LMS J. Comput. Math. 17 (Special issue A) (2014) 181–202
doi:10.1112/S1461157014000394

Hyper-and-elliptic-curve cryptography

Daniel J. Bernstein and Tanja Lange

Abstract

This paper introduces 'hyper-and-elliptic-curve cryptography', in which a single high-security group supports fast genus-2-hyperelliptic-curve formulas for variable-base-point single-scalar multiplication (for example, Diffie–Hellman shared-secret computation) and at the same time supports fast elliptic-curve formulas for fixed-base-point scalar multiplication (for example, key generation) and multi-scalar multiplication (for example, signature verification).

1. *Introduction*

> *We would very much like to see forward secrecy become the norm and hope that our deployment serves as a demonstration of the practicality of that vision.*
> — 'Protecting data for the long term with forward secrecy', 22 November 2011 [**38**]

> *Forward secrecy is just the latest way in which Twitter is trying to defend and protect the user's voice.* — 'Forward secrecy at Twitter', 22 November 2013 [**31**]

The classic Diffie–Hellman (DH) protocol [**17**, § 3] sets up secure communication channels between any number of users as follows. Alice has a long-term secret key a and a long-term public key g^a, where g is a standard element of the multiplicative group of a finite field. Similarly, Bob has a long-term secret key b and a long-term public key g^b; Charlie has a long-term secret key c and a long-term public key g^c; etc. Alice and Bob each compute g^{ab}, which they use as a long-term key for secret-key cryptography to efficiently encrypt and authenticate messages. Alice and Charlie encrypt using g^{ac}; Bob and Charlie encrypt using g^{bc}; etc.

This protocol never erases keys, so it does not provide *forward secrecy*. An attacker who steals Bob's computer, after recording all network communication, sees g^{ab} and g^{bc} and decrypts Bob's past messages, even if Bob has erased all of the messages. Bob cannot stop this attack by erasing g^{ab} and g^{bc}: the attacker simply recomputes g^{ab} and g^{bc} from b. Bob cannot erase b: Bob needs b to compute shared secrets with new users.

The obvious way to provide forward secrecy is to further encrypt messages using an *ephemeral* variant of the DH protocol. Alice and Bob start by setting up a secure channel as above, using Alice's long-term public key g^a and Bob's long-term public key g^b. Then, for each message from Alice to Bob, Alice generates a one-time secret key r and sends g^r to Bob through the secure channel; Bob also generates a one-time secret key s and sends g^s to Alice through the secure channel. Alice encrypts the message using g^{rs}, sends the ciphertext C through the secure channel, and throws away r and g^{rs}. Bob decrypts the message using g^{rs} and throws away s and g^{rs}. An attacker who steals Bob's computer still has the power to decrypt the original channel, obtaining g^r, g^s, and C, but there is no obvious way to recover

Received 27 February 2014; revised 23 May 2014.

2010 Mathematics Subject Classification 11G20 (primary), 14G50, 94A60 (secondary).

Contributed to the Algorithmic Number Theory Symposium XI, GyeongJu, Korea, 6–11 August 2014.

This work was supported by the National Science Foundation under grant 1018836 and by the Netherlands Organisation for Scientific Research (NWO) under grant 639.073.005. Permanent ID of this document: a1ca779b2e78748e02d4dc09618b9588. Date: 2014.05.25.

the original message. Of course, the attack compromises the confidentiality and integrity of *future* messages, but *past* messages are still protected.

Notice that the ephemeral DH protocol has different performance characteristics from the original protocol. In the original protocol, the dominant computation is a variable-base exponentiation $a, g^b \mapsto g^{ab}$: for U users Alice does U variable-base exponentiations g^{ab}, g^{ac}, etc. and only one fixed-base exponentiation $a \mapsto g^a$. In M runs of the ephemeral protocol, Alice performs M fixed-base exponentiations $r \mapsto g^r$ and M variable-base exponentiations $r, g^s \mapsto g^{rs}$, so Alice benefits significantly from speedups in either type of exponentiation. One can consider intermediate possibilities, such as reusing an ephemeral key for several messages, but forward secrecy is strongest when a key is discarded immediately after its first use.

1.1. *Elliptic curves and hyperelliptic curves*

Modern cryptography replaces the multiplicative groups in DH with elliptic-curve groups, as proposed by Miller [40] and independently by Koblitz [36]. This loses an important constant factor in the number of field operations required for a group operation, but it gains much more from avoiding index-calculus attacks. Specifically, to achieve a security level around 2^{128}, elliptic-curve groups use base fields of size around 2^{256}, while multiplicative groups need base fields of size around 2^{3000}. See, for example, [28].

The recent paper [6] by Bos, Costello, Hisil, and Lauter shows that for high-security DH one obtains even better performance from a different option: Jacobian groups of hyperelliptic curves of genus 2. The main advantage of genus 2 over genus 1 is that a much smaller base field, specifically a field of size around 2^{128}, produces a group of size around 2^{256} and a security level around 2^{128}. Reducing the number of bits in the field by a factor of 2 typically produces a speedup factor around 3, depending on various details of field arithmetic. The disadvantage of genus 2 is that each group operation requires many more field operations; but for Gaudry's [24] Kummer-surface formulas this loss factor is only slightly above 2. Even better, 24% of Gaudry's field multiplications are multiplications by curve parameters that can be chosen to be small; a secure small-parameter genus-2 curve was announced by Gaudry and Schost [27] after a massive point-counting computation. A further advantage of genus 2, exploited in a very recent paper [4] by Bernstein, Chuengsatiansup, Lange, and Schwabe, is a synergy between the structure of Gaudry's formulas and the availability of vector operations in modern CPUs.

One can speed up genus 1 using 'non-constant-time' addition chains. However, non-constant-time computations are a security problem; see, for example, the attacks cited in [4, § 1.2].

One can also speed up genus 1 by applying endomorphisms on suitably chosen curves: for example, rewriting aP as $a_0P + a_1\phi(P)$ where a_0 and a_1 have half as many bits as a. See, for example, [19, 44]. Analogous ideas in genus 2 seem less effective; see [7]. However, endomorphisms in this context are patented, and are thus not helpful for users concerned with the real-world cost of cryptography. Furthermore, even with this speedup, genus 1 is not as fast as genus 2; see [4].

1.2. *Hyperelliptic curves and forward secrecy*

The comparison between genus 1 and genus 2 changes when one switches from classic DH to ephemeral DH. Genus 2 is the speed leader for variable-base scalar multiplication $r, sG \mapsto rsG$, but genus 1 is the speed leader for fixed-base scalar multiplication $r \mapsto rG$, and for forward secrecy both operations are important. There is some speedup from variable base to fixed base in genus 2 (see [6] for a detailed analysis), but there is a much larger speedup in genus 1. We summarize the relative time required for each operation as follows:

fixed-base genus 1 < fixed-base genus 2 < variable-base genus 2 < variable-base genus 1.

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

Given this picture, it might seem obvious that one cannot simultaneously take advantage of the fastest fixed-base operations and the fastest variable-base operations. Choosing genus 2 means slowing down key generation. Choosing genus 1 means slowing down the computation of a shared secret.

To resolve this problem we propose using public keys in a group that can be viewed simultaneously as a genus-1 group and a genus-2 group. More precisely, assume that we have:

- an elliptic curve E over $\mathbf{F}_{p^2}$, specifically (for speed and simplicity) an $\mathbf{F}_{p^2}$-complete Edwards curve;
- the Jacobian J of a genus-2 hyperelliptic curve over $\mathbf{F}_p$, specifically one supporting a fast (that is, small-parameter) Kummer surface; and
- an efficient $\mathbf{F}_p$-isogeny from W to J, where W is the Weil restriction of E from $\mathbf{F}_{p^2}$ to $\mathbf{F}_p$.

Setting up this situation is the main work in this paper. Alice then uses fast elliptic-curve formulas for fixed-base scalar multiplication to generate a public key rG in $E(\mathbf{F}_{p^2}) = W(\mathbf{F}_p)$. Either Alice or Bob applies the isogeny to rG, obtaining an equivalent public key in $J(\mathbf{F}_p)$. Bob then uses fast Kummer-surface formulas to compute a shared secret in $J(\mathbf{F}_p)$. We suggest having Bob apply the isogeny, since uncompressing a compressed elliptic-curve point is simpler than uncompressing a compressed Kummer-surface point.

Of course, one can also use the dual isogeny to map from $J(\mathbf{F}_p)$ back to $E(\mathbf{F}_{p^2})$. However, it seems natural to start with $E(\mathbf{F}_{p^2})$, since elliptic-curve formulas are very fast for key generation. The obvious general strategy is to use Edwards coordinates on $E(\mathbf{F}_{p^2})$ for computations where those formulas are fastest, and to use Kummer coordinates on $J(\mathbf{F}_p)$ for computations where those formulas are fastest, using the isogenies to convert whenever necessary.

Any further evolution of coordinate systems and formulas can of course be integrated into the same picture. It seems reasonable to speculate that neither genus 1 nor genus 2 will end up as a clear winner, so the ability to mix genus 1 and genus 2 will remain useful.

1.3. *Further applications*

Our approach is applicable to many contexts in which different types of scalar multiplication are mixed; forward secrecy is obviously an important application but there are other applications. For example, there are safe methods to use a single element of a group of order approximately 2^{256} as a long-term public key for both DH and signatures; see, for example, [**13**, **29**, **45**]. Using two separate keys, one for DH and one for signatures, means transmitting both of those keys, and in some settings also transmitting a signature of the DH key under the signing key; a single key is clearly much more satisfactory. With our techniques, this single key allows fast genus-1 formulas for key generation, signing, and signature verification while simultaneously allowing fast genus-2 formulas for DH shared-secret computation. If signing and signature verification are much more frequent than encryption then the genus-1 operations will be dominant, but in general one should expect many different levels of balance between the genus-1 operations and the genus-2 operations.

A further advantage of genus 1 for signature verification is that there are no exceptional cases in the standard addition law for $E(\mathbf{F}_{p^2})$ when E is an $\mathbf{F}_{p^2}$-complete Edwards curve. For comparison, all fast genus-2 addition laws in the literature have exceptional cases. Using genus-1 addition by default, and moving to a genus-2 ladder for shared-secret computation, means that we avoid all of these exceptional cases.

1.4. *Notes regarding terminology*

We use the geometric definition of an isogeny as a rational map such that 0 maps to 0, all geometric fibers (preimage sets of points over the algebraic closure) are finite, and all geometric

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

fibers are nonempty. (In particular, a constant map from W to J is not an isogeny.) If ι is an isogeny then ι is defined everywhere and is a group homomorphism. This does not mean that any particular rational *functions* defining ι are defined everywhere; recall that a rational map is, by definition, an equivalence class of almost-everywhere-defined rational functions.

It is slightly sloppy to refer to 'the' Weil restriction. There are actually many different choices of Weil restrictions, corresponding to different choices of a basis (b_0, b_1) for $\mathbf{F}_{p^2}$ over $\mathbf{F}_p$: specifically, the affine part of W is the set of (x_0, x_1, y_0, y_1) such that $(x_0b_0 + x_1b_1, y_0b_0 + y_1b_1)$ is a point on E. Modifying (b_0, b_1) produces a linearly isomorphic but not identical variety. If we were defining and evaluating the efficiency of $\mathbf{F}_p$-algebraic algorithms for computing the rational maps that appear in this paper then we would need this extra level of mathematical precision; fortunately, all of the maps that we present are clearly much faster than scalar multiplication, so a detailed cost evaluation is unnecessary. Related choices do become important in § 6, where we choose Δ and lift the whole picture to $\mathbf{Q}(\sqrt{\Delta})$.

2. *Weierstrass to genus-2 Jacobian: efficient isogenies for Scholten curves*

We do not claim credit for the fact that one can construct elliptic curves over $\mathbf{F}_{p^2}$ isogenous (after restriction of scalars) to genus-2 Jacobians over $\mathbf{F}_p$: we reuse a construction published by Scholten ten years ago in [48]. Scholten credits to Diem the case where E has full 2-torsion defined over $\mathbf{F}_{p^2}$, but Scholten's construction is simpler than Diem's construction.

Scholten's goal was to write down hyperelliptic curves that allowed fast point-counting. By constructing curves so that the Weil restriction W of the elliptic curve from $\mathbf{F}_{p^2}$ to $\mathbf{F}_p$ is isogenous to the Jacobian J of a genus-2 hyperelliptic curve over $\mathbf{F}_p$, Scholten guaranteed that $\#J(\mathbf{F}_p) = \#W(\mathbf{F}_p) = \#E(\mathbf{F}_{p^2})$. See [48, Lemma 2.1]. Counting points on elliptic curves is reasonably fast, producing $\#E(\mathbf{F}_{p^2})$ and thus the desired $\#J(\mathbf{F}_p)$.

