemented in fast constant time [Por20,Ham21,AG21], similar to the constant-time binary GCD technique of Bernstein–Yang for field inversion [BY19]. Similarly, the quadratic character over extension extension fields can be computed fast by descending to the prime field, and IsSquare can be trivially computed from χ_2.

We also fix an efficiently computable map $\operatorname{sgn} \colon \mathbb{F}_q \to \{-1, 0, 1\}$ called the "sign", with the property that $\operatorname{sgn} 0 = 0$, $\operatorname{sgn} x \neq 0$ for $x \neq 0$, and $\operatorname{sgn}(-x) = -\operatorname{sgn} x$. The choice is arbitrary, but for example over prime fields, it is customary to use the sign of an integer representative in the interval $(-q/2, q/2)$ (over extension fields, one might choose the sign of the first nonzero coefficient in some basis over the prime field).

An element $x \in \mathbb{F}_q$ which is a square has exactly two square roots (except 0 which has just one), exactly one of which is of nonnegative sign. We denote it by $\sqrt{x}$; it typically requires a single base field exponentiation to compute (although slightly faster approaches may exist over extension fields).

2.2 Elliptic curves and isogenies

An *elliptic curve* is a smooth projective curve of genus 1 endowed with a distinguished rational point. Such curves admit the definition of a point addition law, which gives the curve a structure as group variety, with the distinguished point playing the role of the group identity. Over $\mathbb{F}_q$, any elliptic curve can be written up to isomorphism in the *short Weierstrass form*:

$$E \colon y^2 = x^3 + ax + b,$$

for some $a, b \in \mathbb{F}_q$ such that the discriminant $\Delta_E := -16(4a^3 + 27b^2)$ is nonzero. On such a curve, group inverses are defined by $-(x, y) = (x, -y)$ and the points of order 2 are those with $y = 0$. When Δ_E is a square there are either zero or three points of order 2. Otherwise, there is exactly one.

We denote by $E(\mathbb{F}_q)$ the group of $\mathbb{F}_q$-rational points of E. The cardinality of this group is always $\#E(\mathbb{F}_q) = q - t + 1$ for some t bounded by $|t| \leq 2\sqrt{q}$. We say that the curve is *supersingular* if t is a multiple of the field characteristic, and otherwise the curve is *ordinary*. We focus on the case of ordinary elliptic

curves, where finding adequate and efficient encodings has long been a greater challenge.

An *isogeny* is any non-constant rational map between elliptic curves that is also a group homomorphism. Up to an isomorphism, a separable isogeny is uniquely determined by its kernel and its degree as a rational map is equal to the size of the kernel. Any isogeny $\phi\colon E \to E'$ has a dual isogeny $\hat{\phi}\colon E' \to E$ such that the composition $\hat{\phi}\circ\phi$ equals the multiplication-by-d map, where $d = \deg(\phi)$. Two curves over a finite field are isogenous if and only if they have exactly the same number of points.

2.3 Point counting and character sums

A generalization of the result above on the number of rational points of an elliptic curve is that any (absolutely irreducible) smooth curve of bounded genus over $\mathbb{F}_q$ has a number of points over $\mathbb{F}_q$ close to q. More precisely, the following celebrated result holds:

Lemma 1 (Hasse–Weil bound). *For any smooth projective absolutely irreducible curve $X/\mathbb{F}_q$ of genus g, we have:*

$$\left|\#X(\mathbb{F}_q) - (q+1)\right| \leq 2g\sqrt{q}.$$

For curves of bounded degree, the number of points at infinity is also bounded, and we thus get a bound of the form $\#X^{\mathrm{aff}}(\mathbb{F}_q) = q + c\sqrt{q} + O(1)$ ($|c| \leq 2g$) on the number of affine points on X.

A related result concerns character sums on such curves. Let χ be a multiplicative character of $\mathbb{F}_q$ (a group homomorphism $\mathbb{F}_q^\times \to \mathbb{C}^\times$ extended by 0 at 0), and $f \in \mathbb{F}_q(X)$ a rational function on the curve X. We consider the following character sum:

$$W(X,\chi,f) = \sum_{\substack{P\in X(\mathbb{F}_q)\\ f(P)\neq\infty}} \chi\big(f(P)\big).$$

Using the Bombieri–Weil methodology, Perret [Per91] proves the following bound. See also [CM00,TK17].

Lemma 2 (Perret). *Let X be a smooth projective absolutely irreducible curve of genus g over $\mathbb{F}_q$, χ a nontrivial multiplicative character of order $m|q-1$, and $f \in \mathbb{F}_q(X)$ a rational function which is not a perfect m-th power in $\overline{\mathbb{F}}_q(X)$. The character sum $W(X,\chi,f)$ can be bounded as:*

$$\left|W(X,\chi,f)\right| \leq (2g - 2 + 2\deg f)\sqrt{q}.$$

2.4 Quadratic residuosity over function fields

Many results of classical arithmetic over $\mathbb{Q}$ and number fields have analogues over function fields. This is in particular the case for quadratic reciprocity. We

recall some of the relevant results below. An exhaustive treatment is provided in Rosen's textbook [Ros02, pp. 23-31].

For a fixed monic irreducible polynomial $f \in \mathbb{F}_q[t]$, we define the quadratic residue symbol $\left(\frac{g}{f}\right)_2$ for any $g \in \mathbb{F}_q[t]$ as the image of g under the quadratic character of the finite field $\mathbb{F}_q[t]/(f)$. In other words:

$$\left(\frac{g}{f}\right)_2 = \begin{cases} 0 & \text{if } f \text{ divides } g; \\ 1 & \text{if } g \text{ is coprime to } f \text{ and a square modulo } f; \\ -1 & \text{if } g \text{ is coprime to } f \text{ and a nonsquare modulo } f. \end{cases}$$

We then extend this symbol to not necessarily irreducible f's by multiplicativity, similarly to how the Jacobi symbol extends the Legendre symbol. If $f = \alpha f_1^{e_1} \cdots f_n^{e_n}$ with $\alpha \in \mathbb{F}_q^\times$ and the f_i irreducible, we let:

$$\left(\frac{g}{f}\right)_2 = \prod_{i=1} n \left(\frac{g}{f_i}\right)_2.$$

Note that the symbol does not depend on the leading coefficient $\mathrm{lc}(f) = \alpha$ of f.

Lemma 3. *The quadratic residue symbol has the following properties.*

- *If* $g_1 \equiv g_2 \pmod f$, $\left(\frac{g_1}{f}\right)_2 = \left(\frac{g_2}{f}\right)_2$.
- $\left(\frac{g_1 g_1}{f}\right)_2 = \left(\frac{g_1}{f}\right)_2 \left(\frac{g_2}{f}\right)_2$.
- $\left(\frac{g}{f_1 f_2}\right)_2 = \left(\frac{g}{f_1}\right)_2 \left(\frac{g}{f_2}\right)_2$.
- $\left(\frac{g}{f}\right)_2 \neq 0$ *if and only if* f *and* g *are coprime.*
- *If* g *is a nonzero square modulo* f*, then* $\left(\frac{g}{f}\right)_2 = 1$ *(but the converse does not need to hold).*

Furthermore, it satisfies the following law of quadratic reciprocity. For $f, g \in \mathbb{F}_q[t]$ *coprime and nonzero, it holds that:*

$$\left(\frac{g}{f}\right)_2 \left(\frac{f}{g}\right)_2 = (-1)^{\frac{q-1}{2} \deg f \deg g} \mathrm{lc}(f)^{\frac{q-1}{2} \deg g} \mathrm{lc}(g)^{\frac{q-1}{2} \deg f}.$$

2.5 Statistical notions

For $\mathscr{D}$ a probability distribution on a finite set S, we write $\Pr[s \leftarrow \mathscr{D}]$ for the probability assigned to the singleton $\{s\} \subset S$ by $\mathscr{D}$. The uniform distribution on S is denoted by $\mathscr{U}_S$ (or just $\mathscr{U}$ if the context is clear).

Definition 1 (Statistical distance). *Let* $\mathscr{D}$ *and* $\mathscr{D}'$ *be two probability distributions on a finite set* S*. The* statistical distance *between them is defined as the* ℓ_1 *norm:*

$$\Delta_1(\mathscr{D}, \mathscr{D}') = \frac{1}{2} \sum_{s \in S} \left| \Pr[s \leftarrow \mathscr{D}] - \Pr[s \leftarrow \mathscr{D}'] \right|.$$

We simply denote by $\Delta_1(\mathscr{D})$ the statistical distance between $\mathscr{D}$ and $\mathscr{U}_S$:

$$\Delta_1(\mathscr{D}) = \frac{1}{2}\sum_{s\in S}\left|\Pr[s \leftarrow \mathscr{D}] - \frac{1}{\#S}\right|,$$

and say that $\mathscr{D}$ is ε-statistically close to uniform *when $\Delta_1(\mathscr{D}) \leq \varepsilon$. When $\Delta_1(\mathscr{D})$ is negligible, we simply say than $\mathscr{D}$ is* statistically close to uniform.

