Efficient $(3,3)$-isogenies on fast Kummer surfaces

Maria Corte-Real Santos, Craig Costello, Benjamin Smith

EFFICIENT $(3,3)$-ISOGENIES ON FAST KUMMER SURFACES

MARIA CORTE-REAL SANTOS, CRAIG COSTELLO, AND BENJAMIN SMITH

Abstract. We give an alternative derivation of (N, N)-isogenies between fast Kummer surfaces which complements existing works based on the theory of theta functions. We use this framework to produce explicit formulæ for the case of $N = 3$, and show that the resulting algorithms are more efficient than all prior $(3, 3)$-isogeny algorithms.

1. Introduction

Isogenies of elliptic curves are well-understood, at least from an algorithmic point of view, in theory and in practice. Given the Weierstrass equation of an elliptic curve E, and a generator P of a finite subgroup of E, Vélu's formulæ [52] allow us to write down polynomials defining a normalized quotient isogeny $\Phi : E \to E/\langle P \rangle$ (with variants for alternative curve models [38], or for rational subgroups with irrational generators [34]). Building on these formulæ, there also exist highly efficient algorithms for evaluating an isogeny at points of E *without* deriving a polynomial representation for the isogeny itself (including [2], [19], and [43], for example). Interest in these formulæ and algorithms has recently intensified with the development of *isogeny-based cryptography* as a source of cryptosystems conjectured to be resistant against quantum attacks.

As a generalization of an elliptic curve, we consider principally polarized abelian varieties, and the first non-elliptic examples are Jacobians of genus-2 curves. Genus-2 curves are hyperelliptic curves with affine plane models

$$C : y^2 = f(x) \quad \text{where } f(x) \text{ is squarefree of degree 5 or 6}\,,$$

and the Jacobian $\mathcal{J} = \mathrm{Jac}(C)$ of C is a 2-dimensional principally polarized abelian variety (p.p.a.v.), birational to $C^{(2)}$, parameterizing the degree-0 Picard group of the curve C.

Mumford [39], Cantor [10], Grant [31], and Flynn [26, 28] laid the ground for explicit geometric and number-theoretic computations with genus-2 Jacobians, and Cassels and Flynn's text [11] presented a first unified view of the arithmetic of genus-2 Jacobians. Later, Gaudry [30] proposed Kummer surfaces of genus-2 Jacobians as a setting for efficient discrete-logarithm-based cryptosystems, building on a variant [15] of Lentra's ECM factoring algorithm [33]. The Kummer surface $\mathcal{K}$ of a Jacobian $\mathcal{J}$ is the image of the quotient morphism $\pi : \mathcal{J} \to \mathcal{K} = \mathcal{J}/\{\pm 1\}$; as such, it is the genus-2 analogue of the x-coordinate of elliptic curves. Geometrically, Kummer surfaces have convenient models as singular quartic surfaces in $\mathbb{P}^3$.

Cosset put Chudnovsky and Chudnovsky's Kummer ECM into practice in [17], while high-speed, high-security Kummer-based implementations of Diffie–Hellman key exchange [1, 6, 44] and signature schemes [44, 45] can give significant practical improvements over elliptic curves in many contexts.

However, while the basic arithmetic of genus-2 Jacobians and Kummer surfaces has matured, and while cryptographic applications has driven great improvements in the efficiency of the resulting formulæ and algorithms, the corresponding explicit theory of isogenies lags behind. First, note that just as elliptic isogenies factor naturally into compositions of scalar multiplications and isogenies with prime cyclic kernel (i.e., isomorphic to $\mathbb{Z}/N\mathbb{Z}$ with N prime), isogenies of abelian surfaces (including Jacobians of genus-2 curves) decompose into compositions of scalar multiplications and (N, N)-*isogenies* (with kernel isomorphic to $(\mathbb{Z}/N\mathbb{Z})^2$).[1] The fundamental task, then, is to compute and evaluate (N, N)-isogenies where N is prime. We can do this on the level of the Jacobian (using e.g. correspondences on genus-2 curves [50]), or we can use the fact that isogenies commute with -1 to move down to the more tractable Kummer surfaces. Indeed, as Cassels and Flynn note, "we lose nothing by going down to the Kummers, because [the map] lifts automatically to a map of abelian varieties." [11, §9.3].