The idea of fast point-counting on genus-2 curves by constructive Weil restriction was introduced by Gaudry, Hess, and Smart in [26], but the constructions in [26] were limited to characteristic 2; odd characteristic was called 'hard' in [26, § 7.2] and 'rather difficult' in [23, § 7]. Various odd-characteristic constructions appeared in [14–16, 48, 51]. Special cases with extra small-norm endomorphisms were used in [21, 47]. Many of these papers feature 'Weil-descent attacks' and 'cover attacks' as another application of Weil restriction, as suggested by Frey in [22]; Weil-descent attacks using Scholten curves appeared in [2, 33, 41].

We do claim credit for the idea of using an isogeny to convert keys between E and J, making cryptography faster. At this point one can and should object that [48, Lemma 2.1] merely guarantees the existence of an isogeny from W to J; it does not guarantee the existence of an *efficient* isogeny from W to J. For most pairs of isogenous Abelian varieties, the fastest isogenies known are much slower than scalar multiplication. (This was not an issue for Scholten: any isogeny, no matter how slow, is adequate to show that $\#J(\mathbf{F}_p) = \#W(\mathbf{F}_p)$. It was also not a serious issue for attack papers such as [2]: the use of J in [2] was for carrying out a Weil-descent attack against E, and other steps of this attack were much more expensive.) This could be fatal for our idea of applying an isogeny on demand.

The main challenge addressed in this section is to show that W and J are *efficiently* isogenous. We exhibit efficient formulas for an isogeny $\iota : W \to J$ and efficient formulas for an isogeny $\iota' : J \to W$, and show that the composition of ι' and ι is the doubling map. Section 3 explains how we computed these formulas. Sections 4–6 tackle additional challenges in curve construction, with the goal of accelerating group operations in $E(\mathbf{F}_{p^2})$ and in $J(\mathbf{F}_p)$.

2.1. *Review of the Scholten curves*

Fix an odd prime p. Scholten's construction begins with an elliptic curve E over $\mathbf{F}_{p^2}$ of the form $y^2 = rx^3 + sx^2 + s^px + r^p$, where $r, s \in \mathbf{F}_{p^2}$. Scholten also takes two additional

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

parameters $\alpha, \beta \in \mathbf{F}_{p^2}$ such that $\alpha^{p+1} = 1$, $\beta \notin \mathbf{F}_p$, and $r(\alpha\beta^p)^6 + s(\alpha\beta^p)^4\beta^2 + s^p(\alpha\beta^p)^2\beta^4 + r^p\beta^6 \neq 0$, and observes that

$$r(\alpha - \alpha\beta^p z)^6 + s(\alpha - \alpha\beta^p z)^4(1 - \beta z)^2 + s^p(\alpha - \alpha\beta^p z)^2(1 - \beta z)^4 + r^p(1 - \beta z)^6 = \omega^2 f$$

for some nonzero $\omega \in \mathbf{F}_{p^2}$ and some monic degree-6 polynomial $f \in \mathbf{F}_p[z]$. Scholten proves that the Jacobian J of the hyperelliptic curve $y^2 = f(z)$ over $\mathbf{F}_p$ is isogenous to the Weil restriction W of E.

Note that Scholten has more parameters than necessary: replacing (r, s, α, β) with $(r\alpha^3, s\alpha, 1, \beta)$ produces an isomorphic elliptic curve and the same hyperelliptic curve. We therefore simplify the formulas by taking $\alpha = 1$: from now on

$$r(1 - \beta^p z)^6 + s(1 - \beta^p z)^4(1 - \beta z)^2 + s^p(1 - \beta^p z)^2(1 - \beta z)^4 + r^p(1 - \beta z)^6 = \omega^2 f$$

and $r(\beta^p)^6 + s(\beta^p)^4\beta^2 + s^p(\beta^p)^2\beta^4 + r^p\beta^6 \neq 0$.

Scholten showed in [48, § 3] that all elliptic curves over $\mathbf{F}_{p^2}$ with full 2-torsion are isogenous to Scholten curves. Any general security problem with the algebraic structure of Scholten curves would therefore imply serious trouble for ECC over $\mathbf{F}_{p^2}$.

Note that the characteristic polynomial for J is even, since $\chi_J(t) = \chi_E(t^2)$. This automatically implies twist-security for J: the twist of J has the same number of points as J, even though J is usually not supersingular. This does not imply twist-security for E.

2.2. *A numerical example*

We use the following cryptographically strong example as a running example throughout the paper. Most of our computations used the free Sage [49] computer-algebra system, but we are not aware of any free software for fast point-counting on elliptic curves over quadratic extensions of large prime fields, so for point-counting we used the Magma [8] computer-algebra system.

Define p as the prime $2^{127} - 309$. Note that $p \in 3 + 4\mathbf{Z}$; define $\mathbf{F}_{p^2}$ as $\mathbf{F}_p[i]/(i^2 + 1)$. Define $r = (7 + 4i)^2 = 33 + 56i$ and $s = 159 + 56i$; note that $r^p = 33 - 56i$ and $s^p = 159 - 56i$. The elliptic curve $y^2 = rx^3 + sx^2 + s^p x + r^p$ has 16ℓ points over $\mathbf{F}_{p^2}$, where ℓ is the prime number

$$180925139433306555349329664076074855364919460601081428953145528579282967992 3$$

slightly below 2^{250}, providing roughly 2^{125} security against conventional discrete-logarithm attacks. The order of ℓ in $(\mathbf{Z}/p)^*$ is $1215294167574780226654909312256315038 7$, providing ample security against index calculus. The prime factorization of the number of points on the twist of this curve over $\mathbf{F}_{p^2}$ is

$$2^2 \cdot 3 \cdot 7 \cdot 48862393571594394667013 \cdot 900162973574785449365484 1 \cdot 78350853181970659044891067 3,$$

providing roughly 2^{75} security against active twist attacks.

Define $\beta = i$ and $\omega = 5457036562574784081336510113422448183 27$. Then $\beta^2 = -1$, $(\beta^p)^2 = -1$, and $\omega^2 = -384$ in $\mathbf{F}_p$, so $r(\beta^p)^6 + s(\beta^p)^4\beta^2 + s^p(\beta^p)^2\beta^4 + r^p\beta^6 = -r - s - s^p - r^p = -384 = \omega^2$. The Scholten curve with parameters r, s, β is $y^2 = f(z)$ with $f(z) = z^6 + (7/3)z^5 - (7/4)z^4 - (14/3)z^3 + (7/4)z^2 + (7/3)z - 1$.

2.3. *Explicitly mapping W to J*

Figures 2.4 and 2.5 exhibit formulas for our efficient rational map ι from the Weil restriction of an elliptic curve to the Jacobian of a Scholten curve. See § 2.6 for a proof that this rational map is an isogeny.

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

```
R1 = ZZ
P1.<polyi> = R1[]
R2.<i> = P1.quotient(polyi^2+1)

r,s,b = 33+56*i,159+56*i,i
rp,sp,bp = 33-56*i,159-56*i,-i

ww = R1(r*bp^6+s*bp^4*b^2+sp*bp^2*b^4+rp*b^6)
R2z.<z> = R2[]
wwf = r*(1-bp*z)^6+s*(1-bp*z)^4*(1-b*z)^2+sp*(1-bp*z)^2*(1-b*z)^4+rp*(1-b*z)^6
f = wwf.change_ring(R1) / ww

P1.<X0,X1,Y0,Y1> = R1[]
P2 = P1.change_ring(R2)
X = P2(X0)+P2(X1)*i
Yw = P2(Y0)+P2(Y1)*i
curve = r*X^3+s*X^2+sp*X+rp-ww*Yw^2
curvereal = curve.map_coefficients(lambda u:u[0]).change_ring(R1)
curveimag = curve.map_coefficients(lambda u:u[1]).change_ring(R1)
assumptions = (curvereal,curveimag)*P1

u0num = (240*X0^3*Y0 + 1787*X0^2*X1*Y0 - 1248*X0*X1^2*Y0 - 297*X1^3*Y0 - 224*X0^3*Y1
  + 612*X0^2*X1*Y1 + 1860*X0*X1^2*Y1 - 876*X1^3*Y1 + 2862*X0*X1*Y0 - 1952*X0^2*Y1
  + 744*X0*X1*Y1 + 1952*X1^2*Y1 - 240*X0*Y0 + 535*X1*Y0 - 3232*X0*Y1 + 372*X1*Y1
  - 1504*Y1)
u0den = (504*X0^3*Y0 + 1339*X0^2*X1*Y0 - 984*X0*X1^2*Y0 - 745*X1^3*Y0 - 818*X0^3*Y1
  + 1620*X0^2*X1*Y1 + 1266*X0*X1^2*Y1 + 132*X1^3*Y1 - 264*X0^2*Y0 + 3758*X0*X1*Y0
  + 264*X1^2*Y0 - 1358*X0^2*Y1 - 1272*X0*X1*Y1 + 1358*X1^2*Y1 - 2040*X0*Y0 + 1879*X1*Y0
  + 818*X0*Y1 - 2652*X1*Y1 - 1272*Y0 + 1358*Y1)
u1num = 2*Y1*(-56*X0^3 - 33*X0^2*X1 - 56*X0*X1^2 - 33*X1^3 - 56*X0^2 - 66*X0*X1
  + 56*X1^2 + 56*X0 - 93*X1 + 56)
u1den = (56*X0^3*Y0 + 99*X0^2*X1*Y0 - 168*X0*X1^2*Y0 - 33*X1^3*Y0 - 66*X0^3*Y1
  + 224*X0^2*X1*Y1 + 66*X0*X1^2*Y1 + 56*X0^2*Y0 + 318*X0*X1*Y0 - 56*X1^2*Y0
  - 126*X0^2*Y1 + 126*X1^2*Y1 - 56*X0*Y0 + 159*X1*Y0 + 66*X0*Y1 - 224*X1*Y1
  - 56*Y0 + 126*Y1)
u0 = u0num / u0den
u1 = u1num / u1den
```

FIGURE 2.4. *Formulas for a rational map* $\iota : W \to J$ (*continued in Figure* 2.5).

These formulas assume that the elliptic curve is $y^2 = rx^3 + sx^2 + s^p x + r^p$ with $(r, s) = (33 + 56i, 159 + 56i)$ over $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2 + 1)$ for some prime $p \in 3 + 4\mathbf{Z}$ and that $\beta = i$; this generalizes our example from § 2.2. The Scholten curve is again $y^2 = z^6 + (7/3)z^5 - (7/4)z^4 - (14/3)z^3 + (7/4)z^2 + (7/3)z - 1$ and $\omega \in \mathbf{F}_{p^2}$ satisfies $\omega^2 = -384$.

The inputs to ι are the coordinates (X_0, X_1, Y_0, Y_1) of a point $(X_0 + X_1 i, \omega(Y_0 + Y_1 i))$ on the elliptic curve. The outputs are the Mumford coordinates (u_0, u_1, v_0, v_1) of a point on J; recall that the affine part of J is defined by the equation $(v_1 z + v_0)^2 - f \bmod z^2 + u_1 z + u_0 = 0$. What the figures display is actually a Sage script verifying that (u_0, u_1, v_0, v_1) satisfies this equation. The script takes 25 s to run using Sage 6.1.1 on an Intel Xeon E3-1275 v3.

The exceptional cases of these formulas are not obvious without further calculation. One can see from the monomials appearing in the denominators that the denominators are not generically 0 over $\mathbf{Z}$, but this does not rule out primes of bad reduction for which the denominators are always 0. We changed $\mathbf{Z}[i]/(i^2 + 1)$ to $\mathbf{F}_p[i]/(i^2 + 1)$ with $p = 2^{17} - 1$, added a check that the ideal of assumptions is prime, and ran the script again; this took 3 s. We then changed p to $2^{127} - 309$, removed the primality check (since Sage's tests for ideal primality use Singular and are limited to small characteristic), and ran the script again; this was vastly slower.