Definition 2 (Pushforward). *Let S, T be two finite sets and F any mapping from S to T. For any probability distribution $\mathscr{D}_S$ on S, we can define the* pushforward *$F_*\mathscr{D}_S$ of $\mathscr{D}_S$ by F as the probability distribution on T such that sampling from $F_*\mathscr{D}_S$ is equivalent to sampling a value $s \leftarrow \mathscr{D}_S$ and returning $F(s)$. In other words:*

$$\Pr\left[t \leftarrow F_*\mathscr{D}_S\right] = \Pr\left[s \leftarrow \mathscr{D}_S;\ t = F(s)\right] = \mu_S\big(F^{-1}(t)\big) = \sum_{s\in F^{-1}(t)} \Pr[s \leftarrow \mathscr{D}_S],$$

where μ_S is the probability measure defined by $\mathscr{D}_S$.

Definition 3 (Regularity). *Let S, T be two finite sets and F any mapping from S to T. We say that F is* ε-regular *when $F_*\mathscr{U}_S$ is ε-close to the uniform distribution. We may omit ε if it is negligible.*

2.6 Admissible encodings

In their work on the construction of indifferentiable hashing to elliptic curves, Brier et al. [BCI+10] define the notion of an *admissible* map $F\colon S \to R$ between two sets. The definition, which generalizes an early notion introduced by Boneh and Franklin [BF01], is as follows.

Definition 4 (Admissible encoding). *A function $F\colon S \to R$ between finite sets is an ε-admissible encoding if it satisfies the following properties:*

Computable: *F is computable in deterministic polynomial time.*
Regular: *F is ε-regular (in the sense of the previous section).*
Samplable: *there is an efficient randomized algorithm $\mathscr{I}\colon R \to S \sqcup \{\perp\}$ such that for any $r \in R$, $\mathscr{I}(r)$ induces a distribution that is ε-statistically close to the uniform distribution in $F^{-1}(r)$.*

F is an admissible encoding if it is ε-admissible for some negligible ε.

That notion satisfies the suitable properties such that, given an S-valued random oracle $\mathfrak{h}$, the composition $F\circ\mathfrak{h}$ is indifferentiable from a R-valued random oracle.

Moreover a similar results holds for arbitrary compositions of admissible functions (even though admissibility need not be preserved under composition). Namely, if $F_i\colon S_i \to S_{i-1}$ are admissible encodings for $i = 1, \ldots, n$, then it also holds that, given an S_n-valued random oracle $\mathfrak{h}$, the composition $F_1\circ\cdots\circ F_n\circ\mathfrak{h}$ is indifferentiable from a S_0-valued random oracle (even though it does not always hold that $F_1 \circ \cdots \circ F_n$ is admissible).

3 The SW Encoding Family

In their seminal ANTS–VII paper [SvdW06], Shallue and van de Woestijne constructed the first encoding function to arbitrary elliptic curves. In this section, we give a description of that construction (restricted for simplicity to base fields of characteristic ≥ 5) that is slightly different but essentially equivalent to the original one, and then we state new properties of that construction.

In the entire section, we fix an elliptic curve $E\colon y^2 = x^3 + ax + b$ over the finite field $\mathbb{F}_q$ (q prime power not divisible by 2 or 3), and denote by $X_{E,\mathbb{F}_q}$ the subset of $\mathbb{F}_q$ consisting of x-coordinates of poins in $E(\mathbb{F}_q)$; in other words:

$$X_{E,\mathbb{F}_q} = \big\{x \in \mathbb{F}_q \ ; \ \exists y,\ (x,y) \in E(\mathbb{F}_q)\big\}.$$

3.1 Construction of the Shallue–van de Woestijne encoding

Let g and h be the polynomials over $\mathbb{F}_q$ defined by:

$$g(u) = u^3 + au + b \quad \text{and} \quad h(u) = 3u^2 + 4a.$$

The starting point of the Shallue–van de Woestijne construction is the construction of a rational map $\psi\colon S \to V$ from the following quasi-affine surface in the (x, y, u) affine space:

$$S\colon x^2 + h(u)y^2 = -g(u), \quad y \neq 0 \tag{5}$$

to the following threefold in the (x_1, x_2, x_3, z) affine 4-dimensional space:

$$V\colon z^2 = g(x_1)g(x_2)g(x_3).$$

The rational map ψ is given by the following explicit equations and clearly defined everywhere on S:

$$\begin{aligned} x_1 &= \frac{x}{2y} - \frac{u}{2} \quad & x_2 &= -\frac{x}{2y} - \frac{u}{2} \\ x_3 &= u + 4y^2 & z &= \frac{g(u+y^2)}{y} \cdot R\Big(u, \frac{x}{2y} - \frac{u}{2}\Big) \end{aligned} \tag{6}$$

where $R(u, v) = u^2 + uv + v^2 + a$. When referring to a point P on S, we will denote by $x_1(P)$, $x_2(P)$, $x_3(P)$ and $z(P)$ the corresponding coordinates of $\psi(P)$ in V. In particular, this defines x_1, x_2, x_3 and z as rational functions on the surface.

A remarkable property of the threefold V is that for any point $(x_1, x_2, x_3, z) \in V(\mathbb{F}_q)$, at least one of the three values x_1, x_2, x_3 must be in $X_{E,\mathbb{F}_q}$. Indeed, $g(x_1)g(x_2)g(x_3)$ is a square in $\mathbb{F}_q$, so by multiplicativity of the quadratic character, they cannot be all nonsquares (and in fact, there must be exactly one or three squares among them, except possibly when $z = 0$).

As a result, one can therefore map points on $S(\mathbb{F}_q)$ to $X_{E,\mathbb{F}_q}$ by first mapping to $V(\mathbb{F}_q)$ with ψ, and then selecting one of the coordinates x_1, x_2, x_3 in a prescribed order. For example, in this paper we will consider the following map:

$$F_0\colon S(\mathbb{F}_q) \to X_{E,\mathbb{F}_q}$$

$$P \mapsto \begin{cases} x_3(P) & \text{if } g\big(x_3(P)\big) \text{ is a square;} \\ x_2(P) & \text{if } g\big(x_3(P)\big) \text{ is not a square but } g\big(x_2(P)\big) \text{ is;} \\ x_1(P) & \text{if neither } g\big(x_3(P)\big) \text{ or } g\big(x_2(P)\big) \text{ are squares.} \end{cases} \tag{7}$$

Note that $F_0(P)$ is very efficient to compute from the coordinates (x, y, u) of P using the formulas of (6) and a few quadratic character computations. In particular, it requires no field exponentiation.

Of course, once we have an element $\bar{x} \in X_{E,\mathbb{F}_q}$, it is easy to deduce a point in $E(\mathbb{F}_q)$: simply compute a square root of $g(\bar{x})$ to get the y-coordinate up to sign. Since we prefer to select the sign separately, we define the following extended map to $E(\mathbb{F}_q)$ which takes an additional input bit b:

$$\begin{aligned} F_0^+ \colon S(\mathbb{F}_q) \times \{0,1\} &\to E(\mathbb{F}_q) \\ (P, b) &\mapsto \Big(F_0(P),\ (-1)^b \sqrt{g\big(F_0(P)\big)}\Big). \end{aligned} \tag{8}$$

The construction offers a way to map to $E(\mathbb{F}_q)$ provided that one can construct rational points on the surface S itself, which may not be a priori obvious. Fortunately, as seen from equation (5), each of the curves S_{u_0} on S obtained by fixing u to some $u_0 \in \mathbb{F}_q$ are simply conics over $\mathbb{F}_q$, with equations:

$$x^2 + h(u_0)y^2 = -g(u_0), \quad y \neq 0.$$

Now, a conic over $\mathbb{F}_q$ always admits a rational parametrization. Therefore, we can construct a map $\mathbb{F}_q \to S_{u_0}(\mathbb{F}_q)$ that can then be composed with F_0^+ to obtain an encoding function $F_{0,u_0} \colon \mathbb{F}_q \to X_{E,\mathbb{F}_q}$ (and similarly to $E(\mathbb{F}_q)$). This is basically the approach taken in the original paper of Shallue and van de Woestijne [SvdW06].

Note that obtaining the parametrization of the conic S_{u_0} for a fixed u_0 requires an a priori costly precomputation (it requires finding a point on the conic, typically by trial-and-error: this costs a square root, and a number of quadratic character computations that is hard to bound uniformly). Therefore, while it may be tempting to try and define a two-parameter map $\mathbb{F}_q^2 \to X_{E,\mathbb{F}_q}$ by $(t, u) \mapsto F_{0,u}(t)$, this is not usually workable for hashing purposes, since a new parametrization would have to be computed for any new input u.

Nevertheless, we show in the remainder of this section that the maps F_0 and F_0^+ on the surface $S(\mathbb{F}_q)$ *as a whole* have nice statistical properties, and it would therefore be beneficial to overcome the difficulty of efficiently parametrizing it. That problem will then be addressed, at least for a large class of elliptic curves E, in Section 4 below.