The case $N = 2$ is classical: explicit methods and formulæ go back to Richelot [46], and were re-developed in modern terms by Bost and Mestre [8] and Cassels and Flynn [11, §3]. Going further, we find some first efforts at explicit curve-based formulæ for the case $N = 3$ by Smith [49] (building on an ineffective general method due to Dolgachev and Lehavi [24]), and more general results due to Couveignes and Ezome [21]. Moving to general Kummer surfaces, Bruin, Flynn and Testa [9], Nicholls [40], and Flynn [27] gave more powerful formulæ for $N = 3$, 4, and 5, respectively, in a number-theoretic context; Nicholls even gives a method for general N. Flynn and Ti revisited the formulæ for $N = 3$ in a cryptographic context [29], and Decru and Kunzweiler [23] further optimized these formulæ, drastically improving their efficiency. However, none of these formulæ make use of the special symmetries of the most efficient Kummer surfaces that have been used in cryptographic implementations.

Bisson, Cosset, Lubicz, and Robert have advanced an ambitious program [4, 16, 36, 47, 48] based on the theory of theta functions [39] to provide asymptotically efficient algorithms for arbitrary odd N (and beyond genus 2 to arbitrarily high dimension). The `AVIsogenies` software package based on their results is publicly available [5]. These algorithms are certainly compatible with fast Kummer surfaces, but they target isogeny evaluation for general abelian varieties, rather than the construction of compact explicit formulæ in genus 2 that can be studied, analysed, and optimized in their own right. Nevertheless, these techniques were recently revisited by Dartois, Maino, Pope and Robert [22] in the context of cryptography to efficiently compute chains of $(2, 2)$-isogenies between products of elliptic curves in the theta model.

1.1. **Contributions.** In this article, we give a general method for deriving explicit formulæ for isogenies of fast Kummer surfaces, optimizing the approach of Bruin, Flynn, and Testa by exploiting the high symmetry of these "fast" surfaces, which are the most relevant for applications over finite fields. Our methods are elementary in the sense that they avoid explicitly using the heavy machinery of theta functions required in [18, 22, 37] (though of course theta functions implicitly play a fundamental role in our techniques). We apply these methods to give explicit examples

[1] In some special cases, depending on the endomorphism ring of the Jacobian, we can also have isogenies with cyclic kernel: these isogenies are beyond the scope of this article.

for $N = 3$ and 5. For example, for $N = 3$ we obtain a map $\phi : \mathcal{K} \to \mathcal{K}'$ defined by

$$\phi((X_1 : X_2 : X_3 : X_4)) = (\phi_1(X_1, X_2, X_3, X_4) : \cdots : \phi_4(X_1, X_2, X_3, X_4)) \ ,$$

where

$$\begin{aligned}
\phi_1(X_1, X_2, X_3, X_4) &= X_1 \left(c_1 X_1^2 + c_2 X_2^2 + c_3 X_3^2 + c_4 X_4^2\right) + c_5 X_2 X_3 X_4 \, , \\
\phi_2(X_1, X_2, X_3, X_4) &= X_2 \left(c_2 X_1^2 + c_1 X_2^2 + c_4 X_3^2 + c_3 X_4^2\right) + c_5 X_1 X_3 X_4 \, , \\
\phi_3(X_1, X_2, X_3, X_4) &= X_3 \left(c_3 X_1^2 + c_4 X_2^2 + c_1 X_3^2 + c_2 X_4^2\right) + c_5 X_1 X_2 X_4 \, , \\
\phi_4(X_1, X_2, X_3, X_4) &= X_4 \left(c_4 X_1^2 + c_3 X_2^2 + c_2 X_3^2 + c_1 X_4^2\right) + c_5 X_1 X_2 X_3 \, ,
\end{aligned}$$

and c_i are rational functions in the theta-null constants a, b, c, d defining $\mathcal{K}$ and the coordinates of the generators of the kernel (see Section 4.1 for the explicit expresssions). This map can be evaluated with at most 88 multiplications and 12 squarings in the field containing the theta constants and generator coordinates.