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

```
v0num = 4*(-3136*X0^5*Y0 - 5544*X0^4*X1*Y0 - 2178*X0^3*X1^2*Y0 - 3696*X0^2*X1^3*Y0
  + 958*X0*X1^4*Y0 + 1848*X1^5*Y0 + 1848*X0^5*Y1 - 5183*X0^4*X1*Y1 - 3696*X0^3*X1^2*Y1
  - 6272*X0^2*X1^3*Y1 - 5544*X0*X1^4*Y1 - 1089*X1^5*Y1 + 9472*X0^4*Y0 - 14112*X0^3*X1*Y0
  + 6534*X0^2*X1^2*Y0 - 28896*X0*X1^3*Y0 + 5250*X1^4*Y0 + 8904*X0^4*Y1 + 8316*X0^3*X1*Y1
  + 11088*X0^2*X1^2*Y1 + 128*X0*X1^3*Y1 - 12600*X1^4*Y1 + 44160*X0^3*Y0
  - 28560*X0^2*X1*Y0 + 56570*X0*X1^2*Y0 - 35280*X1^3*Y0 + 7056*X0^3*Y1
  + 19078*X0^2*X1*Y1 + 13776*X0*X1^2*Y1 + 31488*X1^3*Y1 + 55808*X0^2*Y0 - 50400*X0*X1*Y0
  + 51458*X1^2*Y0 - 7056*X0^2*Y1 + 8316*X0*X1*Y1 - 21168*X1^2*Y1 + 32704*X0*Y0
  - 30408*X1*Y0 - 8904*X0*Y1 + 6337*X1*Y1 + 8448*Y0 - 1848*Y1)
v1num = 4*(-1848*X0^5*Y0 - 3267*X0^4*X1*Y0 + 3696*X0^3*X1^2*Y0 - 10628*X0^2*X1^3*Y0
  + 5544*X0*X1^4*Y0 - 7361*X1^5*Y0 + 5314*X0^5*Y1 - 5544*X0^4*X1*Y1 + 14722*X0^3*X1^2*Y1
  - 3696*X0^2*X1^3*Y1 + 9408*X0*X1^4*Y1 + 1848*X1^5*Y1 - 16296*X0^4*Y0 + 8060*X0^3*X1*Y0
  + 33264*X0^2*X1^2*Y0 - 16504*X0*X1^3*Y0 + 5208*X1^4*Y0 + 5378*X0^4*Y1
  - 43680*X0^3*X1*Y1 + 31098*X0^2*X1^2*Y1 + 672*X0*X1^3*Y1 + 1156*X1^4*Y1
  - 50064*X0^3*Y0 + 14734*X0^2*X1*Y0 + 29232*X0*X1^2*Y0 + 19212*X1^3*Y0 - 6540*X0^3*Y1
  - 71568*X0^2*X1*Y1 - 11018*X0*X1^2*Y1 + 7728*X1^3*Y1 - 64176*X0^2*Y0 + 8060*X0*X1*Y0
  - 20496*X1^2*Y0 - 3068*X0^2*Y1 - 20832*X0*X1*Y1 - 23794*X1^2*Y1 - 34104*X0*Y0
  + 8253*X1*Y0 + 1226*X0*Y1 + 12600*X1*Y1 - 5544*Y0 - 2310*Y1)
vden = (7492*X0^6 - 18480*X0^5*X1 + 32449*X0^4*X1^2 - 7392*X0^3*X1^3 + 26046*X0^2*X1^4
  + 11088*X0*X1^5 + 1089*X1^6 + 22904*X0^5 - 9744*X0^4*X1 + 4612*X0^3*X1^2
  - 65184*X0^2*X1^3 + 14460*X0*X1^4 + 3696*X1^5 + 50688*X0^3*Y0^2 - 86016*X0^2*X1*Y0^2
  + 50688*X0*X1^2*Y0^2 - 86016*X1^3*Y0^2 + 4028*X0^4 + 101472*X0^3*X1 + 21758*X0^2*X1^2
  - 114912*X0*X1^3 + 8518*X1^4 - 50688*X0^2*Y0^2 + 172032*X0*X1*Y0^2 + 50688*X1^2*Y0^2
  - 45808*X0^3 + 84000*X0^2*X1 + 159476*X0*X1^2 - 70560*X1^3 - 345600*X0*Y0^2
  + 258048*X1*Y0^2 - 30532*X0^2 - 82992*X0*X1 + 113481*X1^2 - 244224*Y0^2 + 22904*X0
  - 74256*X1 + 19012)
v0 = v0num / vden
v1 = v1num / vden

G.<U0,U1,V0,V1,f0,f1,f2,f3,f4,f5> = ZZ[]
Gz.<z> = G.fraction_field()[]
jac = ((V1*z+V0)^2 - (z^6+f5*z^5+f4*z^4+f3*z^3+f2*z^2+f1*z+f0)) % (z^2+U1*z+U0)
jac0 = G(jac[0])
jac1 = G(jac[1])

thisjac0 = jac0(u0,u1,v0,v1,f[0],f[1],f[2],f[3],f[4],f[5])
thisjac1 = jac1(u0,u1,v0,v1,f[0],f[1],f[2],f[3],f[4],f[5])
print numerator(thisjac0) in assumptions
print numerator(thisjac1) in assumptions
print not denominator(thisjac0) in assumptions
print not denominator(thisjac1) in assumptions
```

FIGURE 2.5. *Continuation of Figure 2.4.*

See § 3 for an explanation of how we computed the polynomials that appear in Figures 2.4 and 2.5. We see no obstacle to computing analogous polynomials given any prime p, any shape of $\mathbf{F}_{p^2}$, and any choices of r, s, β in § 2.1. Presumably a larger computation along the same lines would produce a universal formula for ι (with, for example, the trace of r appearing in the universal formula at the four positions where 66 appears as a coefficient in Figure 2.4), incidentally proving that ι does in fact exist in general, but what we actually need is merely the ability to find ι for whichever curves we decide to use in § 1.

2.6. *Explicitly mapping J to W*

Our strategy for proving that ι is an isogeny is to exhibit another rational map $\iota' : J \to W$, to symbolically compute $\iota' \circ \iota$, and to observe that $\iota' \circ \iota$ matches the doubling map on W. All geometric fibers of the doubling map are nonempty and finite, so the same is true of ι' and ι. The map ι takes 0 to 0 (see below), so $0 = 2 \cdot 0 = \iota'(\iota(0)) = \iota'(0)$, so both ι and ι' are isogenies.

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

```
R.<b,bp,r,rp,s,sp,u0,u1,v0,v1> = ZZ[]
Rz.<z> = R[]
ww = r*bp^6+s*bp^4*b^2+sp*bp^2*b^4+rp*b^6
wwf = r*(1-bp*z)^6+s*(1-bp*z)^4*(1-b*z)^2+sp*(1-bp*z)^2*(1-b*z)^4+rp*(1-b*z)^6
jac = (ww*(v1*z+v0)^2 - wwf) % (z^2+u1*z+u0)
assumptions = (jac[0],jac[1])*R

bT = b+bp
bN = b*bp
D = b^2*u0+b*u1+1
Z = (bp-b)*(2*bN*u0+bT*u1+2)*D
Lw = (b^3*(u0*v0+u0*u1*v1-u1^2*v0)+3*b^2*(u0*v1-u1*v0)-3*b*v0-v1)/Z
# implicitly: L = w*Lw
F = 2*bN^2*u0^2+2*bN*bT*u0*u1+(b^2+bp^2)*u1^2-2*(b^2+bp^2-4*bN)*u0+2*bT*u1+2
X = (ww*Lw^2-s)/r - F/D^2
Yw = (bN*bp*(u0*v0+u0*u1*v1-u1^2*v0)+bp*(b+bT)*(u0*v1-u1*v0)-(bp+bT)*v0-v1)/Z-Lw*X
# implicitly: Y = w*Yw

curve = r*X^3+s*X^2+sp*X+rp-ww*Yw^2
denom = r*Z^6
print R(denom*curve) in assumptions
print not denom in assumptions
```

FIGURE 2.7. *Formulas for a rational map* $\iota' : J \to W$.

The fact that ι is an isogeny implies what we actually need in § 1: namely, ι is a group homomorphism. In particular, if our explicit rational functions for ι are defined for $P, Q, P + Q \in W(\mathbf{F}_p)$ then they produce $\iota(P), \iota(Q), \iota(P + Q) = \iota(P) + \iota(Q)$ respectively in $J(\mathbf{F}_p)$. One can directly prove this fact by a straightforward computation without any reference to the theory of isogenies. If we were applying ι to non-random inputs then we would need a complete system of formulas, supplementing our rational functions with further formulas to handle exceptional cases.

Figure 2.7 exhibits formulas for ι'. These formulas are stated in more generality than our formulas for ι: they apply to all of the curves reviewed in § 2.1. Section 3 explains how we computed these formulas. The inputs to ι' are Mumford coordinates (u_0, u_1, v_0, v_1) for a point on J, and the outputs are four coordinates for a point on W. The script actually produces (X, Y) using arithmetic over $\mathbf{F}_{p^2}$ and verifies that (X, Y) satisfies the curve equation for E; there is no need to give separate names to the four W coordinates that correspond to (X, Y). The script takes 170 s to run.

The exceptional cases of these formulas are clear from inspection, since all denominators are given in factored form as products of constants and linear functions. Specifically, there are divisions by $\beta^2 u_0 + \beta u_1 + 1$, by $2\beta\beta^p u_0 + (\beta^p + \beta)u_1 + 2$, and by the nonzero constants r and $\beta^p - \beta$.

Figure 2.8 is a Sage script verifying that applying ι to a generic point P on W, and ι' to the result, produces exactly $2P$. The script takes 78 s for $p = 2^{17} - 1$; it is much slower for $p = 2^{127} - 309$ and for $\mathbf{Z}$.

The only remaining step is to check that $\iota(0) = 0$. One tedious approach is to replace $X_0 + X_1 i$ and $Y_0 + Y_1 i$ by $(X_0 + X_1 i)/(Z_0 + Z_1 i)$ and $(Y_0 + Y_1 i)/(Z_0 + Z_1 i)$ in Figures 2.4 and 2.5, multiply numerators and denominators by appropriate powers of $Z_0 + Z_1 i$, and substitute $Y_0 + Y_1 i = 1$, obtaining rational functions defined on a patch of W that includes 0; similarly shift the patch of J to include 0; and then observe by substitution that 0 maps to 0. We avoid the shifts of formulas by taking three specific affine points $P, Q, P + Q$ for which the original formulas are defined and checking that $\iota(P) + \iota(Q) = \iota(P + Q)$. Doubling is the composition of ι' and ι, so it is also the composition of $Q \mapsto \iota'(Q + \iota(0))$ and $P \mapsto \iota(P) - \iota(0)$;

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

```
p = 2^17-1
# p = 0 for generic

if p:
  R1 = GF(p)
  P1.<polyi> = R1[]
  R2.<i> = GF(p^2,name='i',modulus=polyi^2+1)
else:
  R1 = ZZ
  P1.<polyi> = R1[]
  R2.<i> = P1.quotient(polyi^2+1)

r,s,b = 33+56*i,159+56*i,i
rp,sp,bp = 33-56*i,159-56*i,-i

ww = R1(r*bp^6+s*bp^4*b^2+sp*bp^2*b^4+rp*b^6)
R2z.<z> = R2[]
wwf = r*(1-bp*z)^6+s*(1-bp*z)^4*(1-b*z)^2+sp*(1-bp*z)^2*(1-b*z)^4+rp*(1-b*z)^6
f = wwf.change_ring(R1) / ww

P2.<X0,X1,Y0,Y1> = R2[]
X = X0+X1*i
Yw = Y0+Y1*i
curve = r*X^3+s*X^2+sp*X+rp-ww*Yw^2
if p:
  curvereal = curve.map_coefficients(lambda u:u.polynomial()[0])
  curveimag = curve.map_coefficients(lambda u:u.polynomial()[1])
else:
  curvereal = curve.map_coefficients(lambda u:u[0])
  curveimag = curve.map_coefficients(lambda u:u[1])
assumptions = (curvereal,curveimag)*P2
if p > 0 and p < 2^20: # unimplemented for large p in Sage (via Singular)
  print assumptions.is_prime()

# Weierstrass doubling:
Xorig = X
Yworig = Yw
la = (3*r*Xorig^2+2*s*Xorig+sp)/(2*ww*Yworig)
Xdbl = (ww*la^2-s)/r - 2*Xorig
Ywdbl = la*(Xorig-Xdbl) - Yworig

# isogeny from W to J:
u0num = (240*X0^3*Y0 + 1787*X0^2*X1*Y0 - 1248*X0*X1^2*Y0 - 297*X1^3*Y0 - 224*X0^3*Y1
  + 612*X0^2*X1*Y1 + 1860*X0*X1^2*Y1 - 876*X1^3*Y1 + 2862*X0*X1*Y0 - 1952*X0^2*Y1
  + 744*X0*X1*Y1 + 1952*X1^2*Y1 - 240*X0*Y0 + 535*X1*Y0 - 3232*X0*Y1 + 372*X1*Y1
  - 1504*Y1)
u0den = (504*X0^3*Y0 + 1339*X0^2*X1*Y0 - 984*X0*X1^2*Y0 - 745*X1^3*Y0 - 818*X0^3*Y1
  + 1620*X0^2*X1*Y1 + 1266*X0*X1^2*Y1 + 132*X1^3*Y1 - 264*X0^2*Y0 + 3758*X0*X1*Y0
  + 264*X1^2*Y0 - 1358*X0^2*Y1 - 1272*X0*X1*Y1 + 1358*X1^2*Y1 - 2040*X0*Y0 + 1879*X1*Y0
  + 818*X0*Y1 - 2652*X1*Y1 - 1272*Y0 + 1358*Y1)
u1num = 2*Y1*(-56*X0^3 - 33*X0^2*X1 - 56*X0*X1^2 - 33*X1^3 - 56*X0^2 - 66*X0*X1
  + 56*X1^2 + 56*X0 - 93*X1 + 56)
```