3.2 Geometry of the SW family

For a fixed element $\bar{x} \in X_{E,\mathbb{F}_q}$, we now want to describe the set of points in $S(\mathbb{F}_q)$ that map to x under the encoding F_0 of (7). By the previous description of the encoding, this is the union of three disjoint sets:

$$F_0^{-1}(\bar{x}) = C_{\bar{x}}^{(3)}(\mathbb{F}_q) \sqcup C_{\bar{x}}^{(2)}(\mathbb{F}_q)^+ \sqcup C_{\bar{x}}^{(1)}(\mathbb{F}_q)^+,$$

where $C_{\bar{x}}^{(i)}$ are algebraic curves on S defined by the condition that $x_i = \bar{x}$ $(i = 1, 2, 3)$ and $C_{\bar{x}}^{(i)}(\mathbb{F}_q)^+$ is the subset of $C_{\bar{x}}^{(i)}(\mathbb{F}_q)$ under the condition that $g\big(x_j(P)\big)$ is not a square for $j \neq i$. Note that since there are always exactly only 1 or 3 squares, it suffices to define

$$C_{\bar{x}}^{(1)}(\mathbb{F}_q)^+ := \{P \in C_{\bar{x}}^{(1)}(\mathbb{F}_q);\ x_2(P) \text{ not a square}\}$$

$$C_{\bar{x}}^{(2)}(\mathbb{F}_q)^+ := \{P \in C_{\bar{x}}^{(2)}(\mathbb{F}_q);\ x_1(P) \text{ not a square}\}$$

We would like to count the number of points in each of these sets. The first step is to understand the geometry of the curves $C_{\bar{x}}^{(i)}$. It is easy to see that, for a generic $\bar{x}$, they are hyperelliptic curves of genus 2.

Consider for example $C_{\bar{x}}^{(3)}$. It is given by the equations (*cf.* (6)):

$$u + 4y^2 = \bar{x} \quad \text{and} \quad x^2 + h(u)y^2 = -g(u).$$

Eliminating $u = \bar{x} - 4y^2$ between those two equations, we see that that $C_{\bar{x}}^{(3)}$ is isomorphic to the curve in the (y, x) affine plane given by the equation:

$$x^2 = -g(\bar{x} - 4y^2) - h(\bar{x} - 4y^2)y^2.$$

The right-hand side is a polynomial of degree 6 in y, namely:

$$16y^6 - 24\bar{x}y^4 + 9\bar{x}^2y^2 - g(\bar{x}),$$

whose discriminant is a polynomial of degree exactly 11 in $\bar{x}$ (or exactly 9 if $a = 0$). We thus get that $C_{\bar{x}}^{(3)}$ is a hyperelliptic curve of genus 2, except for at most 11 points $\bar{x}$. Other than for those exceptional points, we have:

$$\#C_{\bar{x}}^{(3)}(\mathbb{F}_q) = q + c_3\sqrt{q} + O(1), \quad \text{for some } c_3 \text{ such that } |c_3| \leq 4.$$

by the Hasse–Weil bound. Note that the $O(1)$ term comes from the fact that we consider the affine situation rather than the projective one, and we could easily provide an explicit bound for it, but this is typically not of interest for cryptographic applications.

Similarly, $C_{\bar{x}}^{(2)}$ is given by the equations:

$$-\frac{x}{2y} - \frac{u}{2} = \bar{x} \quad \text{and} \quad x^2 + h(u)y^2 = -g(u).$$

Eliminating $x = -y(u + 2\bar{x})$ between those two equations, we see that that $C_{\bar{x}}^{(2)}$ is isomorphic to the curve in the (u, y) affine plane given by the equation:

$$y^2\big[(u + 2\bar{x})^2 + h(u)\big] = -g(u),$$

which is again isomorphic to the curve in the (u, v) affine plane, $v = y\big[(u + 2\bar{x})^2 + h(u)\big]$, of equation:

$$v^2 = -g(u) \cdot \big[(u + 2\bar{x})^2 + h(u)\big].$$

The right-hand side is a polynomial of degree 5 in u, namely:

$$-4\big(u^5 + \bar{x}u^4 + (\bar{x}^2 + 2a)u^3 + (a\bar{x} + b)u^2 + (a\bar{x}^2 + b\bar{x} + a^2)u + b(\bar{x}^2 + a)\big),$$

and its discriminant is always of degree 14 in $\bar{x}$ (the degree 14 coefficient is $2^{16} \cdot 3 \cdot (4a^3 + 27b^2) \neq 0$). Thus, $C^{(2)}_{\bar{x}}$ is a hyperelliptic curve of genus 2, except for at most 14 points $\bar{x}$. Other than for those exceptional points, we therefore have:

$$\#C^{(2)}_{\bar{x}}(\mathbb{F}_q) = q + c_2\sqrt{q} + O(1) \quad \text{for some } c_2 \text{ such that } |c_2| \leq 4$$

by the Hasse–Weil bound.

A similar computation gives the same result for $C^{(1)}_{\bar{x}}$, given by the equations:

$$\frac{x}{2y} - \frac{u}{2} = \bar{x} \quad \text{and} \quad x^2 + h(u)y^2 = -g(u).$$

Indeed, eliminating $x = y(u + 2\bar{x})$ between those two equations shows that $C^{(1)}_{\bar{x}}$ is also isomorphic to the curve in the (u, y) affine plane given by the equation:

$$y^2\big[(u + 2\bar{x})^2 + h(u)\big] = -g(u),$$

the same as above. It therefore holds again that, except for at most 14 points $\bar{x}$, $C^{(2)}_{\bar{1}}$ is a hyperelliptic curve of genus 2, and:

$$\#C^{(1)}_{\bar{x}}(\mathbb{F}_q) = q + c_1\sqrt{q} + O(1) \quad \text{for some } c_1 \text{ such that } |c_1| \leq 4$$

by the Hasse–Weil bound.

It remains to evaluate the cardinality of the subsets $C^{(i)}_{\bar{x}}(\mathbb{F}_q)^+ \subset C^{(i)}_{\bar{x}}(\mathbb{F}_q)$ for $i = 1, 2$. One can do so in various ways, but the simplest is probably to relate them to character sums. Consider for example the following character sum on $C^{(1)}_{\bar{x}}$:

$$W_1 := W\big(C^{(1)}_{\bar{x}}, \chi_2, g \circ x_2\big) = \sum_{P \in C^{(1)}_{\bar{x}}(\mathbb{F}_q)} \chi_2\Big(g\big(x_2(P)\big)\Big),$$

where χ_2 is the quadratic multiplicative character of $\mathbb{F}_q$. The term $\chi_2\Big(g\big(x_2(P)\big)\Big)$ is equal to -1 if $g\big(x_2(P)\big)$ is not a square in $\mathbb{F}_q$, which is exactly when $P \in C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+$. Moreover, it is otherwise equal to 1 (for points outside $C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+$ such that $x_2(P) \neq 0$) or 0 (for points outside $C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+$ such that $x_2(P) = 0$). As a result, we have:

$$\begin{aligned} W_1 &= (-1) \cdot \#C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+ + 1 \cdot (\#C^{(1)}_{\bar{x}}(\mathbb{F}_q) - \#C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+ - N_0) + 0 \cdot N_0 \\ &= \#C^{(1)}_{\bar{x}}(\mathbb{F}_q) - 2 \cdot \#C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+ - N_0, \end{aligned}$$

where $N_0 = O(1)$ is the number of points in $C^{(1)}_{\bar{x}}(\mathbb{F}_q)$ such that $x_2(P) = 0$. This gives:

$$\#C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+ = \frac{1}{2}\#C^{(1)}_{\bar{x}}(\mathbb{F}_q) - \frac{W_1}{2} + O(1) = \frac{q}{2} + \frac{c_1}{2}\sqrt{q} - \frac{W_1}{2} + O(1),$$

where the $O(1)$ term accounts both for N_0 and for the fact that we consider an affine situation instead of a projective one.

Then, by the character sum estimate of Lemma 2, we have:

$$|W_1| \leq \big(4-2+2\deg(g\circ x_2)\big)\sqrt{q}+O(1) = (2+2\cdot 3\cdot 2)\sqrt{q}+O(1) = 14\sqrt{q}+O(1)$$

since $x_2 = u - \bar{x}$ on $C^{(1)}_{\bar{x}}$ is a rational function of degree 2. It then follows that:

$$\#C^{(1)}_{\bar{x}}(\mathbb{F}_q)^+ = \frac{q}{2} + c_1^+\sqrt{q} + O(1) \quad \text{for some } c_1^+ \text{ such that } |c_1^+| \leq \frac{4+14}{2} = 9.$$

Obviously, the exact same argument applies to $C^{(2)}_{\bar{x}}$, yielding:

$$\#C^{(2)}_{\bar{x}}(\mathbb{F}_q)^+ = \frac{q}{2} + c_2^+\sqrt{q} + O(1) \quad \text{for some } c_2^+ \text{ such that } |c_2^+| \leq 9.$$

Combining all the previous estimates, we finally obtain the following result.

Theorem 1. *For all $\bar{x} \in X_{E,\mathbb{F}_q}$ except at most 39 of them, the number of preimages of $\bar{x}$ under the F_0 map of equation (7) is close to $2q$, and the difference is bounded as:*

$$\big|\#F_0^{-1}(\bar{x}) - 2q\big| \leq 22\sqrt{q} + O(1).$$

Proof. Indeed, except for the at most $11 + 14 + 14 = 39$ exceptional points mentioned above, we have:

$$\#F_0^{-1}(\bar{x}) = \Big(1 + \frac{1}{2} + \frac{1}{2}\Big)q + \big(c_1^+ + c_2^+ + c_3\big)\sqrt{q} + O(1)$$

and since $\big|c_1^+ + c_2^+ + c_3\big| \leq 4 + 9 + 9 = 22$, the result follows.