To illustrate the potential benefits of our formulæ in practical applications, we give experimental results on cryptographic hash functions based on chains of $(3, 3)$-isogenies, as in [12] and [23]. In Section 5 we present 3DAC: a three-dimensional *differential addition chain*, and use it to construct (N^k, N^k)-isogeny kernels correctly, efficiently, and securely. Combined with our $(3, 3)$-isogeny formulæ, this allows us to efficiently compute $(3^k, 3^k)$-isogenies on fast Kummer surfaces. The hash function we define in Section 6 uses these isogenies, exploiting the efficient arithmetic of fast Kummer surfaces for the first time, to gain speed-ups of between 1.4× and 2.3× over the Castryk–Decru hash function [12] and between 27.0× and 28.3× over the Decru–Kunzweiler hash function [23].

1.2. **Software.** The source code accompanying this paper is written in `MAGMA` [7], Python and SageMath [51] and is publicly available under the MIT license. It is available at

https://github.com/mariascrs/KummerIsogenies.

Acknowledgements. We thank Chris Nicholls for making the code accompanying his thesis available to us, which we used to obtain the formulae for $(2, 2)$-isogenies presented in Section 3. We also thank Sam Frengley for many helpful conversations during the preparation of this paper. The first author was supported by UK EPSRC grant EP/S022503/1. This work received funding from the France 2030 program, managed by the French National Research Agency under grant agreement No. ANR-22-PETQ-0008 PQ-TLS.

2. Fast Kummer surfaces and their arithmetic

Let $\Bbbk$ be a perfect field—typically, a finite field or a number field—of characteristic not 2, and fix an algebraic closure $\overline{\Bbbk}$. If $\Bbbk = \mathbb{F}_q$, then we measure the time complexity of our algorithms in terms of elementary operations in $\mathbb{F}_q$. We let `M`, `S`, and `a` denote the cost of a single multiplication, squaring, and addition (or subtraction) in $\mathbb{F}_q$, respectively.

2.1. **Genus-2 curves and their Jacobians.** Every smooth genus-2 curve over $\Bbbk$ is isomorphic to a curve of the form $C : y^2 = f(x)$, where $f(x) \in \Bbbk[x]$ is a squarefree polynomial of degree 6.

For our applications we may suppose that $f(x)$ has all its roots in $\Bbbk$, so (after, for example, mapping one root to 0, one to 1, and one to ∞, as in [41, §2.1]) we can assume C has a *Rosenhain model*

$$C \cong C_{\lambda,\mu,\nu}/\Bbbk : y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu) \quad \text{with } \lambda, \mu, \nu \in \Bbbk\,;$$

the values λ, μ, and ν are called *Rosenhain invariants* of $C_{\lambda,\mu,\nu}$.

Fix a prime N not divisible by $\mathrm{char}(\Bbbk)$. The N-torsion subgroup $\mathcal{J}[N]$ of $\mathcal{J}$ is a $\mathbb{Z}/N\mathbb{Z}$-module of rank 4, that is, $\mathcal{J}[N] \cong (\mathbb{Z}/N\mathbb{Z})^4$; and the (canonical) principal polarisation on $\mathcal{J}$ induces a non-degenerate, bilinear, and antisymmetric *N-Weil pairing*

$$e_N : \mathcal{J}[N] \times \mathcal{J}[N] \to \mu_N\,.$$

A subgroup $G \subseteq \mathcal{J}[N]$ is *isotropic* if $e_N(P,Q) = 1$ for all $P, Q \in G$, and *maximal isotropic* if it is not properly contained in any isotropic subgroup of $\mathcal{J}[N]$. Since N is prime, if G is maximal isotropic then $G \cong (\mathbb{Z}/N\mathbb{Z})^2$; we say G is an *(N,N)-subgroup*. If G is a maximal isotropic subgroup of $\mathcal{J}[N]$, then the quotient isogeny of abelian varieties