FIGURE 2.8. *Verification that doubling on W matches the composition of $\iota' : J \to W$ and $\iota : W \to J$ (continued in Figure 2.9).*

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

```
u1den = (56*X0^3*Y0 + 99*X0^2*X1*Y0 - 168*X0*X1^2*Y0 - 33*X1^3*Y0 - 66*X0^3*Y1
  + 224*X0^2*X1*Y1 + 66*X0*X1^2*Y1 + 56*X0^2*Y0 + 318*X0*X1*Y0 - 56*X1^2*Y0
  - 126*X0^2*Y1 + 126*X1^2*Y1 - 56*X0*Y0 + 159*X1*Y0 + 66*X0*Y1 - 224*X1*Y1
  - 56*Y0 + 126*Y1)
u0 = u0num / u0den
u1 = u1num / u1den
v0num = 4*(-3136*X0^5*Y0 - 5544*X0^4*X1*Y0 - 2178*X0^3*X1^2*Y0 - 3696*X0^2*X1^3*Y0
  + 958*X0*X1^4*Y0 + 1848*X1^5*Y0 + 1848*X0^5*Y1 - 5183*X0^4*X1*Y1 - 3696*X0^3*X1^2*Y1
  - 6272*X0^2*X1^3*Y1 - 5544*X0*X1^4*Y1 - 1089*X1^5*Y1 + 9472*X0^4*Y0 - 14112*X0^3*X1*Y0
  + 6534*X0^2*X1^2*Y0 - 28896*X0*X1^3*Y0 + 5250*X1^4*Y0 + 8904*X0^4*Y1 + 8316*X0^3*X1*Y1
  + 11088*X0^2*X1^2*Y1 + 128*X0*X1^3*Y1 - 12600*X1^4*Y1 + 44160*X0^3*Y0
  - 28560*X0^2*X1*Y0 + 56570*X0*X1^2*Y0 - 35280*X1^3*Y0 + 7056*X0^3*Y1
  + 19078*X0^2*X1*Y1 + 13776*X0*X1^2*Y1 + 31488*X1^3*Y1 + 55808*X0^2*Y0 - 50400*X0*X1*Y0
  + 51458*X1^2*Y0 - 7056*X0^2*Y1 + 8316*X0*X1*Y1 - 21168*X1^2*Y1 + 32704*X0*Y0
  - 30408*X1*Y0 - 8904*X0*Y1 + 6337*X1*Y1 + 8448*Y0 - 1848*Y1)
v1num = 4*(-1848*X0^5*Y0 - 3267*X0^4*X1*Y0 + 3696*X0^3*X1^2*Y0 - 10628*X0^2*X1^3*Y0
  + 5544*X0*X1^4*Y0 - 7361*X1^5*Y0 + 5314*X0^5*Y1 - 5544*X0^4*X1*Y1 + 14722*X0^3*X1^2*Y1
  - 3696*X0^2*X1^3*Y1 + 9408*X0*X1^4*Y1 + 1848*X1^5*Y1 - 16296*X0^4*Y0 + 8060*X0^3*X1*Y0
  + 33264*X0^2*X1^2*Y0 - 16504*X0*X1^3*Y0 + 5208*X1^4*Y0 + 5378*X0^4*Y1
  - 43680*X0^3*X1*Y1 + 31098*X0^2*X1^2*Y1 + 672*X0*X1^3*Y1 + 1156*X1^4*Y1
  - 50064*X0^3*Y0 + 14734*X0^2*X1*Y0 + 29232*X0*X1^2*Y0 + 19212*X1^3*Y0 - 6540*X0^3*Y1
  - 71568*X0^2*X1*Y1 - 11018*X0*X1^2*Y1 + 7728*X1^3*Y1 - 64176*X0^2*Y0 + 8060*X0*X1*Y0
  - 20496*X1^2*Y0 - 3068*X0^2*Y1 - 20832*X0*X1*Y1 - 23794*X1^2*Y1 - 34104*X0*Y0
  + 8253*X1*Y0 + 1226*X0*Y1 + 12600*X1*Y1 - 5544*Y0 - 2310*Y1)
vden = (7492*X0^6 - 18480*X0^5*X1 + 32449*X0^4*X1^2 - 7392*X0^3*X1^3 + 26046*X0^2*X1^4
  + 11088*X0*X1^5 + 1089*X1^6 + 22904*X0^5 - 9744*X0^4*X1 + 4612*X0^3*X1^2
  - 65184*X0^2*X1^3 + 14460*X0*X1^4 + 3696*X1^5 + 50688*X0^3*Y0^2 - 86016*X0^2*X1*Y0^2
  + 50688*X0*X1^2*Y0^2 - 86016*X1^3*Y0^2 + 4028*X0^4 + 101472*X0^3*X1 + 21758*X0^2*X1^2
  - 114912*X0*X1^3 + 8518*X1^4 - 50688*X0^2*Y0^2 + 172032*X0*X1*Y0^2 + 50688*X1^2*Y0^2
  - 45808*X0^3 + 84000*X0^2*X1 + 159476*X0*X1^2 - 70560*X1^3 - 345600*X0*Y0^2
  + 258048*X1*Y0^2 - 30532*X0^2 - 82992*X0*X1 + 113481*X1^2 - 244224*Y0^2 + 22904*X0
  - 74256*X1 + 19012)
v0 = v0num / vden
v1 = v1num / vden

# isogeny from J to W:
bT = b+bp
bN = b*bp
D = b^2*u0+b*u1+1
Z = (bp-b)*(2*bN*u0+bT*u1+2)*D
Lw = (b^3*(u0*v0+u0*u1*v1-u1^2*v0)+3*b^2*(u0*v1-u1*v0)-3*b*v0-v1)/Z
F = 2*bN^2*u0^2+2*bN*bT*u0*u1+(b^2+bp^2)*u1^2-2*(b^2+bp^2-4*bN)*u0+2*bT*u1+2
X = (ww*Lw^2-s)/r - F/D^2
Yw = (bN*bp*(u0*v0+u0*u1*v1-u1^2*v0)+bp*(b+bT)*(u0*v1-u1*v0)-(bp+bT)*v0-v1)/Z-Lw*X

Xequal = X - Xdbl
Ywequal = Yw - Ywdbl
print numerator(Xequal) in assumptions
print numerator(Ywequal) in assumptions
print not denominator(Xequal) in assumptions
print not denominator(Ywequal) in assumptions
```

FIGURE 2.9. *Continuation of Figure 2.8.*

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

the second map $P \mapsto \iota(P) - \iota(0)$ is now forced to be an isogeny, so it is a group homomorphism, so $\iota(P+Q) = \iota(P) + \iota(Q) - \iota(0)$.

3. *Finding efficient isogenies*

We now discuss various algorithmic issues that arose in finding ι and ι', that is, computing (not evaluating!) the polynomials that appear in Figures 2.4, 2.5, and 2.7. We emphasize here that *finding* and *verifying* are not the same task. By separating these tasks we accelerated both tasks: we were free to take, and did take, many unproven steps in finding ι and ι'.

3.1. *The covering map*

Define a rational map ϕ from the hyperelliptic curve $H : y^2 = f(z)$ to the elliptic curve $E : y^2 = rx^3 + sx^2 + s^p x + r^p$, namely $\phi(z, y) = (x^2, \omega y/(1-\beta z)^3)$ where $x = (1-\beta^p z)/(1-\beta z)$. To see that this works, observe that $\omega^2 f(z)/(1-\beta z)^6 = rx^6 + sx^4 + s^p x^2 + r^p$ by definition of f. The map ϕ, modulo notation, appeared in Scholten's proof of [48, Lemma 2.1].

Next define a rational map ϕ_2 from $H \times H$ to W as follows: map (P_1, P_2) to the sum $\phi(P_1) + \phi(P_2)$ on E, and then to the coordinates of the sum in W. These coordinates are symmetric between P_1 and P_2, so ϕ_2 must factor as a composition of the standard map $H \times H \to J$ and some rational map $\iota' : J \to W$.

Of course, a rational map from J to W is not necessarily an isogeny. The map might shift 0 to something nonzero (which would not be a disaster for us), or it might lose one or two dimensions (which would be a disaster). On the other hand, it is at least intuitively clear that 0 maps to 0, since $\phi(z, -y) = -\phi(z, y)$. Furthermore, if $\#E(\mathbf{F}_{p^2})$ has a large prime divisor (not far below p^2), as often happens, then one expects a 'random' size-p subset S of $E(\mathbf{F}_{p^2})$ to have $S + S$ covering a considerable fraction of $E(\mathbf{F}_{p^2})$, while any drop of dimension would make $\#\iota'(J(\mathbf{F}_p))$ much smaller than $\#W(\mathbf{F}_p) = \#E(\mathbf{F}_{p^2})$ for large p. Not all subsets are 'random' (for example, p consecutive multiples of a generator have only $2p-1$ sums), but the algebraic constraints on $\phi(H)$ seem unlikely to produce such behavior. So it is reasonable to hope that ι' is an isogeny.

3.2. *The hard approach*

At this point the conventional analysis of isogenies would continue by carrying out various time-consuming computations:

- Prove that ι' really is an isogeny. The main work here is analyzing the fibers of ι' via the fibers of ϕ_2.
- Deduce that, for various positive integers d, multiplication by d on W can be expressed as $\iota' \circ \iota$, and multiplication by d on J can be expressed as $\iota \circ \iota'$, where ι is an isogeny. Figure out the smallest possible d by comparing the structure of the fibers of ι' to the group structure of J.
- Compute explicit formulas for ι' as follows. Start with generic points $P_1 = (z_1, y_1)$ and $P_2 = (z_2, y_2)$ on H, in other words, the points (z_1, y_1) and (z_2, y_2) on H over $\mathbf{F}_p(z_1, z_2)[y_1, y_2]/(y_1^2 - f(z_1), y_2^2 - f(z_2))$. Compose the definition of ϕ with the addition formulas on E to obtain $\phi(P_1) + \phi(P_2)$ as explicit rational functions in z_1, z_2, y_1, y_2. Eliminate z_1, z_2, y_1, y_2 in favor of the Mumford coordinates $u_0 = z_1 z_2$, $u_1 = -z_1 - z_2$, $v_1 = (y_2 - y_1)/(z_2 - z_1)$, $v_0 = y_1 - v_1 z_1$.
- Observe that these explicit formulas for ι' involve many terms. Search for simpler formulas, presumably accelerating evaluation of ι' and also accelerating the rest of the analysis, by strategically exploiting equations satisfied by the Mumford coordinates. See [42] for a systematic 'rational simplification' algorithm; see [30] for the first use of this algorithm to simplify elliptic-curve formulas.

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

- View $d(u_0, u_1, v_0, v_1) = \iota(\iota'(u_0, u_1, v_0, v_1))$, or the analogous equation on W, as a system of equations for the dual isogeny ι. Solve these equations somehow.

One could carry out this type of analysis for specific choices of the parameters p, r, s, β, obtaining formulas for ι' and ι for those parameters, which is what we actually need. Alternatively, with more computation, one could leave the parameters as variables, obtaining general formulas for ι' and ι and then specializing the formulas upon demand.

One way to map $P \in W$ to J is to compute all the preimages of P under ϕ, compute the sum of the preimages, and then take the trace of the sum. This 'norm–conorm' map is studied in, for example, [**26**, § 3], [**14**, § 3], and [**2**, § 2.1]. One can show, starting from the fact that ι' is an isogeny, that applying ι' to the trace produces exactly $2P$, so this map is exactly ι, and $d = 2$. One can obtain explicit formulas for this map from explicit formulas for addition on J, and one can then search for simpler formulas as above.