3.3 The SW family is admissible

Using Theorem 1, we are now in a position to prove that the encoding function F_0 is *admissible* in the sense of Section 2.6. The main step in doing so is to prove that it is *regular*.

Lemma 4. *The map $F_0\colon S(\mathbb{F}_q) \to X_{E,\mathbb{F}_q}$ of equation (7) is ε-regular for $\varepsilon = \big(6 + o(1)\big)q^{-1/2}$.*

Proof. Let $\Delta = \Delta_1\big((F_0)_*\mathscr{U}_{S(\mathbb{F}_q)}\big)$ be the statistical distance between the distribution induced by F_0 on $X_{E,\mathbb{F}_q}$ and the uniform distribution. By definition, we have:

$$\Delta = \frac{1}{2}\sum_{\bar{x}\in X_{E,\mathbb{F}_q}} \left|\frac{\#F^{-1}(\bar{x})}{\#S(\mathbb{F}_q)} - \frac{1}{\#X_{E,\mathbb{F}_q}}\right|.$$

Now for each element $\bar{x} \in X_{E,\mathbb{F}_q}$, there are exactly two points of $E(\mathbb{F}_q)$ with x-coordinate equal to $\bar{x}$, except if $g(\bar{x}) = 0$, in which case there is exactly one

(and this happens for at most three values of $\bar{x}$). Taking the point at infinity into account, we therefore get:

$$\#X_{E,\mathbb{F}_q} = \frac{1}{2}\#E(\mathbb{F}_q) + O(1) = \frac{q}{2} + c_E\sqrt{q} + O(1) \quad \text{for some } c_E \text{ with } |c_E| \leq 1$$

by yet another application of the Hasse–Weil bound. Up to sign, the constant c_E is half the normalized Frobenius trace of E.

Moreover, $S(\mathbb{F}_q)$ is the disjoint union of the various affine conics $\{x^2 + h(u_0)y^2 = -g(u_0),\ u = u_0\}$ for all $u_0 \in \mathbb{F}_q$. Those conics are nondegenerate whenever $g(u_0)h(u_0) \neq 0$, in which case they have $q + O(1)$ points. In remaining exceptional cases, they have at most $2q$ points. As a result, we get:

$$\#S(\mathbb{F}_q) = \big(q - O(1)\big) \cdot \big(q + O(1)\big) + O(1) \cdot O(q) = q^2 + O(q).$$

As for the number of preimages of F, we know by Theorem 1 that for each $\bar{x} \in X_{E,\mathbb{F}_q} \setminus X_{\text{bad}}$, where X_{bad} is a set of 39 points, there exists $c_{0,\bar{x}} \in [-22, 22]$ such that:

$$\#F^{-1}(\bar{x}) = 2q + c_{0,\bar{x}}\sqrt{q} + O(1) \qquad \forall \bar{x} \in X_{E,\mathbb{F}_q} \setminus X_{\text{bad}}$$

For $\bar{x} \in X_{\text{bad}}$, we can still obtain a less strict but simpler bound: note that for any fixed $u = u_0 \in \mathbb{F}_q$ the equations $\bar{x} = x_1(x, y, u_0)$, $\bar{x} = x_2(x, y, u_0)$ and $\bar{x} = x_3(x, y, u_0)$ have at most 2, 2, and 4 solutions in S, respectively (these solutions are given explicitly in Section 6). Hence, any point can have at most 8 preimages for any fixed u_0 and at most $8q$ preimages in all.

We can now bound Δ as follows:

$$\begin{aligned}
2\Delta &= \sum_{\bar{x} \in X_{E,\mathbb{F}_q} \setminus X_{\text{bad}}} \left| \frac{\#F^{-1}(\bar{x})}{\#S(\mathbb{F}_q)} - \frac{1}{\#X_{E,\mathbb{F}_q}} \right| + \sum_{\bar{x} \in X_{\text{bad}}} \left| \frac{\#F^{-1}(\bar{x})}{\#S(\mathbb{F}_q)} - \frac{1}{\#X_{E,\mathbb{F}_q}} \right| \\
&= \sum_{\bar{x} \in X_{E,\mathbb{F}_q} \setminus X_{\text{bad}}} \left| \frac{2q + c_{0,\bar{x}}\sqrt{q} + O(1)}{q^2 + O(q)} - \frac{1}{q/2 + c_E\sqrt{q} + O(1)} \right| + \\
&\qquad \sum_{\bar{x} \in X_{\text{bad}}} \left| \frac{c_{\text{bad},\bar{x}} q}{q^2 + O(q)} - \frac{1}{q/2 + c_E\sqrt{q} + O(1)} \right| \\
&= \sum_{\bar{x} \in X_{E,\mathbb{F}_q} \setminus X_{\text{bad}}} \frac{1}{q} \left| \big(2 + c_{0,\bar{x}} q^{-1/2} + O(q^{-1})\big) - \big(2 - c_E q^{-1/2} + O(q^{-1})\big) \right| + \\
&\qquad \sum_{\bar{x} \in X_{\text{bad}}} \frac{1}{q} \left| \big(c_{\text{bad},\bar{x}} + O(q^{-3})\big) - \big(2 - c_E q^{-1/2} + O(q^{-1})\big) \right| \\
&= \sum_{\bar{x} \in X_{E,\mathbb{F}_q} \setminus X_{\text{bad}}} \frac{1}{q} \left| (c_{0,\bar{x}} + c_E) q^{-1/2} + O(q^{-1}) \right| + \sum_{\bar{x} \in X_{\text{bad}}} \frac{1}{q} \left| c_{\text{bad},\bar{x}} - 2 + O(q^{-1/2}) \right|
\end{aligned}$$

where each of the constants $c_{0,\bar{x}}$ is in $[-22, 22]$ and each of the constants $c_{\text{bad},\bar{x}}$ is in $[0, 8]$. In particular, $|c_{0,\bar{x}} + c_E| \leq 23$ and $|c_{\text{bad},\bar{x}} - 2| \leq 6$ for all $\bar{x}$, and we

have:

$$\begin{aligned} 2\Delta &\leq \frac{\#\big(X_{E,\mathbb{F}_q} \setminus X_{\text{bad}}\big)}{q} \cdot \big(23q^{-1/2} + O(q^{-1})\big) + \frac{\#X_{\text{bad}}}{q} \cdot \big(6 + O(q^{-1/2})\big) \\ &= \frac{\frac{1}{2}q + O(\sqrt{q})}{q} \cdot \big(23 + o(1)\big)q^{-1/2} + \frac{39}{q} \cdot \big(6 + o(1)\big) \\ &= \Big(\frac{23}{2} + o(1)\Big)q^{-1/2} \leq 2 \cdot \big(6 + o(1)\big)q^{-1/2} \end{aligned}$$

as required.

As an easy consequence, we obtain the following theorem.

Theorem 2. *The map $F_0 \colon S(\mathbb{F}_q) \to X_{E,\mathbb{F}_q}$ of equation (7) is ε-admissible for $\varepsilon = \big(6 + o(1)\big)q^{-1/2}$. In particular, if $\mathfrak{h}$ is a random oracle with values in $S(\mathbb{F}_q)$, $F_0 \circ \mathfrak{h}$ is indifferentiable from an $X_{E,\mathbb{F}_q}$ random oracle.*

Moreover, the same results hold for $F_0^+ \colon S(\mathbb{F}_q) \times \{0,1\} \to E(\mathbb{F}_q)$.

Proof. By definition, we need to prove that F_0 is efficiently computatable, ε-regular and ε-samplable. Computability is obvious. Regularity is the result of Lemma 4. And 0-samplability is obtained using the preimage sampling algorithm discussed in Section 6 below. To fix ideas, we sketch its construction.

Fix $\bar{x} \in X_{E,\mathbb{F}_q}$. As previously mentioned, for any fixed $u_0 \in \mathbb{F}_q$, there are at most 8 preimages $(x, y, u) \in F^{-1}(\bar{x})$ such that $u = u_0$ (at most two coming from each of x_1 and x_2 and four coming from x_3). We can efficiently compute all those preimages and in particular count them. Therefore, the following simple rejection sampling algorithm has an output distribution uniform in $F^{-1}(\bar{x})$: pick u_0 uniformly at random, compute the list L_{u_0} of preimages with $u = u_0$, restart with probability $1 - \#L_{u_0}/8$ and otherwise return a random element of L_{u_0}.

Finally, the extension to F_0^+ is straightforward.

4 Parametrizing the SW Conic

4.1 Parametrizability conditions

In the previous section, we have seen how the Shallue–van de Woestijne construction could be leveraged to construct admissible encodings $F_0 \colon S(\mathbb{F}_q) \to X_{E,\mathbb{F}_q}$ and $F_0^+ \colon S(\mathbb{F}_q) \times \{0,1\} \to E(\mathbb{F}_q)$. However, we have also seen that mapping to $\mathbb{F}_q$-points on the surface S efficiently (without base field exponentiations) is a priori not straightforward, since the most naive approach involves finding points on new conics for all inputs.