$$\Phi : \mathcal{J} \to A' := \mathcal{J}/G$$

is an isogeny of p.p.a.v.s: that is, there is a principal polarization on A' that pulls back via Φ to N times the principal polarization on $\mathcal{J}$. We say that Φ is an *(N,N)-isogeny*.

Being a principally polarized abelian surface, A' is (as a p.p.a.v.) the Jacobian of a genus 2 curve, say $\mathcal{J}'$, or a product of elliptic curves $E_1' \times E_2'$ (equipped with the product polarisation). The case $A' = \mathcal{J}$ is the general case, and the primary focus of this paper.

2.2. **Isogenies and Kummer surfaces.** The Kummer surface $\mathcal{K}$ of a Jacobian $\mathcal{J}$ is the image of the quotient map $\pi : \mathcal{J} \to \mathcal{K} = \mathcal{J}/\{\pm 1\}$. Geometrically, it has a quartic model in $\mathbb{P}^3$ with sixteen point singularities, called *nodes*; the nodes are the images in $\mathcal{K}$ of the 2-torsion points of $\mathcal{J}$, since these are precisely the points fixed by -1.

Any (N,N)-isogeny $\Phi : \mathcal{J} \to \mathcal{J}'$ descends to a morphism of Kummer surfaces $\phi : \mathcal{K} \to \mathcal{K}'$, such that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{J} & \xrightarrow{\Phi} & \mathcal{J}' \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi'} \\ \mathcal{K} & \xrightarrow{\phi} & \mathcal{K}' \end{array}$$

Abusing terminology, we say a morphism ϕ of Kummer surfaces is an (N,N)-isogeny if it is induced by an (N,N)-isogeny Φ between the corresponding Jacobians.

2.3. **Fast Kummer surfaces.** Following Gaudry, fast Kummer surfaces are defined by four *fundamental theta constants*, which can be computed from the Rosenhain invariants of a genus-2 curve $C/\Bbbk$. Given a hyperelliptic curve $C/\Bbbk$ with Rosenhain invariants $\lambda, \mu, \nu \in \Bbbk$, we define the *fundamental theta constants* $a, b, c, d \in \overline{\Bbbk}$ and *dual theta constants* as $A, B, C, D \in \overline{\Bbbk}$ such that

$$A^2 = a^2 + b^2 + c^2 + d^2, \quad B^2 = a^2 + b^2 - c^2 - d^2,$$
$$C^2 = a^2 - b^2 + c^2 - d^2, \quad D^2 = a^2 - b^2 - c^2 + d^2.$$

The theta constants are related to Rosenhain invariants through the relations

$$\lambda = \frac{a^2c^2}{b^2d^2}\,, \qquad \mu = \frac{c^2e^2}{d^2f^2}\,, \qquad \nu = \frac{a^2e^2}{b^2f^2}\,,$$

where $e, f \in \overline{\Bbbk}$ satisfy $e^2/f^2 = (AB+CD)/(AB-CD)$.

We define the *fast* Kummer model $\mathcal{K}$ corresponding to $\mathcal{C}$ as

$$\begin{aligned}\mathcal{K} : X_1^4 + X_2^4 + X_3^4 + X_4^4 - 2E \cdot X_1X_2X_3X_4 - F \cdot (X_1^2X_4^2 + X_2^2X_3^2) \\ - G \cdot (X_1^2X_3^2 + X_2^2X_4^2) - H \cdot (X_1^2X_2^2 + X_3^2X_4^2) = 0,\end{aligned} \tag{1}$$

where X_1, X_2, X_3, X_4 are coordinates on $\mathbb{P}^3$ and the coefficients E, F, G, H are rational functions in a, b, c, d, namely

$$\begin{aligned} E &:= 256abcdA^2B^2C^2D^2/(a^2d^2 - b^2c^2)(a^2c^2 - b^2d^2)(a^2b^2 - c^2d^2), \\ F &:= (a^4 - b^4 - c^4 + d^4)/(a^2d^2 - b^2c^2), \\ G &:= (a^4 - b^4 + c^4 - d^4)/(a^2c^2 - b^2d^2), \\ H &:= (a^4 + b^4 - c^4 - d^4)/(a^2b^2 - c^2d^2). \end{aligned} \tag{2}$$