3.3. *The easy approach*

We take a different, much easier, approach to compute formulas for ι'. We fix parameters, take *random* points $(z_1, y_1), (z_2, y_2) \in H(\mathbf{F}_p)$, compute the corresponding Mumford coordinates u_0, u_1, v_0, v_1 in $\mathbf{F}_p$ (skipping the degenerate case $z_1 = z_2$), and compute $\phi(z_1, y_1) + \phi(z_2, y_2)$ as two coordinates in $\mathbf{F}_{p^2}$, in other words, four coordinates in $\mathbf{F}_p$. This computation tells us a specific value of ι' for a specific input (u_0, u_1, v_0, v_1); this linearly constrains the coefficients in the numerator and denominator of each coordinate of ι'. Taking more random points gives us more linear constraints. For each coordinate we guess a limit on the degree (or a more refined set of monomials) for the smallest possible numerator and denominator; take significantly more points than monomials; and solve the resulting system of linear equations. If there are enough points then all nonzero solutions will define the same rational map, and if the guess was correct then this rational map must be ι'.

The same idea easily produces ι. We start by guessing that $d = 2$ will work, that is, that we will be able to find ι with $2(u_0, u_1, v_0, v_1) = \iota(\iota'(u_0, u_1, v_0, v_1))$. For random points $(z_1, y_1), (z_2, y_2) \in H(\mathbf{F}_p)$, we compute (u_0, u_1, v_0, v_1) and $\iota'(u_0, u_1, v_0, v_1)$ as above, and also double (u_0, u_1, v_0, v_1) on J (skipping degenerate cases) to obtain $2(u_0, u_1, v_0, v_1)$. This tells us an input–output pair for ι, and thus linearly constrains the coefficients in the numerator and denominator of each coordinate of ι.

Of course, this does not *prove* that the formulas that we obtain are the same as the ι' and ι defined in this section. It is conceivable that our formulas were amazingly lucky, matching ι' and ι on many random points, while the real ι' and ι escaped detection by requiring monomials of higher degree than the monomials in our computation. One response is that, having verified (see § 2.6) that our formulas are in fact efficient dual isogenies, we can simply use these formulas, and no longer need this section's definition of a possibly different function ι'. A different response is to verify symbolically that the functions are in fact the same. Figure 3.4 does exactly this; the script takes 24 s to run.

The interpolation strategy described in this section can be viewed as a way to simplify formulas for ι' (and ι). We emphasize, however, that the input to interpolation does not need to be a *formula* for ι'; it can be any method of *computing* enough input–output pairs for ι'. This is what lets us skip all of the intermediate computations in $\mathbf{F}_p(z_1, z_2)[y_1, y_2]/(y_1^2 - f(z_1), y_2^2 - f(z_2))$. We also comment that, even though all we need is to be able to compute ι' and ι efficiently for specific curve parameters, one can also interpolate generic formulas that work for arbitrary parameters. This is what we did for ι', producing the extra generality of Figure 2.7 compared to Figures 2.4 and 2.5.

We actually deviated slightly from the strategy stated above: rather than interpolating formulas for ι', we interpolated formulas for certain intermediate results that obviously play

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

```
R.<r,s,w,b,bp,u0,u1,v0,v1,z1,z2,y1,y2> = ZZ[]
assumptions = (u0-z1*z2,u1+z1+z2,v1*(z1-z2)-(y1-y2),y1-v1*z1-v0)*R

x1 = (1-bp*z1)/(1-b*z1)
x2 = (1-bp*z2)/(1-b*z2)
sumxin = r*x1^2+r*x2^2
la = (r*w*y1/(1-b*z1)^3-r*w*y2/(1-b*z2)^3)/(r*x1^2-r*x2^2)
x3 = (la^2-s-sumxin)/r
y3 = la*(x1^2-x3)-w*y1/(1-b*z1)^3

ww = w*w
bT = b+bp
bN = b*bp
D = b^2*u0+b*u1+1
Z = (bp-b)*(2*bN*u0+bT*u1+2)*D
Lw = (b^3*(u0*v0+u0*u1*v1-u1^2*v0)+3*b^2*(u0*v1-u1*v0)-3*b*v0-v1)/Z
L = w*Lw
F = 2*bN^2*u0^2+2*bN*bT*u0*u1+(b^2+bp^2)*u1^2-2*(b^2+bp^2-4*bN)*u0+2*bT*u1+2
X = (ww*Lw^2-s)/r - F/D^2
Yw = (bN*bp*(u0*v0+u0*u1*v1-u1^2*v0)+bp*(b+bT)*(u0*v1-u1*v0)-(bp+bT)*v0-v1)/Z-Lw*X
Y = w*Yw

denom = Z*(z2-z1)*(bp-b)*(1-b*z2)*(1-b*z1)*(2*b*bp*z1*z2-(b+bp)*(z1+z2)+2)
print R(denom*(la-L)) in assumptions
print R(denom^2*(sumxin-r*F/D^2)) in assumptions
print R(r*denom^2*(x3-X)) in assumptions
check = r*denom^3*y3-r*denom^3*Y
# R(check) segfaults
print denominator(check) == 1
print R(numerator(check)) in assumptions
print not r*denom^3 in assumptions
```

FIGURE 3.4. *Verification that the formulas in Figure 2.7 map a generic element of J with affine part $(P_1)+(P_2)$ to $\phi(P_1)+\phi(P_2)$.*

an important role in ι' and that remain visible in Figure 2.7. Specifically, $F/D^2 = x_1^2 + x_2^2$ where $x_i = (1-\beta^p z_i)/(1-\beta z_i)$, and L is the slope in the usual Weierstrass-curve addition formulas.

4. *Jacobian to Kummer*

In § 2 we constructed efficient isogenies between W and J, where W is the Weil restriction of an elliptic curve from $\mathbf{F}_{p^2}$ to $\mathbf{F}_p$ and J is the Jacobian of a hyperelliptic curve $y^2 = f(z)$ in Scholten form over $\mathbf{F}_p$.

We now restrict the choice of hyperelliptic curve to improve the efficiency of scalar multiplication in $J(\mathbf{F}_p)$. Specifically, in this section, we force the corresponding Kummer surface K to be defined over $\mathbf{F}_p$, allowing the action of $\mathbf{Z}$ on the group $J(\mathbf{F}_p)$ to be computed via Gaudry's highly efficient formulas for the action of $\mathbf{Z}$ on the $\mathbf{Z}$-set $K(\mathbf{F}_p)$, that is, for the standard scalar-multiplication function $\mathbf{Z} \times K(\mathbf{F}_p) \to K(\mathbf{F}_p)$. See § 6 for further curve constraints that save time inside Gaudry's formulas.

4.1. *Constraints on f*

We reject Scholten's sextic polynomial f unless it splits completely over $\mathbf{F}_p$. Rather than testing this we construct f in a way that enforces it; see § 4.2.

Once we have an f that splits, we convert the Scholten curve to twisted Rosenhain form $\delta y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu)$ over $\mathbf{F}_p$, by defining z as a linear fractional transformation

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

of x that moves three roots of f in z to $0, 1, \infty$ in x. This transformation of curves induces a transformation of the Jacobians. Note that there are many choices of three roots and thus many choices of Rosenhain curve.

Explicitly, write three distinct roots of f in the form T_{12}/T_{22}, $(T_{11}+T_{12})/(T_{21}+T_{22})$, T_{11}/T_{21}. Define $g(x) = (T_{21}x + T_{22})^6 f((T_{11}x + T_{12})/(T_{21}x + T_{22}))$. Then g is a polynomial of degree 5 with distinct roots $0, 1, \lambda, \mu, \nu$; that is, $\delta g = x(x-1)(x-\lambda)(x-\mu)(x-\nu)$ for some δ. If (z, Y) is a point on the Scholten curve $Y^2 = f(z)$ and $z \neq T_{11}/T_{21}$ then $(x, Y(T_{21}x + T_{22})^3)$ is a point on the twisted Rosenhain curve $\delta y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu)$, where $x = (T_{22}z - T_{12})/(-T_{21}z + T_{11})$. The interpolation approach of § 3 efficiently computes formulas for the corresponding map between Jacobians.

We then obtain the Kummer surface corresponding to (λ, μ, ν) as in [6, full version, § 5.2]: compute $d^2 = 1$, $c^2 = \pm\sqrt{\lambda\mu/\nu}$, $b^2 = \sqrt{\mu(\mu-1)(\lambda-\nu)/(\nu(\nu-1)(\lambda-\mu))}$, and $a^2 = b^2c^2\nu/\mu$. We reject (λ, μ, ν) if these two square roots are not in $\mathbf{F}_p$. Note that there are six choices of (λ, μ, ν) for each Rosenhain curve.

We also check the 'genericity conditions' hypothesized by [25]. Specifically, we check that the quantities $a^2d^2-b^2c^2$, $a^2c^2-b^2d^2$, $a^2b^2-c^2d^2$, $A^2 = a^2+b^2+c^2+d^2$, $B^2 = a^2+b^2-c^2-d^2$, $C^2 = a^2-b^2+c^2-d^2$, $D^2 = a^2-b^2-c^2+d^2$ are nonzero. This ensures that the denominators are nonzero in the quantities E, F, G, H appearing below. (The conditions stated in [25] are that A, B, C, D are nonzero and that various theta constants $\theta_5, \theta_6, \theta_7, \theta_8, \theta_9, \theta_{10}$ are all nonzero when $(\theta_1 : \theta_2 : \theta_3 : \theta_4) = (a : b : c : d)$. The formula $\theta_5^2\theta_6^2 = \theta_1^2\theta_4^2 - \theta_2^2\theta_3^2$ shows that if $\theta_5 = 0$ or $\theta_6 = 0$ then $a^2d^2 - b^2c^2 = 0$; similar comments apply to $\theta_7^2\theta_9^2 = \theta_1^2\theta_3^2 - \theta_2^2\theta_4^2$ and $\theta_8^2\theta_{10}^2 = \theta_1^2\theta_2^2 - \theta_3^2\theta_4^2$.)

Beware that the reverse formulas for λ, μ, ν in terms of a, b, c, d, A, B, C, D in [6, full version, § 5.2] are correct for our definitions of A, B, C, D but incorrect for the definitions of A, B, C, D in [6]. Further warnings regarding formulas in the literature appear in § 4.4.

This procedure forces J to have full 2-torsion. Consequently the group order $\#J(\mathbf{F}_p)$ is divisible by 16. For cryptographic purposes we need a large prime in the group order; we restrict attention to the simplest case, namely groups of order 16ℓ where ℓ is a large prime. Our numerical example in § 2.2 shows that this case does occur; we return to this example in § 4.3.

4.2. *Scholten Jacobians with full 2-torsion*

By hypothesis $y^2 = rx^3 + sx^2 + s^px + r^p$ is elliptic. Consequently $r \neq 0$, and the cubic $rx^3 + sx^2 + s^px + r^p$ factors over an extension of $\mathbf{F}_p$ as $r(x-\rho_1)(x-\rho_2)(x-\rho_3)$ for distinct ρ_1, ρ_2, ρ_3. The product $-r\rho_1\rho_2\rho_3$ equals r^p so all of ρ_1, ρ_2, ρ_3 are nonzero. Choose a square root $\sqrt{\rho_j}$ of each ρ_j in a suitable extension of $\mathbf{F}_p$.

Now assume that the Jacobian of Scholten's hyperelliptic curve has full 2-torsion defined over $\mathbf{F}_p$, that is, that there are 6 distinct roots in $\mathbf{F}_p$ of the degree-6 polynomial

$$\begin{aligned} &r(1-\beta^pz)^6 + s(1-\beta^pz)^4(1-\beta z)^2 + s^p(1-\beta^pz)^2(1-\beta z)^4 + r^p(1-\beta z)^6 \\ &\quad = r((1-\beta^pz)^2 - \rho_1(1-\beta z)^2)((1-\beta^pz)^2 - \rho_2(1-\beta z)^2)((1-\beta^pz)^2 - \rho_3(1-\beta z)^2). \end{aligned}$$

This polynomial visibly splits into linear factors of the form $(1-\beta^pz \pm \sqrt{\rho_j}(1-\beta z))$, so each $(1\pm\sqrt{\rho_j})/(\beta^p \pm \beta\sqrt{\rho_j})$ must be in $\mathbf{F}_p$; in other words, each $\sqrt{\rho_j}$ has the form $(1-\beta^p\zeta)/(1-\beta\zeta)$ for some $\zeta \in \mathbf{F}_p$. This implies $\sqrt{\rho_j} \in \mathbf{F}_{p^2}$ and $\sqrt{\rho_j}^p = (1-\beta\zeta)/(1-\beta^p\zeta) = 1/\sqrt{\rho_j}$; that is, each $\sqrt{\rho_j}$ has norm 1.