Fortunately, the surface S has a fairly simple description: it can be seen as a *one-parameter family* of conics (the conics S_u; this is also called a relative conic over the u-line, or a fibration in conics, etc.). In any case, finding a global, two-parameter parametrization of S is thus a function field analogue of the classical problem, studied by Legendre, of finding rational points on conic over $\mathbb{Q}$.

In their paper [vHC06], van Hoeij and Cremona show that Legendre's original approach can be directly adapted to the function field case. They provide necessary and sufficient conditions for the existence of solutions, as well as an effective algorithm to compute the parametrization if it exists.

A special case of their main result in as follows.

Lemma 5 (van Hoeij–Cremona). *Let r, s be polynomials in $\mathbb{F}_q[u]$ that are coprime, squarefree, and such that at least one of them is of odd degree. Then, the following projective conic over $\mathbb{F}_q(t)$:*

$$X^2 + rY^2 + sZ^2 = 0$$

admits rational points over $\mathbb{F}_q(u)$ (i.e., a global rational parametrization) if and only if the following two conditions hold:

1. *$-r$ is a square in $\mathbb{F}_q[u]/(s)$*
2. *$-s$ is a square in $\mathbb{F}_q[u]/(r)$.*

Moreover, if this is the case, there is an efficient algorithm to compute those points.

Proof. This is a special case of [vHC06, Th. 1]. More precisely, the assumptions ensure that the conic is in reduced form and in "case 1", in the terminology of van Hoeij and Cremona, and the squareness conditions are equivalent to the existence of a "solubility certificate".

The proof presented by van Hoeij and Cremona is constructive in that it yields an explicit algorithm for finding the rational parametrization. Our case of interest, corresponding to the surface S, is $r = h(u) = 3u^2 + 4a$ and $s = g(u) = u^3 + au + b$ (except when $a = 0$, in which case a slight adjustment is necessary to meet the assumptions of the theorem). In that case, if a parametrization exists, it can be put in the form where $Z = 1$, and X, Y are polynomials of degree 2 and 1 in u respectively, as will be shown below. These polynomials depend only on the parameters a, b of the target elliptic curve, so the polynomial coefficients can be precomputed while their evaluation at a given u is done at runtime.

4.2 Curves with a parametrizable SW conic

Due to the conditions in Lemma 5, the SwiftEC encoding is not applicable to every ordinary elliptic curve. We present a different characterization of these conditions from the point of view of the target curve's geometric properties.

Theorem 3. *The surface S, as a one-parameter family of conics, admits a global two-parameter parametrization if and only if the following three conditions are satisfied.*

1. *The size of the field satisfies $q \equiv 1 \mod 3$ (i.e., -3 is a square in $\mathbb{F}_q$).*
2. *The discriminant $\Delta_E = -16(4a^3 + 27b^2)$ is a square in $\mathbb{F}_q$ (i.e. E has either zero or three points of order 2).*

3. At least one of the constants $\nu_\pm = \frac{1}{2}(-b \pm \sqrt{-3\Delta_E}/36)$ *is a square in* $\mathbb{F}_q$.

Proof. As a first observation, note that if we let $r = h(u)$ and $s = g(u)$, then r and s are indeed coprime (their resultant is $4a^3 + 27b^2 = -\Delta_E/16 \neq 0$) and s is of odd degree and squarefree. Moreover, r is squarefree if and only if $a \neq 0$. For now, we assume that $a \neq 0$, so that Lemma 5 applies directly. We will treat the special case of $a = 0$ at the end.

Let us first assume that $-h$ is a square in $\mathbb{F}_q[u]/(g)$ and $-g$ is a square in $\mathbb{F}_q[u]/(h)$. Note that h and g are coprime since their resultant is $4a^3 + 27b^2 = -\Delta_E/16 \neq 0$, so the law of quadratic reciprocity over function fields gives

$$\left(\frac{-h}{g}\right)_2 \left(\frac{g}{-h}\right)_2 = (-1)^{\frac{q-1}{2}\deg g \deg h} \chi_2(1)^{\deg h} \chi_2(-3)^{\deg g}$$
$$1 \cdot \left(\frac{g}{-h}\right)_2 = 1 \cdot 1 \cdot \chi_2(-3), \tag{9}$$

where $\left(\frac{\cdot}{f}\right)_2$ and $\chi_2(\cdot)$ denote quadratic residue symbols over $\mathbb{F}_q[u]/(f)$ and $\mathbb{F}_q$, respectively.

On the other hand, we have

$$1 = \left(\frac{-g}{h}\right)_2 = \left(\frac{-1}{h}\right)_2 \left(\frac{-g}{h}\right)_2 = \chi_2(-1)^2 \left(\frac{g}{-h}\right)_2 = \left(\frac{g}{-h}\right)_2,$$

so (9) reduces to $\chi_2(-3) = 1$, which shows the necessity of condition 1.

Next, since $-g$ is a square in $\mathbb{F}_q[u]/(h)$, there exists $\alpha, \beta \in \mathbb{F}_q$ such that:

$$-g \equiv (\alpha u + \beta)^2 \pmod{h}$$
$$-u^3 - au - b \equiv \alpha^2 u^2 + 2\alpha\beta u + \beta^2 \pmod{3u^2 + 4a}$$
$$\frac{4a}{3}u - au - b \equiv -\frac{4a}{3}\alpha^2 + 2\alpha\beta u + \beta^2 \pmod{3u^2 + 4a}$$
$$\frac{a}{3}u - b = 2\alpha\beta u + \big(-\frac{4a}{3}\alpha^2 + \beta^2\big).$$

It follows that the constants α, β satisfy

$$\frac{a}{3} = 2\alpha\beta \tag{10}$$
$$b = \frac{4a}{3}\alpha^2 - \beta^2. \tag{11}$$

Recalling that $a \neq 0$, it follows from (10) that $\alpha, \beta \neq 0$ and we can substitute $\beta = a/(6\alpha)$ into (11) to obtain

$$48a\alpha^4 - 36b\alpha^2 - a^2 = 0, \tag{12}$$

which is a quadratic equation on α^2 whose discriminant is $36^2b^2 + 192a^3 = -3\Delta_E$. Since -3 is a square, it follows that Δ_E must also be a square for α^2 to exist, showing the necessity of condition 2. The solution to (12) is then given by

$$\alpha^2 = \frac{36b \pm \sqrt{-3\Delta_E}}{96a} = \frac{-3}{4a}\nu_\pm. \tag{13}$$

If a is a square this means that at least one of $\nu_\pm$ must be a square for α to exist. On the other hand, if a is not a square then the same condition always holds since the product $\nu_+\nu_- = -a^3/27$ is a non-square.

The proof of the converse is similar: if conditions 2 and 3 are met then there exists $\alpha, \beta \in \mathbb{F}_q$ that are solutions to (10) and (11), which shows that $-g$ has a square root mod h, and then condition 1 together with (9) shows that $-h$ is a square mod g.

Finally, consider the special case $a = 0$. In that case, since $h(u) = 3u^2$, we can apply the change of variables $Y' = uY$ to reduce to the case of the conic:

$$X^2 + 3Y^2 + gZ^2 = 0,$$

i.e., $r = 3$ and $s = g$. It is then clear that r and s are coprime, squarefree, and one of them is of odd degree. Moreover, the condition that $-s$ is a square modulo r is vacuous, and the condition that $-r$ is a square modulo s simply says that -3 is a square in $\mathbb{F}_q[u]/(g)$; since that etale algebra admits either $\mathbb{F}_q$ or $\mathbb{F}_{q^3}$ as a factor, this is equivalent to -3 being a square in $\mathbb{F}_q$, namely $q \equiv 1 \pmod 3$ as required. This shows that in this case, condition 1 is necessary and sufficient. The result still holds, however, because conditions 2 and 3 become vacuous: the discriminant $\Delta_E = -16(27b^2) = -3 \cdot 12^2 b^2$ is always a square, and one of $\nu_\pm$ is always zero.

Out of the three conditions in Theorem 3, condition 1 is the most restrictive discarding half of the prime fields. Condition 3 only fails about 1/4 of the time, whereas condition 2 fails half of the time but can be circumvented half of those times as discussed in the next section. Notable curves that satisfy the conditions for SwiftEC include the NIST `P-256` curve, the curve `secp256k1` used in Bitcoin [SEC10] and the pairing-friendly curve `BLS12-381` [Bow17] as well as all BN curves [BN06] and BLS curves [BLS03] over any field with $q \equiv 1 \mod 3$. On the other hand, curves such as the Ed448-Goldilocks curve [Ham15] and the NIST `P-384` curve are incompatible due to the field size alone.

4.3 Reaching more curves with 2-isogenies

While Theorem 3 discards the possibility of applying SwiftEC directly to curves with a non-square discriminant, here we present a small modification that can work around this condition, at least some of the times. The condition that the discriminant be a square is invariant under isomorphisms, but not under isogenies. Hence, we may hope that there is an isogenous curve that satisfies the condition and compose the SwiftEC encoding to this curve with the isogeny to obtain a map to the original curve. Curves with a non-square discriminant always contain exactly one point of order 2, so one may be tempted to exploit the small 2-isogeny that is available. The following result shows that this intuition is correct, and indicates exactly when this is possible.