This model of $\mathcal{K}$ is often referred to as the *canonical* parameterisation. Note that A^2, B^2, C^2, and D^2 are linear combinations of a^2, b^2, c^2, and d^2, so the equation of $\mathcal{K}$ is determined entirely by a, b, c, d; in fact, $\mathcal{K}$ is determined by the projective point $(a : b : c : d) \in \mathbb{P}^3$.

2.4. **Nodes of the Kummer surface.** The *nodes* of $\mathcal{K}$ are the sixteen points

$$\begin{array}{llll} T_0 = (a:b:c:d), & T_1 = (a:b:-c:-d), & T_2 = (a:-b:c:-d), & T_3 = (a:-b:-c:d), \\ T_4 = (b:a:d:c), & T_5 = (b:a:-d:-c), & T_6 = (b:-a:d:-c), & T_7 = (b:-a:-d:c), \\ T_8 = (c:d:a:b), & T_9 = (c:d:-a:-b), & T_{10} = (c:-d:a:-b), & T_{11} = (c:-d:-a:b), \\ T_{12} = (d:c:b:a), & T_{13} = (d:c:-b:-a), & T_{14} = (d:-c:b:-a), & T_{15} = (d:-c:-b:a). \end{array}$$

Each of the T_i is the image in $\mathcal{K}$ of a two-torsion point $\widetilde{T}_i$ in $\mathcal{J}[2]$. Since $\tilde{T}_i = -\widetilde{T}_i$, the translation-by-$\widetilde{T}_i$ map on $\mathcal{J}$ induces a morphism $\sigma_i : \mathcal{K} \to \mathcal{K}$. In fact, σ_i lifts to a linear map on $\mathbb{A}^4$: that is, it acts like a matrix on the coordinates (X_1, X_2, X_3, X_4) on $\mathbb{P}^3$. Further, σ_i and σ_j commute resp. anticommute if $e_2(\widetilde{T}_i, \widetilde{T}_j) = 1$ resp. -1.

In particular, if we define

$$U_1 := \operatorname{diag}(1, 1, -1, -1), \qquad\qquad U_2 := \operatorname{diag}(1, -1, 1, -1),$$

and

$$V_1 := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad\qquad V_2 := \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Then

$$U_1^2 = U_2^2 = U_1U_2 = U_2U_1 = I_4\,, \qquad V_1^2 = V_2^2 = V_1V_2 = V_2V_1 = I_4\,, \tag{3}$$

and

$$U_1V_2 = V_2U_1\,, \qquad U_2V_1 = V_1U_2\,, \qquad U_1V_1 = -V_1U_1\,, \qquad U_2V_2 = -V_2U_2\,. \tag{4}$$

Taking the labelling of the nodes above, with $T_0 = (a : b : c : d)$ as the image of $\widetilde{T}_0 = 0_{\mathcal{J}}$, the corresponding translations are such that

$$T_i = \sigma_i((a : b : c : d)) \qquad \text{for } 0 \le i \le 15\,;$$

that is,

$$\begin{aligned}
\sigma_0 &= I_4 & \sigma_1 &= U_1 & \sigma_2 &= U_2 & \sigma_3 &= U_1U_2 \\
\sigma_4 &= V_2 & \sigma_5 &= V_2U_1 & \sigma_6 &= V_2U_2 & \sigma_7 &= V_2U_1U_2 \\
\sigma_8 &= V_2V_1 & \sigma_9 &= V_2V_1U_1 & \sigma_{10} &= V_2V_1U_2 & \sigma_{11} &= V_2V_1U_1U_2 \\
\sigma_{12} &= V_1 & \sigma_{13} &= V_1U_1 & \sigma_{14} &= V_1U_2 & \sigma_{15} &= V_1U_1U_2
\end{aligned}$$