Conversely, take any three norm-1 elements $\sqrt{\rho_1}, \sqrt{\rho_2}, \sqrt{\rho_3} \in \mathbf{F}_{p^2}$ with distinct squares. The product $-\rho_1\rho_2\rho_3$ has norm 1 and thus can be written as r^p/r for some $r \in \mathbf{F}_{p^2}^*$, for example for $r = i(\sqrt{\rho_1}\sqrt{\rho_2}\sqrt{\rho_3})^p$. Define $s = -r(\rho_1+\rho_2+\rho_3)$; then $rx^3+sx^2+s^px+r^p = r(x-\rho_1)(x-\rho_2)(x-\rho_3)$ so $y^2 = rx^3 + sx^2 + s^px + r^p$ is elliptic. Choose any $\beta \in \mathbf{F}_{p^2}$ such that $\beta \notin \mathbf{F}_p$ and

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

$\beta^{p-1} \neq \pm\sqrt{\rho_j}$. The ratio $(1 \pm \sqrt{\rho_j})/(\beta^p \pm \beta\sqrt{\rho_j})$ has conjugate $(1 \pm 1/\sqrt{\rho_j})/(\beta \pm \beta^p/\sqrt{\rho_j}) = (\sqrt{\rho_j} \pm 1)/(\beta\sqrt{\rho_j} \pm \beta^p) = (1 \pm \sqrt{\rho_j})/(\beta^p \pm \beta\sqrt{\rho_j})$ and is thus in $\mathbf{F}_p$. There are six such ratios, all distinct since $z \mapsto (1 - \beta^p z)/(1 - \beta z)$ maps them to $\pm\sqrt{\rho_j}$, and each of these ratios is a root of Scholten's degree-6 polynomial.

4.3. *A numerical example, continued*

As a generalization of the example in § 2.2, take $p \in 3 + 4\mathbf{Z}$ with $\mathbf{F}_{p^2} = \mathbf{F}_p[i]/(i^2 + 1)$, assume $p > 13$, and take $\sqrt{\rho_1} = i$, $\sqrt{\rho_2} = (3 + 4i)/5$, and $\sqrt{\rho_3} = (5 + 12i)/13$. The product $-\rho_1\rho_2\rho_3 = -(2047 + 3696i)/4225$ then has the form r^p/r for, for example, $r = 33 + 56i$. Define $s = -r(\rho_1 + \rho_2 + \rho_3) = 159 + 56i$. This is how we constructed the pair (r, s) in § 2.2. Our choice $\beta = i$ has $\beta^{p-1} = -1$, avoiding $\pm\sqrt{\rho_j}$.

This structure forces the polynomial f in § 2.2 to split over $\mathbf{F}_p$: specifically, $z^6 + (7/3)z^5 - (7/4)z^4 - (14/3)z^3 + (7/4)z^2 + (7/3)z - 1$ has roots 1 and -1 via $\sqrt{\rho_1}$, roots $1/2$ and -2 via $\sqrt{\rho_2}$, and roots $2/3$ and $-3/2$ via $\sqrt{\rho_3}$.

The linear fractional transformation $z \mapsto 5(1 - z)/(2 + z)$ takes $1, 1/2, -2, -1, 2/3, -3/2$ to $0, 1, \infty, \lambda, \mu, \nu$ respectively, where $\lambda = 10$, $\mu = 5/8$, $\nu = 25$. The ratio $\lambda\mu/\nu$ is a square, namely $1/2^2$, and the ratio $\mu(\mu - 1)(\lambda - \nu)/(\nu(\nu - 1)(\lambda - \mu))$ is a square, namely $1/40^2$. Taking positive signs for the square roots produces $(a^2, b^2, c^2, d^2) = (1/2, 1/40, 1/2, 1)$ and $(A^2, B^2, C^2, D^2) = (81/40, -39/40, -1/40, 39/40)$; we scale these to $(20, 1, 20, 40)$ and $(81, -39, -1, 39)$, respectively. The differences $a^2b^2 - c^2d^2$, $a^2c^2 - b^2d^2$, $a^2d^2 - b^2c^2$ are nonzero.

4.4. *Explicit maps from the Jacobian to the Kummer surface*

It is not easy to find correct formulas in the literature for the standard rational map from a Jacobian J of a genus-2 hyperelliptic curve to a Kummer surface K. The conventional view arises from expressing J (in Mumford coordinates) and K (a particular quartic surface with various symmetries) in terms of 16 different Riemann theta functions and then solving for the coordinates of one in terms of the other; but this involves a huge thicket of theta formulas, with many opportunities for errors. For example:

- The formula for $\theta_7^4 + \theta_9^4$ in [**25**, p. 262] is incorrect: it needs to be negated.
- The formula for v_0^2 in [**12**, p. 1206] is incorrect: the second minus sign needs to be a plus, as pointed out in [**6**].
- The definitions of A, B, C, D in [**6**, § 5.1] are incorrect for the stated parameter relationship between J and K: they need to be replaced by A^2, B^2, C^2, D^2. On the other hand, the definitions are consistent with some other formulas in [**6**], so those other formulas also need to be modified.

We need to actually compute this map, rather than merely to write papers about it, so we need correct formulas. To avoid errors we present in Figure 4.5 a Sage script verifying our formulas for this map. We have also put considerable effort into simplifying these formulas, eliminating unnecessary detours through theta functions. The script takes 70 s to run, and the formulas have bad reduction only at 2.

Computer-algebra scripts for 'Kummer surface' formulas have been published before: see the web site [**20**] accompanying the book [**10**] by Cassels and Flynn. However, the 'Kummer surface' K' in [**10**, **20**] is much less efficient than the highly symmetric Kummer surface K used in [**4**, **6**, **11**, **12**, **25**] and used in this paper. (We question the use of the terminology 'Kummer surface' for K'; we speculate that Kummer would have been horrified to have his name attached to K'.) Both K and K' are isomorphic to $J/\{\pm 1\}$ and therefore to each other, but the choice of coordinates is critical for performance, and there is no reason to think that finding formulas for this isomorphism from K' to K would be easier than finding formulas for the map from J to K.

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

```
R.<u0,u1,v0,v1,la,mu,nu,A,B,C,D,a,b,c,d,twist> = ZZ[]
Rz.<z> = R.fraction_field()[]
jac = (twist*(v1*z+v0)^2 - z*(z-1)*(z-la)*(z-mu)*(z-nu)) % (z^2+u1*z+u0)

assumptions = (
  A^2-(a^2+b^2+c^2+d^2),
  B^2-(a^2+b^2-c^2-d^2),
  C^2-(a^2-b^2+c^2-d^2),
  D^2-(a^2-b^2-c^2+d^2),
  a^2*c^2-la*b^2*d^2,
  mu*d^2*(A*B-C*D)-c^2*(A*B+C*D),
  nu*b^2*(A*B-C*D)-a^2*(A*B+C*D),
  R(jac[0]),R(jac[1])
)*R

U = a^2*b^2-c^2*d^2
I = (a^2*d^2-b^2*c^2)*(a^2*c^2-b^2*d^2)*U
E = a*b*c*d*A^2*B^2*C^2*D^2 / I
F = (a^4-b^4-c^4+d^4) / (a^2*d^2-b^2*c^2)
G = (a^4-b^4+c^4-d^4) / (a^2*c^2-b^2*d^2)
H = (a^4+b^4-c^4-d^4) / U
X0 =     c^2*(a^2*b^2*c^2+a^4*d^2+b^4*d^2-2*c^4*d^2-d^6)*nu/(U*d^4)    -a^2*c^2/(b^2*d^2)
Y0 = a^2*c^2*(a^4*c^2+b^4*c^2-c^6+a^2*b^2*d^2-2*c^2*d^4)*nu/(U*b^2*d^4)-a^4*c^4/(b^4*d^4)
Z0 =     c^2*(2*a^4*b^2+b^6-b^2*c^4-a^2*c^2*d^2-b^2*d^4)*nu/(U*b^2*d^2)-a^2*c^2/(b^2*d^2)
T0 =     c^4*(a^6+2*a^2*b^4-a^2*c^4-b^2*c^2*d^2-a^2*d^4)*nu/(U*b^2*d^4)-a^4*c^4/(b^4*d^4)
X1 = 2*nu +   nu*b^2*c^2/(a^2*d^2) + 2*a^2*c^2/(b^2*d^2) + 1
Y1 =   nu + 2*nu*b^2*c^2/(a^2*d^2) +   a^2*c^2/(b^2*d^2) + 2
Z1 =   nu + 2*nu*b^2*c^2/(a^2*d^2) + 2*a^2*c^2/(b^2*d^2) + 1
T1 = 2*nu +   nu*b^2*c^2/(a^2*d^2) +   a^2*c^2/(b^2*d^2) + 2
V = a^4*b^4*c^4+a^4*b^4*d^4-a^4*c^4*d^4-b^4*c^4*d^4+2*a^2*b^2*c^2*d^2*H*U
s = u0^2-2*u0*u1^2-2*twist*v0*v1-(nu*V/(U*a^2*b^2*d^4))*u0+(nu*H*a^2/b^2-a^4/b^4)*c^4/d^4
x = a^2*(s+X0*u1-X1*u0*u1+nu*(b^2*c^2/(a^2*d^2))*u1^2)
y = b^2*(s+Y0*u1-Y1*u0*u1+nu*(a^2*c^2/(b^2*d^2))*u1^2)
z = c^2*(s+Z0*u1-Z1*u0*u1+nu*u1^2)
t = d^2*(s+T0*u1-T1*u0*u1+nu*(c^4/d^4)*u1^2)

x = R(a^2*b^4*d^4*U*x)
y = R(a^2*b^4*d^4*U*y)
z = R(a^2*b^4*d^4*U*z)
t = R(a^2*b^4*d^4*U*t)
EI = R(E*I)
FI = R(F*I)
GI = R(G*I)
HI = R(H*I)
C = 4*EI^2*x*y*z*t-(FI*(x*t+y*z)+GI*(x*z+y*t)+HI*(x*y+z*t)-I*(x^2+y^2+z^2+t^2))^2
print 4*C in assumptions
```

FIGURE 4.5. *Formulas to map the Jacobian of a Rosenhain curve to a Kummer surface, assuming certain relationships between the surface parameters and the curve parameters.*

Our script starts with a generic point (u_0, u_1, v_0, v_1) in Mumford coordinates on the Jacobian of a twisted Rosenhain curve $\delta Y^2 = X(X-1)(X-\lambda)(X-\mu)(X-\nu)$ with $0, 1, \lambda, \mu, \nu$ distinct. Recall, as in § 2, that the affine part of the Jacobian is defined by the equation

$$\delta(v_1X + v_0)^2 - X(X-1)(X-\lambda)(X-\mu)(X-\nu) \bmod X^2 + u_1X + u_0 = 0.$$

The script computes particular linear combinations x, y, z, t of $u_0^2, u_0u_1^2, v_0v_1, u_0, 1, u_1, u_0u_1, u_1^2$, and verifies that $(x : y : z : t)$ satisfies the Kummer-surface equation

$$4E^2xyzt = (F(xt+yz) + G(xz+yt) + H(xy+zt) - (x^2+y^2+z^2+t^2))^2,$$

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

assuming certain relationships between the Kummer-surface parameters E, F, G, H and the Rosenhain-curve parameters λ, μ, ν. The assumptions are

$$\lambda = \frac{b^2d^2}{a^2c^2}, \qquad F = \frac{a^4 - b^4 - c^4 + d^4}{a^2d^2 - b^2c^2}, \qquad A^2 = a^2 + b^2 + c^2 + d^2,$$
$$\mu = \frac{c^2(AB + CD)}{d^2(AB - CD)}, \qquad G = \frac{a^4 - b^4 + c^4 - d^4}{a^2c^2 - b^2d^2}, \qquad B^2 = a^2 + b^2 - c^2 - d^2,$$
$$\nu = \frac{a^2(AB + CD)}{b^2(AB - CD)}, \qquad H = \frac{a^4 + b^4 - c^4 - d^4}{a^2b^2 - c^2d^2}, \qquad C^2 = a^2 - b^2 + c^2 - d^2,$$
$$E = \frac{abcdA^2B^2C^2D^2}{(a^2d^2 - b^2c^2)(a^2c^2 - b^2d^2)(a^2b^2 - c^2d^2)}, \qquad D^2 = a^2 - b^2 - c^2 + d^2.$$

The formulas above were typed by hand; readers are encouraged to instead consult Figure 4.5 for the original computer-verified formulas, including the details of the linear combinations.

Since the map does not use v_0 and v_1 except as v_0v_1, it does not distinguish $-(u_0, u_1, v_0, v_1) = (u_0, u_1, -v_0, -v_1)$ from (u_0, u_1, v_0, v_1). With more work one can verify that a generic output point $(x : y : z : t)$ has exactly two preimages, but we do not actually need this fact. For example, it would not be a problem if the map were actually doubling on J followed by the standard map from $J/\{\pm 1\}$ to K, since this would still act as a nonzero $\mathbf{Z}$-set morphism from the order-ℓ subgroup of $J(\mathbf{F}_p)$ to the corresponding subset of $K(\mathbf{F}_p)$. Given any particular curve, we check that a generator of the order-ℓ subgroup maps to a nonzero element of $K(\mathbf{F}_p)$.