Theorem 4. *Let $E/\mathbb{F}_q$ be an elliptic curve with non-square discriminant. There exists a curve E' with square discriminant isogenous to E over $\mathbb{F}_q$ if and only if*

$E(\mathbb{F}_q)$ has a point of order 4. In this case, the isogeny can always be taken to be of degree 2.

Proof. First suppose we have a point $P_4 \in E(\mathbb{F}_q)$ of order 4, and let $P_2 = 2P_4$ be the unique point of order 2 in $E(\mathbb{F}_q)$. If $\phi : E \to E'$ is the isogeny with kernel $< P_2 >$, then $\phi(P_4)$ is a point of order 2 in E'. There must also exist a point $P_2' \in E'(\mathbb{F}_q)$ of order 2 generating the dual isogeny $\hat{\phi}$, and we cannot have $\phi(P_4) = P_2'$ because $\hat{\phi}(P_2') = 0$ but $\hat{\phi}(\phi(P_4)) = 2P_4 \neq 0$. This means we have two distinct points of order 2 in E', and their addition yields a third point of order 2, so E' must have a square discriminant as desired.

Conversely, if E has no point of order 4 then the group order is divisible by 2 exactly once, so any isogenous curve will also have exactly one point of order 2 and hence have a non-square discriminant.

Note that the application of the 2-isogeny is a 2-to-1 map that would make the distribution easily distinguishable from uniform. However, in essentially all cases of interest, one needs to sample points only in a specific subgroup orthogonal to the 2-torsion subgroup. For instance, consider `Curve25519` [Ber06] which is non-compatible with our construction because it does not have a square discriminant. The curve is given by

$$E_{25519} : y^2 = x^3 + 486662x^2 + x$$

over the prime field of size $p = 2^{255} - 19$. The group order for this curve is $\#E_{25519} = 8\ell$ where ℓ is a large prime, and points in the ℓ-torsion subgroup are used in the ECDH scheme. We can use SwiftEC to map onto the 2-isogenous curve

$$E' : y^2 = x^3 - 102314837774592x + 398341948567736549376$$

which does satisfy all conditions of Theorem 3. By composing with the 2-isogeny generated by $P_2' = (-11679888, 0)$ and the multiplication-by-4 map, we are able to hash into the $\ell-$torsion subgroup of Curve25519 at the cost of only an additional 20 field multiplications, 7 squarings and 11 additions. This is to our knowledge the only currently known way of hashing deterministically and indistinguishably into this subgroup using a single square root.

Unfortunately, some curves remain out of reach for SwiftEC due to condition 3 alone, even with this isogeny trick. One such example is NIST curve `P-521`.

5 The SwiftEC Encoding

5.1 Efficient computation

As a proof of principle, we have prepared a Sage implementation of SwiftEC that allows adding new compatible curves in a simple way. This implementation makes explicit the number of field operations needed and uses a constant number of them, but is non-constant time to the degree that the built-in field operations

are. Our implementation is freely available at https://github.com/Jchavezsaab/SwiftEC.

For curves with $a \neq 0$, the implementation makes use of the polynomials $X_0(u), Y_0(u)$ that evaluate a point in S_u as discussed in Section 4. Since these polynomials only depend on the curve coefficients a, b, they are precomputed and stored in the form of five field elements as detailed in Appendix A. On input u, t, the initial point $(X_0(u), Y_0(u)) \in S_u$ is evaluated and then a second point $(X, Y) \in S_u$ is obtained from the parametrization

$$\begin{aligned} X(u,t) &= \frac{g(u) + h(u)(Y_0(u) - tX_0(u))^2}{X_0(u)(1 + t^2 h(u))}, \\ Y(u,t) &= Y_0(u) + t(X - X_0(u)). \end{aligned} \tag{14}$$

In the case where $a = 0$, we have simply $g(u) = u^3 + b$ and $h(u) = 3u^2$. In this case the van Hoeij-Cremona algorithm described in Section 4 always yields the point at infinity $(X_0 : Y_0 : Z_0) = (\sqrt{-3} : 1 : 0)$, so the formulas for the parametrization have to be adjusted. We can skip the computation of $X_0(u), Y_0(u)$ altogether and apply the following formulas directly:

$$\begin{aligned} X(u,t) &= \frac{u^3 + b - t^2}{2t}, \\ Y(u,t) &= \frac{X(u,t) + t}{u\sqrt{-3}}. \end{aligned} \tag{15}$$

Finally, we apply the map ψ from (6) to get a point $(x_1, x_2, x_3, z) \in V(\mathbb{F}_q)$. It is not actually necessary to compute the z-coordinate of this point, and the x_i coordinates are computed projectively so that what we actually obtain is a projective triplet $(x_1 : x_2 : x_3 : \lambda)$. Note that this introduces a small bias towards the point at infinity: if any of the x_i are infinite then we have to set $\lambda = 0$ and all three points will be interpreted as being infinite. However, we neglect this since the bias is negligible and dealing with this case explicitly would produce a non-constant-time implementation.

We must then find which of the x_i is the x-coordinate of a point in $E(\mathbb{F}_q)$, choosing one arbitrarily but deterministically if all three are. This can be implemented in constant time as shown in Algorithm 1 which prioritizes x_3.

Finally, we use a single inverse to compute the affine x-coordinate and a square root computation (the only one throughout the whole program) to recover the y-coordinate. Note that there is a free choice for the sign of y in the end, which we integrate as an additional input bit.

5.2 XSwiftEC: x-only computation without exponentiation

Note that the only inverse and square root needed for SwiftEC are at the very end when the affine x, y coordinates are computed. However, there are many applications where obtaining an output in x-only projective coordinates is acceptable, and these operations can be omitted. The resulting XSwiftEC

Algorithm 1 x-picking algorithm.

Input: The projective x_i coordinates $(x_1 : x_2 : x_3 : \lambda)$ of a point in $V(\mathbb{F}_q)$
Output: One of the x_i which is the x-coordinate of a point in $E(\mathbb{F}_q)$.

1: $s_2 \leftarrow x_2^3\lambda + ax_2\lambda^3 + b\lambda^4$
2: $s_3 \leftarrow x_3^3\lambda + ax_3\lambda^3 + b\lambda^4$
3: $c_2 \leftarrow IsSquare(s_2)$
4: $c_3 \leftarrow IsSquare(s_3)$
5: $cswap(c_2, x_1, x_2)$
6: $cswap(c_3, x_1, x_3)$
7: **return** $(x_1 : \lambda)$

Table 1. Cost in operations of our implementations of SwiftEC for field additions, squarings, multiplications, Jacobi symbol computations, inversions, and square roots.

	Add	Sqr	Mul	Jac	Inv	Sqrt
SwiftEC	25	7	18	2	1	1
SwiftEC with isogeny	36	14	38	2	1	1
XSwiftEC	22	9	23	2	0	0
XSwiftEC with isogeny	33	14	35	2	0	0

algorithm requires no inversions, square roots or exponentiations of any kind, but only two Jacobi symbol computations that are considerably cheaper and other elementary field operations.

This is particularly useful for the cases when SwiftEC is composed with a 2-isogeny as described in Section 4.3: even if an affine x, y output is desired, we are better off using XSwiftEC and recovering the affine coordinates until after applying the projective x-only 2-isogeny formulas.

Although the output $(x : \lambda)$ that is obtained is indistinguishable from uniform as a projective pair, the individual values of x and λ are not and may leak information about the input. This can be easily circumvented by multiplying both coordinates by a random field element, or it may be ignored to avoid relying on randomness in applications where this leakage is not a concern.

5.3 Implementation results

We summarize in Table 1 the cost in operations for each version of SwiftEC. The most noteworthy feature is the requirement of only one square root computation (and none when the y coordinate is not required), which is an improvement on previous admissible encodings to ordinary elliptic curves.

The results shown are for the $a \neq 0$ implementation. The implementations for $a = 0$ always save exactly 7 additions and 6 multiplications due to the simpler formulas in (15).

6 SwiftEC For Point Representation: ElligatorSwift

In this section we describe an algorithm to efficiently compute a uniformly random preimage of any point under SwiftEC. The existence of this algorithm is required for the encoding to be *admissible*, which is crucial for using SwiftEC as part of a cryptographically secure hash function as described in Section 2. Moreover, it is important in practice because it allows us to encode points in an elliptic curve as uniform bitstrings, as is done in Elligator [BHKL13] and Elligator Squared [Tib14a].

Compared to Elligator Squared, our ElligatorSwift construction has the advantage that it does not need to compute any encodings in the forward direction. Indeed, all we need is to sample a random $u \in \mathbb{F}_q$ and then find an inverse $F_{0,u}^{-1}(P)$ of the SW encoding.