Now Eqs. (3) and (4) show that $(\widetilde{T}_1, \widetilde{T}_2, \widetilde{T}_{12}, \widetilde{T}_4)$ is a 2-Weil symplectic basis of $\mathcal{J}[2]$.

2.5. **Operations on the Kummer surface.** Let $\pi : \mathcal{J} \to \mathcal{K}$ be the quotient by -1. The doubling map $[2] : \mathcal{J} \to \mathcal{J}$ commutes with $[-1]$, and hence induces a *pseudo-doubling* map on $\mathcal{K}$, mapping $\pi(P) \mapsto \pi([2]P)$. More generally, for each integer m, the multiplication-by-m map $[m]$ on $\mathcal{J}$ induces a *pseudo-multiplication* $\pi(P) \mapsto \pi([m]P)$ on $\mathcal{K}$.

We can express pseudo-doubling on $\mathcal{K}$ as a composition of four basic building blocks, each a morphism from $\mathbb{P}^3$ to $\mathbb{P}^3$:

(1) the *Hadamard involution* $\mathcal{H} : \mathbb{P}^3 \to \mathbb{P}^3$, which is induced by the linear map on $\mathbb{A}^4$ defined by the matrix

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix};$$

(2) the *squaring* map

$$\mathcal{S} : (X_1 : X_2 : X_3 : X_4) \longmapsto (X_1^2 : X_2^2 : X_3^2 : X_4^2);$$

(3) the *scaling* maps

$$\mathcal{C}_{(\alpha:\beta:\gamma:\delta)} : (X_1 : X_2 : X_3 : X_4) \longmapsto (\alpha X_1 : \beta X_2 : \gamma X_3 : \delta X_4)$$

for each $(\alpha : \beta : \gamma : \delta) \in \mathbb{P}^3(\Bbbk)$; and

(4) the *inversion* map

$$\begin{aligned}
\mathcal{I} : (X_1 : X_2 : X_3 : X_4) &\longmapsto (X_2X_3X_4 : X_1X_3X_4 : X_1X_2X_4 : X_1X_2X_3) \\
&= (1/X_1 : 1/X_2 : 1/X_3 : 1/X_4).
\end{aligned}$$

We can readily see that $\mathcal{H}$ costs 8 $\Bbbk$-additions, $\mathcal{S}$ costs 4 $\Bbbk$-squarings, $\mathcal{C}$ costs 4 $\Bbbk$-multiplications, and $\mathcal{I}$ costs 6 $\Bbbk$-multiplications. Now, if $\mathcal{K}$ is a Kummer surface with fundamental theta constants $(a : b : c : d)$, then pseudo-doubling is given by

$$[2]_* = \mathcal{C}_{\mathcal{I}(\mathcal{O}_\mathcal{K})} \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{C}_{\mathcal{I}((A:B:C:D))} \circ \mathcal{H} \circ \mathcal{S}.$$

While $\mathcal{K}$ inherits scalar multiplication from $\mathcal{J}$, it loses the group law: $\pi(P) = \pm P$ and $\pi(Q) = \pm Q$ in $\mathcal{K}$ do not uniquely determine $\pi(P + Q) = \pm(P + Q)$ (unless at least one of P and Q is in $\mathcal{J}[2]$). However, the operation $\{\pi(P), \pi(Q)\} \mapsto \{\pi(P + Q), \pi(P - Q)\}$ is well-defined, so we have a *pseudo-addition* operation $(\pi(P), \pi(Q), \pi(P - Q)) \mapsto \pi(P + Q)$.