4.6. *Explicit maps from the Kummer surface to the Jacobian*

We do not report similarly optimized computer-verified formulas for computing preimages of the map in Figure 4.5. These preimages are not needed in § 1. However, we briefly comment on such computations for applications that might need them.

A rational map cannot compute the preimages in J for a given element of K: the choice between two preimages is determined by a sign choice in a square root. This square root is, however, unnecessary for an application that actually wants to compute $P \mapsto nP$ on J. There is a rational map that produces nP in J given P in J and the images of $nP, (n+1)P$ in K, and Gaudry's formulas naturally compute $(n+1)P$ for free while computing nP. See [**43**] for the analogous genus-1 case.

5. *Weierstrass to Edwards: genus-1 efficiency and simplicity*

Recall from § 4 that we construct a group $J(\mathbf{F}_p)$ having order 16ℓ where ℓ is a large prime. This also forces the group $W(\mathbf{F}_p) = E(\mathbf{F}_{p^2})$ to have order 16ℓ. This in turn forces $E(\mathbf{F}_{p^2})$ to have at least one point of order 4, and thus to be expressible as an Edwards curve. Computer experiments suggest that this procedure actually forces the order-16 subgroup of $E(\mathbf{F}_{p^2})$ to have shape $(\mathbf{Z}/4) \times (\mathbf{Z}/4)$, which we do not consider optimal, but in a moment we will force the shape to be what we actually want.

To simplify elliptic-curve arithmetic we apply further 2-isogenies to obtain a *complete* Edwards curve; the 2-isogenies change the structure of the order-16 subgroups but act as group isomorphisms between the order-ℓ subgroups. Specifically, starting from Legendre form $y^2 = (x - r_0)(x - r_1)(x - r_2)$, we shift x by either r_0 or r_1 or r_2 to obtain $y^2 = x(x - s_1)(x - s_2)$, which is 2-isogenous to $\bar{y}^2 = \bar{x}^3 + 2(s_1 + s_2)\bar{x}^2 + (s_1 - s_2)^2\bar{x}$, which in turn is birationally equivalent to the twisted Edwards curve $4s_1x^2 + y^2 = 1 + 4s_2x^2y^2$, as in [**3**, Theorem 5.1]. The 2-isogeny here is $(x, y) \mapsto (\bar{x}, \bar{y}) = (y^2/x^2, y(s_1s_2 - x^2)/x^2)$, with dual $(\bar{x}, \bar{y}) \mapsto (x, y) = (\bar{y}^2/(4\bar{x}^2), \bar{y}((s_1 - s_2)^2 - \bar{x}^2)/(8\bar{x}^2))$.

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

In the example below we use a chain of two 2-isogenies followed by the birational equivalence. These isogenies replace $(\mathbf{Z}/4) \times (\mathbf{Z}/4)$ first with $(\mathbf{Z}/8) \times (\mathbf{Z}/2)$ and then with $\mathbf{Z}/16$. For background on the underlying 'volcano' structure see, for example, [50].

5.1. *A numerical example, part III*

Consider again the example in § 2.2, with $p = 2^{127} - 309$, $i^2 = -1$, $r = 33 + 56i$, and $s = 159 + 56i$. We convert the elliptic curve $y^2 = rx^3 + sx^2 + s^p x + r^p$ over $\mathbf{F}_{p^2}$ to a complete Edwards curve as follows.

Substitute $y = \bar{y}/r$ and $x = \bar{x}/r$, obtaining the isomorphic curve $\bar{y}^2 = \bar{x}^3 + s\bar{x}^2 + rs^p\bar{x} + r^2r^p$, that is, $\bar{y}^2 = (\bar{x} + (63 + 16i))(\bar{x} + (63 - 16i))(\bar{x} + (33 + 56i))$.

Substitute $\bar{y} = y$ and $\bar{x} = x - 63 - 16i$, obtaining $y^2 = x(x - 32i)(x - 30 + 40i)$. Apply the standard 2-isogeny to $\bar{y}^2 = \bar{x}^3 + (60 - 16i)\bar{x}^2 + (-4284 - 4320i)\bar{x}$, which (for $p = 2^{127} - 309$) factors as $\bar{y}^2 = \bar{x}(\bar{x} - s_1)(\bar{x} - s_2)$ where s_1, s_2 are respectively

$$4653686483403895416558974226954473597 6 + 3053058895835236923491807690724 9897409i,$$

$$1236043186264302775660975614463393693 83 + 1396105945021168624967692268086342 08026i.$$

Substitute $\bar{y} = y$ and $\bar{x} = x + s_1$, obtaining $y^2 = x(x + s_1)(x + s_1 - s_2)$. Apply the standard 2-isogeny to $\bar{y}^2 = \bar{x}^3 + 2(s_2 - 2s_1)\bar{x}^2 + s_2^2\bar{x}$ and the standard birational equivalence to the twisted Edwards curve $-4s_1x^2 + y^2 = 1 + 4(s_2 - s_1)x^2y^2$. This is a complete twisted Edwards curve since $4(s_2 - s_1)$ is not a square while $-4s_1$ is a square, specifically t^2 where t is

$$9670480793874435440724108742532823671 9 + 2326843241701947708279487113477236 8011i.$$

Finally substitute $y = \bar{y}$ and $x = \bar{x}/t$ to obtain the complete Edwards curve $\bar{x}^2 + \bar{y}^2 = 1 + d\bar{x}^2\bar{y}^2$ with $d = 4(s_2 - s_1)/t^2$. We double-checked that the points on this curve form a cyclic group of order 16ℓ: we used the Edwards addition law to compute $16P$ and $8\ell P$ for random points P until we found a generator.

6. *The search for small parameters*

Gaudry and Schost [27] used more than 1 000 000 CPU hours to count points on many genus-2 curves with small parameters; eventually they found a secure twist-secure curve. Specifically, the quantities a^2, b^2, c^2, d^2 in § 4 (and therefore also A^2, B^2, C^2, D^2) are small integers for the Gaudry–Schost surface. The importance of this condition, as mentioned in § 1, is that many of the multiplications in Gaudry's $K(\mathbf{F}_p)$ formulas are multiplications by these parameters. The Gaudry–Schost surface was used for the speed records in [4, 6].

Scholten curves allow much faster point-counting; recall from § 2 that this was Scholten's motivation. However, Scholten curves are quite rare among hyperelliptic curves. The easiest way to see this (as in [23]) is to classify varieties by their number of rational points: the number of points on a uniform random genus-2 Jacobian over $\mathbf{F}_p$ is well distributed over a range of $\Theta(p^{3/2})$ integers, while the number of points on an elliptic curve over $\mathbf{F}_{p^2}$ is limited to a range of $\Theta(p)$ integers.

From these statistics one might guess that asymptotically there do not exist any Scholten curves over $\mathbf{F}_p$ whose parameters a^2, b^2, c^2, d^2 are integers bounded by $(\log p)^{O(1)}$, or even by $p^{o(1)}$. In other words, one might guess that searching through small integer parameters will take a very long time to find a Scholten curve, never mind a secure Scholten curve. Presumably there still exist a^2, b^2, c^2, d^2 much smaller than average, saving time, but the *existence* of a fast cryptosystem is of no use if we cannot *find* the cryptosystem.

However, as the reader can see from our numerical example, these guesses are incorrect. The curve $y^2 = z^6 + (7/3)z^5 - (7/4)z^4 - (14/3)z^3 + (7/4)z^2 + (7/3)z - 1$ is a Scholten curve over

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

$\mathbf{F}_p$ for every large $p \in 3 + 4\mathbf{Z}$ and nevertheless has very small Kummer-surface parameters $(20 : 1 : 20 : 40)$. It is reasonable to conjecture that this example is a secure Scholten curve for a considerable fraction of all p, often also a twist-secure Scholten curve.

To explain what is going on in this example we generalize the concept of Scholten curves to any degree-2 Galois field extension $K \subset L$: that is, any degree-2 field extension $K \subset L$ with an order-2 automorphism $x \mapsto \overline{x}$ of L having fixed field K. Scholten's case is $K = \mathbf{F}_q$, $L = \mathbf{F}_{q^2}$, and $\overline{x} = x^q$, where q is an odd prime power. The point of our generalization is to allow $K = \mathbf{Q}$, for example with $L = \mathbf{Q}[i]/(i^2 + 1)$ and $\overline{i} = -i$.

We define a Scholten curve in this generality as a hyperelliptic curve

$$y^2 = \frac{r(1-\overline{\beta}z)^6 + s(1-\overline{\beta}z)^4(1-\beta z)^2 + \overline{s}(1-\overline{\beta}z)^2(1-\beta z)^4 + \overline{r}(1-\beta z)^6}{r\overline{\beta}^6 + s\overline{\beta}^4\beta^2 + \overline{s}\overline{\beta}^2\beta^4 + \overline{r}\beta^6}$$

assuming that the denominator is nonzero, that $y^2 = rx^3 + sx^2 + \overline{s}x + \overline{r}$ is elliptic (note that this prevents the field characteristic from being 2), and that $r, s, \beta \in L$ with $\beta \notin K$. This hyperelliptic curve is defined over K.

Any such hyperelliptic curve over $K = \mathbf{Q}$ with $L = \mathbf{Q}(\sqrt{\Delta})$ can be reduced to a Scholten curve over $\mathbf{F}_p$ modulo half of all primes p: specifically, almost all primes p for which Δ is not a square in $\mathbf{F}_p$. The only reason we say 'almost' is that a few bad primes p can make the reduction fail, for example by reducing the elliptic curve to a non-elliptic curve.

Our numerical example $\sqrt{\rho_1} = i$, $\sqrt{\rho_2} = (3 + 4i)/5$, $\sqrt{\rho_3} = (5 + 12i)/13$, $r = 33 + 56i$, $s = 159 + 56i$, $\beta = i$ illustrates that there are Scholten curves over $\mathbf{Q}$ whose Jacobians also have Kummer surfaces defined over $\mathbf{Q}$. (As in § 4.4, we write 'Kummer surface' only for the traditional highly symmetric Kummer surfaces, allowing use of Gaudry's efficient formulas from [**25**].) The resulting Kummer-surface parameters a^2, b^2, c^2, d^2 are constants: they do not grow with p. What we are doing here is viewing the entire hyper-and-elliptic picture for $\mathbf{F}_p(\sqrt{\Delta})$ over $\mathbf{F}_p$ as a reduction modulo p of a generic hyper-and-elliptic picture for $\mathbf{Q}(\sqrt{\Delta})$ over $\mathbf{Q}$, with conjugation $\sqrt{\Delta} \mapsto -\sqrt{\Delta}$ as a p-independent view of the pth-power Frobenius map used by Scholten.

We scanned through various other norm-1 elements $\sqrt{\rho_1}, \sqrt{\rho_2}, \sqrt{\rho_3} \in \mathbf{Q}(i)$, together with choices of permutations of the six roots of f, and quickly found many further cases in which $\lambda\mu/\nu$ and $\mu(\mu-1)(\lambda-\nu)/(\nu(\nu-1)(\lambda-\mu))$ are squares in $\mathbf{Q}$. For example,

$$y^2 = (x + 7/4)(x - 4/7)(x + 17/7)(x - 7/17)(x + 37/16)(x - 16/37)$$

is a Scholten curve for $r = 8648575 - 15615600i$, $s = -40209279 - 33245520i$, and $\beta = i$, corresponding to $(a^2 : b^2 : c^2 : d^2) = (6137, 833, 2275, 2275)$. Having many such examples means that one can find secure small-parameter Kummer-compatible Scholten curves for any desired prime $p \in 3 + 4\mathbf{Z}$. (Further characterization of the solution set might allow even faster enumeration of solutions but of course would not save time in point-counting.) Presumably there are also many solutions for other quadratic extensions of $\mathbf{Q}$, although $\mathbf{Q}(i)$ is adequate for, for example, the very convenient prime $p = 2^{127} - 1$.

References

1. K. Alster, J. Urbanowicz and H. C. Williams (eds), *Public-key cryptography and computational number theory, proceedings of the international conference held in Warsaw, September 11–15, 2000* (Walter de Gruyter, 2001) ISBN 3-11-017046-9; MR 2002h:94001, see [23].
2. S. Arita, K. Matsuo, K. Nagao and M. Shimura, 'A Weil descent attack against elliptic curve cryptosystems over quartic extension fields', 2004, https://eprint.iacr.org/2004/240 (citations in this document: § 2, § 2, § 2, § 3.2).
3. D. J. Bernstein, P. Birkner, M. Joye, T. Lange and C. Peters, 'Twisted Edwards curves', in Africacrypt 2008 [53], 2008, 389–405, https://eprint.iacr.org/2008/013 (citations in this document: § 5).

https://doi.org/10.1112/S1461157014000394 Published online by Cambridge University Press

4. D. J. Bernstein, C. Chuengsatiansup, T. Lange and P. Schwabe, 'Kummer strikes back: new DH speed records', 2014, https://eprint.iacr.org/2014/134 (citations in this document: § 1.1, § 1.1, § 1.1, § 4.4, § 6).