We first focus on inverting the map Ψ and note that under a change of variables $v = x/2y - u/2$ and $w = 2y$, the image in (6) becomes

$$\begin{aligned} x_1 &= v \\ x_2 &= -u - v \\ x_3 &= u + w^2, \end{aligned}$$

while the equation for the conic becomes

$$w^2(u^2 + uv + v^2 + a) = -(u^3 + au + b). \tag{16}$$

This yields up to four possible preimages for a given point $(x, y) \in E(F)$, namely:

1. $v = x$ and w^2 derived from (16), if x was drawn from x_1
2. $v = -u - x$ and w^2 derived from (16), if x was drawn from x_2

3,4. $w^2 = x - u$ and v derived from (16), if x was drawn from x_3,

where the last case actually contains two preimages since (16) is a quadratic equation for v with solutions

$$v = \frac{-u}{2} \pm \frac{\sqrt{-w^2(4u^3 + 4au + 4b + 3w^2u^2 + 4aw^2)}}{2w^2}.$$

Moreover, all cases have a duplicity from choosing the sign of $w = \sqrt{w^2}$, so there are up to 8 preimages in total. Of course, some of the square roots needed may not exist and so different values of u will yield a different number of preimages of a given point (including possibly none). On top of this, if the preimage comes from cases 1 or 2 but results in values where all three x_i yield points in $E(\mathbb{F}_q)$, then the preimage will be invalid even if the square root is well-defined since Algorithm 1 in the forward encoding would have prioritized x_3 over the intended one. Care must therefore be taken to check for the existence of the various square roots and restart the procedure when appropriate, as shown in Algorithm 2. This makes the algorithm run in non-constant time but ensures that the preimage is uniformly sampled.

What remains is just to switch back to x, y coordinates and invert the parametrization (14) to recover the parameter t.

Remark: For implementations with $a = 0$ we must take into account the different parametrization formulas in (15). In this case, lines 29 and 30 of Algorithm 2 can be replaced by simply $t \leftarrow Yu\sqrt{-3} - X$, where the constant $\sqrt{-3}$ is part of the precomputed parameters.

We assume that the square root function makes a random choice of sign each time it is called, and that it returns $Null$ for non-squares. It is easy to see that the output of Algorithm 2 is uniformly distributed since each u is attempted with a random choice of one of the 4 cases, so the probability of each u being successful is proportional to how many preimages exist under it.

The main cost of Algorithm 2 is an average of 1.5 square root computations per iteration. Since most points have roughly $2q$ preimages as per Theorem 1, we can expect each choice of u to contain on average 2 valid preimages out of the 8 possible ones, and so the expected number of iterations is 4. Notice however that a failed iteration can be aborted before computing any square roots by first computing the corresponding Jacobi symbols, which can be done much more efficiently with constant-time efficient implementations such as [Por20,Ham21,AG21]. The cost of ElligatorSwift is therefore always exactly 1 or 2 square root computations, and 6 Jacobi symbol computations on average. This is a considerable improvement over Elligator Squared, where each failed iteration would have contributed an additional square root from computing the forward map and the average total cost is 6.5 square roots.

Note that in the case of x-only arithmetic there are no savings on ElligatorSwift since Algorithm 2 is already agnostic to y but still requires several square roots. As for curves where we need to compose SwiftEC with a 2-isogeny, we can obtain a corresponding variant of ElligatorSwift by composing with the dual isogeny, but this has the side effect of introducing a multiplication-by-2 in the round trip. This can be circumvented by starting with a point halving before applying ElligatorSwift, which is important for demonstrating that the encoding with the isogeny trick is still admissible. However, the resulting ElligatorSwift construction is unappealing in terms of efficiency.

7 Conclusion

The SwiftEC construction introduced in this paper is the first admissible and constant-time encoding using a single square root that is applicable to a large class of ordinary elliptic curves. While some curves are still incompatible, the construction applies to roughly 9/16 of all curves over fields with $q \equiv 1 \mod 3$. The inverse encoding also results in an Elligator-like encoding that is significantly more efficient than previous constructions, using more than 4 times less square roots on average than Elligator Squared, while retaining the same data transmission size of two field elements.

It is still an open problem to determine if there are any workarounds that could extend this encoding to more of the non-compatible curves, or even to find

Algorithm 2 ElligatorSwift.

Input: $(x, y) \in E(\mathbb{F}_q)$

Output: $u, t, b \xleftarrow{\$} \text{SwiftEC}^{-1}(x, y)$

$u \xleftarrow{\$} \mathbb{F}_q$
$case \xleftarrow{\$} \{1, 2, 3, 4\}$
if $case == 1$ **then**
    $v \leftarrow x$
    if IsSquare$((-v-u)^3 + a(-v-u) + b)$ **then**
        go to 1
    end if
    $w^2 \leftarrow -(u^3 + au + b)/(u^2 + uv + v^2 + a)$
else if $case == 2$ **then**
    $v \leftarrow -x - u$
    if IsSquare$(v^3 + av + b)$ **then**
        go to 1
    end if
    $w^2 \leftarrow -(u^3 + au + b)/(u^2 + uv + v^2 + a)$
else
    $w^2 \leftarrow x - u$
    $r \leftarrow \sqrt{-w^2(4u^3 + 4au + 4b + 3w^2u^2 + 4aw^2)}$
    if $r == Null$ **then**
        go to 1
    end if
    $v \leftarrow -u/2 + r/2w^2$
end if
$w \leftarrow \sqrt{w^2}$
if $w == Null$ **then**
    go to 1
end if
$Y \leftarrow 2w/2$
$X \leftarrow 2Y(v + u/2)$
Evaluate $X_0(u)$ and $Y_0(u)$ from precomputed polynomials
$t \leftarrow (Y - Y_0)/(X - X_0)$
$b \leftarrow sign(y)$
return u, t, b

a single-square root admissible encoding that could be applied to all ordinary elliptic curves.

References

AG21. Diego F. Aranha and Conrado P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic/blob/symbol-asm/src/fp/relic_fp_smb.c, 2021.

BCI+10. Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 237–254. Springer, Heidelberg, August 2010.

Ber06. Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, Heidelberg, April 2006.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BHKL13. Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 967–980. ACM Press, November 2013.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

BLS03. Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 257–267. Springer, Heidelberg, September 2003.

BN06. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, Heidelberg, August 2006.

Bow17. Sean Bowe. BLS12-381: New zk-SNARK elliptic curve construction. https://electriccoin.co/blog/new-snark-curve/, 2017.

BY19. Daniel J. Bernstein and Bo-Yin Yang. Fast constant-time gcd computation and modular inversion. *IACR TCHES*, 2019(3):340–398, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8298.

CM00. Francis N. Castro and Carlos J. Moreno. Mixed exponential sums over finite fields. *Proc. Amer. Math. Soc.*, 128(9):2529–2537, 2000.

Far11. Reza Rezaeian Farashahi. Hashing into Hessian curves. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 278–289. Springer, Heidelberg, July 2011.

FFS+13. Reza Rezaeian Farashahi, Pierre-Alain Fouque, Igor E. Shparlinski, Mehdi Tibouchi, and José Felipe Voloch. Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. *Math. Comput.*, 82(281):491–512, 2013.

FHSS+22. Armando Faz-Hernandez, Sam Scott, Nick Sullivan, Riad S. Wahby, and Christopher A. Wood. Hashing to elliptic curves. https://tools.ietf.org/id/draft-irtf-cfrg-hash-to-curve-14.html, February 2022.

FJT13. Pierre-Alain Fouque, Antoine Joux, and Mehdi Tibouchi. Injective encodings to elliptic curves. In Colin Boyd and Leonie Simpson, editors, *ACISP 13*, volume 7959 of *LNCS*, pages 203–218. Springer, Heidelberg, July 2013.

FT10. Pierre-Alain Fouque and Mehdi Tibouchi. Deterministic encoding and hashing to odd hyperelliptic curves. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *PAIRING 2010*, volume 6487 of *LNCS*, pages 265–277. Springer, Heidelberg, December 2010.

FT12. Pierre-Alain Fouque and Mehdi Tibouchi. Indifferentiable hashing to Barreto-Naehrig curves. In Alejandro Hevia and Gregory Neven, editors, *LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 1–17. Springer, Heidelberg, October 2012.

Ham15. Mike Hamburg. Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625, 2015. https://eprint.iacr.org/2015/625.

Ham21. Mike Hamburg. Computing the Jacobi symbol using Bernstein–Yang. Cryptology ePrint Archive, Paper 2021/1271, 2021. https://eprint.iacr.org/2021/1271.

Ica09. Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316. Springer, Heidelberg, August 2009.

KLR10. Jean-Gabriel Kammerer, Reynald Lercier, and Guénaël Renault. Encoding points on hyperelliptic curves over finite fields in deterministic polynomial time. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *PAIRING 2010*, volume 6487 of *LNCS*, pages 278–297. Springer, Heidelberg, December 2010.

Kos22. Dmitrii Koshelev. Indifferentiable hashing to ordinary elliptic $\mathbb{F}_q$-curves of $j = 0$ with the cost of one exponentiation in $\mathbb{F}_q$. *Des. Codes Cryptogr.*, 90(3):801–812, 2022.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

Per91. Marc Perret. Multiplicative character sums and kummer coverings. *Acta Arith.*, 59:279–290, 1991.

Por20. Thomas Pornin. Faster modular inversion and Legendre symbol, and an X25519 speed record. https://research.nccgroup.com/2020/09/28/faster-modular-inversion-and-legendre-symbol-and-an-x25519-speed-record/, September 2020.

Ros02. Michael Rosen. *Number Theory in Function Fields*. Springer New York, NY, 2002.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

SEC10. Certicom research, standards for efficient cryptography 2: Recommended elliptic curve domain parameters, January 2010.

SvdW06. Andrew Shallue and Christiaan E. van de Woestijne. Construction of rational points on elliptic curves over finite fields. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *Algorithmic Number Theory, 7th International Symposium, ANTS-VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 510–524. Springer, 2006.

Tib14a. Mehdi Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 139–156. Springer, Heidelberg, March 2014.

Tib14b. Mehdi Tibouchi. Impossibility of surjective Icart-like encodings. In Sherman S. M. Chow, Joseph K. Liu, Lucas C. K. Hui, and Siu-Ming Yiu, editors, *ProvSec 2014*, volume 8782 of *LNCS*, pages 29–39. Springer, Heidelberg, October 2014.

TK17. Mehdi Tibouchi and Taechan Kim. Improved elliptic curve hashing and point representation. *Des. Codes Cryptogr.*, 82(1-2):161–177, 2017.

Ula07. Maciej Ulas. Rational points on certain hyperelliptic curves over finite fields. *Bull. Pol. Acad. Sci. Math.*, 55(2):97–104, 2007.

vHC06. Mark van Hoeij and John Cremona. Solving conics over function fields. *Journal de Théorie des Nombres de Bordeaux*, 18(3):595–606, 2006.

WB19. Riad S. Wahby and Dan Boneh. Fast and simple constant-time hashing to the BLS12-381 elliptic curve. *IACR TCHES*, 2019(4):154–179, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8348.

A Explicit formulas for finding a point in the parametrized conic

Here we detail the formulas used for the precomputation of the polynomials $X_0(u), Y_0(u)$ that are used for finding a fixed point $(X_0, Y_0) \in S_u$. The polynomials can be written as

$$X_0(u) = (Au^2 + Bu + C)/Z$$

$$Y_0(u) = (Du + E)/Z,$$

where the coefficients A, B, C, D, E, Z are derived from van Hoeij and Cremona's algorithm FindPoint [vHC06] given the elliptic curve coefficients a, b. There are four distinct cases, depending on whether the polynomails h, g from (5) are reducible or not. Note that h is reducible if and only if a is a square, and g is reducible if and only if $E : y^2 = g(x)$ has points of order 2.

A.1 Case 1: h, g are both irreducible

The coefficients are as follows:

$$\begin{aligned}
A &= -9\beta\sqrt{\Delta_E} \\
B &= -6a\alpha\sqrt{\Delta_E} \\
C &= -6a\beta\sqrt{\Delta_E} \\
D &= 48a^2\alpha + 108b\beta \\
E &= 72ab\alpha - 24a^2\beta \\
Z &= 2a(4a\alpha^2 + 3\beta^2)\sqrt{\Delta_E},
\end{aligned}$$

where

$$\alpha = -3\sqrt{\frac{-36b - \sqrt{-3\Delta_E}}{2a\Delta_E}}, \quad \beta = \alpha(36b - \sqrt{-3\Delta_E})/(12a).$$

Note that there is exactly one choice of sign for $\sqrt{-3\Delta_E}$ that makes the radicand in α a square, and the same sign must be used throughout.

A.2 Case 2: h reducible, g irreducible

Let $s = 2\sqrt{a/-3}$ be a root of h, and define $\xi_\pm = 1/\sqrt{g(\pm s)}$ for arbitrarily chosen square-root signs. The coefficients are then as follows:

$$\begin{aligned}
A &= -3s(\xi_+ + \xi_-)\sqrt{\Delta_E} \\
B &= -2a(\xi_+ - \xi_-)\sqrt{\Delta_E} \\
C &= -2as(\xi_+ + \xi_-)\sqrt{\Delta_E} \\
D &= 16a^2(\xi_+ - \xi_-) + 36bs(\xi_+ + \xi_-) \\
E &= 24ab(\xi_+ - \xi_-) - 8a^2s(\xi_+ + \xi_-) \\
Z &= 4as\xi_+\xi_-\sqrt{\Delta_E},
\end{aligned}$$

A.3 Case 3: h irreducible, g reducible

Let r_1, r_2, r_3 be the roots of g, i.e. the x-coordinate of the points of order 2 in E (note that g must split completely since Δ_E is a square). Then, the coefficients are as follows:

$$\begin{aligned}
A &= 3\sqrt{\Delta_E}\Big((r_1 + r_2 + r_3)\alpha - 3\beta\Big) \\
B &= -3\sqrt{\Delta_E}\alpha(4a + r_1^2 + r_2^2 + r_3^2) \\
C &= \sqrt{\Delta_E}\Big(4a\alpha(r_1 + r_2 + r_3) + 3\beta(r_1^2 + r_2^2 + r_3^2)\Big) \\
D &= -32ar_1^2\alpha + 32ar_1r_2\alpha + 12r_1^3r_2\alpha - 32ar_2^2\alpha - 24r_1^2r_2^2\alpha + 12r_1r_2^3\alpha \\
&\quad + 32ar_1r_3\alpha + 12r_1^3r_3\alpha + 32ar_2r_3\alpha + 12r_2^3r_3\alpha - 32ar_3^2\alpha - 24r_1^2r_3^2\alpha \\
&\quad - 24r_2^2r_3^2\alpha + 12r_1r_3^3\alpha + 12r_2r_3^3\alpha - 24r_1^3\beta + 12r_1^2r_2\beta + 12r_1r_2^2\beta \\
&\quad - 24r_2^3\beta + 12r_1^2r_3\beta + 12r_2^2r_3\beta + 12r_1r_3^2\beta + 12r_2r_3^2\beta - 24r_3^3\beta \\
E &= 16ar_1^2r_2\alpha + 16ar_1r_2^2\alpha + 16ar_1^2r_3\alpha - 96ar_1r_2r_3\alpha \\
&\quad - 24r_1^3r_2r_3\alpha + 16ar_2^2r_3\alpha + 24r_1^2r_2^2r_3\alpha - 24r_1r_2^3r_3\alpha \\
&\quad + 16ar_1r_3^2\alpha + 16ar_2r_3^2\alpha + 24r_1^2r_2r_3^2\alpha + 24r_1r_2^2r_3^2\alpha \\
&\quad - 24r_1r_2r_3^3\alpha + 12r_1^3r_2\beta + 12r_1r_2^3\beta + 12r_1^3r_3\beta - 24r_1^2r_2r_3\beta - 24r_1r_2^2r_3\beta \\
&\quad + 12r_2^3r_3\beta - 24r_1r_2r_3^2\beta + 12r_1r_3^3\beta + 12r_2r_3^3\beta \\
Z &= \sqrt{\Delta_E}(4a + r_1^2 + r_2^2 + r_3^2)(4a\alpha^2 + 3\beta^2)
\end{aligned}$$

A.4 Case 4: h, g are both irreducible

With all constants as before, the coefficients are as follows.

$$
\begin{aligned}
A &= 3\sqrt{\Delta_E}\Big((r_1+r_2+r_3)(\xi_+-\xi_-) - 3s(\xi_++\xi_-)\Big)\\
B &= -3\sqrt{\Delta_E}(r_1^2+r_2^2+r_3^2+4a)(\xi_+-\xi_-)\\
C &= 3\sqrt{\Delta_E}\Big(s(r_1^2+r_2^2+r_3^2)(\xi_++\xi_-) + 4a(r_1+r_2+r_3)(\xi_+-\xi_-)\Big)\\
D &= 12s\Big(\xi_++\xi_-\Big)\Big(-2r_1^3+r_1^2r_2+r_1r_2^2-2r_2^3+r_1^2r_3+r_2^2r_3+r_1r_3^2+r_2r_3^2-2r_3^3\Big)\\
&\quad + 4\Big(\xi_+-\xi_-\Big)\Big(-8ar_1^2+8ar_1r_2+3r_1^3r_2-8ar_2^2-6r_1^2r_2^2+3r_1r_2^3+8ar_1r_3+3r_1^3r_3\\
&\qquad\qquad + 8ar_2r_3+3r_2^3r_3-8ar_3^2-6r_1^2r_3^2-6r_2^2r_3^2+3r_1r_3^3+3r_2r_3^3\Big)\\
E &= 12s\Big(\xi_++\xi_-\Big)\Big(r_1^3r_2+r_1r_2^3+r_1^3r_3-2r_1^2r_2r_3-2r_1r_2^2r_3+r_2^3r_3-2r_1r_2r_3^2+r_1r_3^3+r_2r_3^3\Big)\\
&\quad + 8\Big(\xi_+-\xi_-\Big)\Big(2ar_1^2r_2+2ar_1r_2^2+2ar_1^2r_3-12ar_1r_2r_3-3r_1^3r_2r_3+2ar_2^2r_3+3r_1^2r_2^2r_3\\
&\qquad\qquad - 3r_1r_2^3r_3+2ar_1r_3^2+2ar_2r_3^2+3r_1^2r_2r_3^2+3r_1r_2^2r_3^2-3r_1r_2r_3^3\Big)\\
Z &= 6\sqrt{\Delta_E}(r_1^2+r_2^2+r_3^2+4a)s\xi_+\xi_-
\end{aligned}
$$