5. G. Bertoni and J.-S. Coron (eds), *Cryptographic hardware and embedded systems, CHES 2013, 15th international workshop, Santa Barbara, CA, USA, August 20–23, 2013*, Lecture Notes in Computer Science 8086 (Springer, 2013) ISBN 978-3-642-40348-4, see [7, 44].

6. J. W. Bos, C. Costello, H. Hisil and K. Lauter, 'Fast cryptography in genus 2', in Eurocrypt 2013 [34], 2013, 194–210, https://eprint.iacr.org/2012/670 (citations in this document: § 1.1, § 1.2, § 4.1, § 4.1, § 4.1, § 4.4, § 4.4, § 4.4, § 6).

7. J. W. Bos, C. Costello, H. Hisil and K. Lauter, 'High-performance scalar multiplication using 8-dimensional GLV/GLS decomposition', in CHES 2013 [5], 2013, 331–348, https://eprint.iacr.org/2013/146 (citations in this document: § 1.1).

8. W. Bosma, J. Cannon and C. Playoust, 'The Magma algebra system. I. The user language', *J. Symbolic Comput.* 24 (1997) 235–265, http://www.math.ru.nl/∼bosma/pubs/JSC1997Magma.pdf (citations in this document: § 2.2).

9. L. Brankovic and W. Susilo (eds), *Seventh Australasian information security conference (AISC 2009), Wellington, New Zealand,* Conferences in Research and Practice in Information Technology (CRPIT) 98 (ACS, 2009), see [30].

10. J. W. S. Cassels and E. V. Flynn, *Prolegomena to a middlebrow arithmetic of curves of genus* 2, London Mathematical Society Lecture Note Series 230 (Cambridge University Press, 1996) ISBN 0-521-48370-0 (citations in this document: § 4.4, § 4.4).

11. D. V. Chudnovsky and G. V. Chudnovsky, 'Sequences of numbers generated by addition in formal groups and new primality and factorization tests', *Adv. Appl. Math.* 7 (1986) 385–434; MR 88h:11094 (citations in this document: § 4.4).

12. R. Cosset, 'Factorization with genus 2 curves', *Math. Comput.* 79 (2010) 1191–1208; http://arxiv.org/pdf/0905.2325 (citations in this document: § 4.4, § 4.4).

13. J. P. Degabriele, A. Lehmann, K. G. Paterson, N. P. Smart and M. Strefler, 'On the joint security of encryption and signature in EMV', in CT-RSA 2012 [18], 2012, 116–135, http://www.isg.rhul.ac.uk/∼psai074/publications/EMV_Joint_Sec.pdf (citations in this document: § 1.3).

14. C. Diem, 'The GHS attack in odd characteristic', *J. Ramanujan Math. Soc.* 18 (2003) 1–32; (citations in this document: § 2, § 3.2).

15. C. Diem and J. Scholten, 'Cover attacks: a report for the AREHCC project', 2003, http://www.math.uni-leipzig.de/∼diem/preprints/cover-attacks.pdf (citations in this document: § 2).

16. C. Diem and J. Scholten, 'An attack on a trace-zero cryptosystem', 2004, http://www.math.uni-leipzig.de/∼diem/preprints/trace-zero.pdf (citations in this document: § 2).

17. W. Diffie and M. Hellman, 'New directions in cryptography', *IEEE Trans. Inform. Theory* 22 (1976) 644–654; ISSN 0018-9448; MR 55:10141 (citations in this document: § 1).

18. O. Dunkelman (ed.), *Topics in cryptology, CT-RSA 2012, the cryptographers' track at the RSA Conference 2012, San Francisco, CA, USA, February 27–March 2, 2012*, Lecture Notes in Computer Science 7178 (Springer, 2012) ISBN 978-3-642-27953-9, see [13].

19. A. Faz-Hernandez, P. Longa and A. H. Sanchez, 'Efficient and secure algorithms for GLV-based scalar multiplication and their implementation on GLV–GLS curves', 2013, https://eprint.iacr.org/2013/158 (citations in this document: § 1.1).

20. E. V. Flynn, 'Genus 2 site', 2007, http://people.maths.ox.ac.uk/flynn/genus2 (citations in this document: § 4.4, § 4.4).

21. D. M. Freeman and T. Satoh, 'Constructing pairing-friendly hyperelliptic curves using Weil restriction', *J. Number Theory* 131 (2011) 959–983, http://eprint.iacr.org/2009/103 (citations in this document: § 2).

22. G. Frey, 'How to disguise an elliptic curve (Weil descent)', Presentation at ECC 1998, 1998, http://www.cacr.math.uwaterloo.ca/conferences/1998/ecc98/slides.html (citations in this document: § 2).

23. S. D. Galbraith, 'Limitations of constructive Weil descent', in [1], 2001, 59–70; MR 2002m:11052 (citations in this document: § 2, § 6).

24. P. Gaudry, 'Variants of the Montgomery form based on theta functions', 2006, see also newer version [25], http://www.loria.fr/∼gaudry/publis/toronto.pdf (citations in this document: § 1.1).

25. P. Gaudry, 'Fast genus 2 arithmetic based on theta functions', *J. Math. Cryptol.* 1 (2007) 243–265; see also older version [24], http://webloria.loria.fr/∼gaudry/publis/arithKsurf.pdf (citations in this document: § 4.1, § 4.1, § 4.4, § 4.4, § 6).

26. P. Gaudry, F. Hess and N. P. Smart, 'Constructive and destructive facets of Weil descent on elliptic curves', *J. Crypt.* 15 (2002) 19–46 (citations in this document: § 2, § 2, § 2, § 3.2).

27. P. Gaudry and É. Schost, 'Genus 2 point counting over prime fields', *J. Symbolic Comput.* 47 (2012) 368–400, http://www.csd.uwo.ca/∼eschost/publications/countg2.pdf (citations in this document: § 1.1, § 6).

28. D. Giry, 'Cryptographic key length recommendation', 2014, http://keylength.com (citations in this document: § 1.1).

29. S. Haber and B. Pinkas, 'Securely combining public key cryptosystems', in CCS 2001 [46] 2001, 215–224 (citations in this document: § 1.3).

30. H. Hisil, K. K.-H. Wong, G. Carter and E. Dawson, 'Faster group operations on elliptic curves', in AISC 2009 [9], 2009, 7–19, https://eprint.iacr.org/2007/441 (citations in this document: § 3.2).

31. J. Hoffman-Andrews, 'Forward secrecy at Twitter', 2013, https://blog.twitter.com/2013/forward-secrecy-at-twitter-0 (citations in this document: § 1).

32. E. W. Howe and K. S. Kedlaya (eds), *Tenth algorithmic number theory symposium* (Mathematical Sciences Publishers, 2013) ISBN 978-1-935107-01-9, see [50].

33. T. Iijima, F. Momose and J. Chao, 'Classification of elliptic/hyperelliptic curves with weak coverings against GHS attack under an isogeny condition', 2013, http://eprint.iacr.org/2013/487 (citations in this document: § 2).

34. T. Johansson and P. Q. Nguyen (eds), *Advances in cryptology — EUROCRYPT 2013, 32nd annual international conference on the theory and applications of cryptographic techniques, Athens, Greece, May 26–30, 2013*, Lecture Notes in Computer Science 7881 (Springer, 2013) ISBN 978-3-642-38347-2, see [6].

35. A. Joux (ed.), *Advances in cryptology — EUROCRYPT 2009, 28th annual international conference on the theory and applications of cryptographic techniques, Cologne, Germany, April 26–30, 2009*, Lecture Notes in Computer Science 5479 (Springer, 2009), see [47].

36. N. Koblitz, 'Elliptic curve cryptosystems', *Math. Comput.* 48 (1987) 203–209; ISSN 0025-5718; MR 88b:94017 (citations in this document: § 1.1).

37. Ç. K. Koç, D. Naccache and C. Paar (eds), *Cryptographic hardware and embedded systems — CHES 2001, third international workshop, Paris, France, May 14–16, 2001*, Lecture Notes in Computer Science 2162 (Springer, 2001) ISBN 3-540-42521-7; MR 2003g:94002, see [43].

38. A. Langley, 'Protecting data for the long term with forward secrecy', 2011, http://googleonlinesecurity.blogspot.nl/2011/11/protecting-data-for-long-term-with.html (citations in this document: § 1).

39. D. H. Lee and X. Wang (eds), *Advances in cryptology — ASIACRYPT 2011, 17th international conference on the theory and application of cryptology and information security, Seoul, South Korea, December 4–8, 2011*, Lecture Notes in Computer Science 7073 (Springer, 2011) ISBN 978-3-642-25384-3, see [45].

40. V. S. Miller, 'Use of elliptic curves in cryptography', in [54], 1986, 417–426; MR 88b:68040 (citations in this document: § 1.1).

41. F. Momose and J. Chao, 'Scholten forms and elliptic/hyperelliptic curves with weak Weil restrictions', 2005, http://eprint.iacr.org/2005/ (citations in this document: § 2).

42. M. B. Monagan and R. Pearce, 'Rational simplification modulo a polynomial ideal', in ISSAC 2006 [52], 2006, 239–245 (citations in this document: § 3.2).

43. K. Okeya and K. Sakurai, 'Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve', in CHES 2001 [37], 2001, 126–141 (citations in this document: § 4.6).

44. T. Oliveira, J. López, D. F. Aranha and F. Rodríguez-Henríquez, 'Lambda coordinates for binary elliptic curves', in CHES 2013 [5], 2013, 311–330, https://eprint.iacr.org/2013/131 (citations in this document: § 1.1).

45. K. G. Paterson, J. C. N. Schuldt, M. Stam and S. Thomson, 'On the joint security of encryption and signature, revisited', in Asiacrypt 2011 [39] 2011, 161–178, https://eprint.iacr.org/2011/486 (citations in this document: § 1.3).

46. M. K. Reiter and P. Samarati (eds), CCS 2001, *proceedings of the 8th ACM conference on computer and communications security, Philadelphia, Pennsylvania, USA, November 6–8, 2001* (Association for Computing Machinery, 2001) ISBN 1-58113-385-5, see [29].

47. T. Satoh, 'Generating genus two hyperelliptic curves over large characteristic finite fields', in EUROCRYPT 2009 [35], 2009, 536–553 (citations in this document: § 2).

48. J. Scholten, 'Weil restriction of an elliptic curve over a quadratic extension', 2003, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.7987&rep=rep1&type=pdf (citations in this document: § 2, § 2, § 2, § 2, § 2.1, § 3.1).

49. W. Stein (ed.), *Sage Mathematics Software* (*Version 6.1.1*) (The Sage Group, 2014) http://www.sagemath.org (citations in this document: § 2.2).

50. A. V. Sutherland, 'Isogeny volcanoes', in ANTS X [32], 2013, http://arxiv.org/abs/1208.5370 (citations in this document: § 5).

51. N. Thériault, 'Weil descent attack for Kummer extensions', *J. Ramanujan Math. Soc.* 18 (2003) 281–312; (citations in this document: § 2).

52. B. M. Trager (ed.), *ISSAC '06: proceedings of the 2006 international symposium on symbolic and algebraic computation, Genoa, Italy, July 09–12, 2006* (ACM, 2006) ISBN 1-59593-276-3, see [42].

53. S. Vaudenay (ed.), *Progress in cryptology — AFRICACRYPT 2008, first international conference on cryptology in Africa, Casablanca, Morocco, June 11–14, 2008*, Lecture Notes in Computer Science 5023 (Springer, 2008) ISBN 978-3-540-68159-5, see [3].

54. H. C. Williams (ed.), *Advances in cryptology: CRYPTO '85*, Lecture Notes in Computer Science 218 (Springer, 1986) ISBN 3-540-16463-4, see [40].

Daniel J. Bernstein
Computer Science
University of Illinois at Chicago
Chicago, IL 60607-7045
USA

and

Mathematics and Computer Science
Technische Universiteit Eindhoven
PO Box 513, 5600 MB Eindhoven
The Netherlands

djb@cr.yp.to

Tanja Lange
Mathematics and Computer Science
Technische Universiteit Eindhoven
PO Box 513, 5600 MB Eindhoven
The Netherlands

tanja@hyperelliptic.org