Let $R = (r_1 : r_2 : r_3 : r_4)$ and $S = (s_1 : s_2 : s_3 : s_4)$ be points on $\mathcal{K}$, and let $T^+ = (t_1^+ : t_2^+ : t_3^+ : t_4^+)$ and $T^- = (t_1^- : t_2^- : t_3^- : t_4^-)$ denote the sum $R + S$ and difference $R - S$, respectively. There exist biquadratic forms B_{ij} [11, Theorem 3.9.1] for $\mathcal{K}$ such that for $1 \le i, j \le 4$ we have

$$t_i^+ t_j^- + t_i^- t_j^+ = \lambda B_{ij}\big(r_1, r_2, r_3, r_4; s_1, s_2, s_3, s_4\big) = \lambda B_{ij}(R; S),$$

where $\lambda \in \overline{\Bbbk}$ is a common projective factor depending only on the affine representations chosen for R, S, T^+, T^-. The biquadratic forms $B_{i,j}$ for fast Kummer surfaces are given explicitly by Renes and Smith [45, §5.2].

These biquadratic forms are the basis of explicit pseudo-addition and doubling laws on $\mathcal{K}$. For example, if the difference T^- is known, then the B_{ij} can be used to compute the coordinates of T^+. As we will see in Section 3, the B_{ij} will also be crucial in determining equations for our (N, N)-isogenies.

3. (N, N)-isogenies on fast Kummer surfaces

Throughout this section, $\Phi : \mathcal{J} = \mathrm{Jac}(C) \to \mathcal{J}'$ is an (N, N)-isogeny with kernel $G \subset \mathcal{J}[N]$ (a maximal N-Weil isotropic subgroup of $\mathcal{J}[N]$), where N is a prime number not equal to the characteristic of the base field $\Bbbk$. Our goal is to compute an explicit and efficiently-computable collection of polynomials defining the induced map $\phi : \mathcal{K} \to \mathcal{K}'$ when $\mathcal{K}$ and $\mathcal{K}'$ admit *fast* models.

3.1. A warm-up with $N = 2$.

We first dispose of the case $N = 2$. Let $\widetilde{T}_0, \ldots, \widetilde{T}_{15}$ be the 16 points in $\mathcal{J}[2]$, and let $T_0, \ldots, T_{15}$ be their images in $\mathcal{K}$. Recall that $T_i = \sigma_i(T_0)$, where $\sigma_i : \mathcal{K} \to \mathcal{K}$ is a morphism defining the translation-by-T_i map. There are precisely fifteen (images of) $(2, 2)$-subgroups in $\mathcal{K}$, and they are the images of

$$\widetilde{G}_{ij} := \{\widetilde{T}_0,\ \widetilde{T}_i,\ \widetilde{T}_j,\ \widetilde{T}_i + \widetilde{T}_j\} \subset \mathcal{J}[2]$$

in $\mathcal{K}$, where $1 \leq i \neq j \leq 15$ such that the linear maps corresponding to σ_i and σ_j commute. This gives 15 corresponding $(2, 2)$-isogenies given by $\phi : \mathcal{K} \to \mathcal{K}' = \mathcal{K}/G_{i,j}$. For each unique $(2, 2)$-subgroup, we can associate a morphism $\alpha : \mathcal{K} \to \mathcal{K}$ induced by a linear map on $\mathbb{A}^4$ defined by a matrix $\mathbf{A} = (a_{i,j})_{1 \leq i,j \leq 4}$ with $a_{i,j}^2 \in \{0, 1\}$, such that the corresponding $(2, 2)$-isogeny $\mathcal{K} \to \widetilde{\mathcal{K}}$ is given by

$$\psi(P) := \mathcal{H} \circ \mathcal{S} \circ \alpha(P).$$

We fully specify $\mathbf{A}$ for each $(2, 2)$-subgroup in Appendix A.

Note however, that $\widetilde{\mathcal{K}}$ will not be in the correct form, as given by Equation (1). We must therefore apply a final scaling $\mathcal{C}_U$ where $U = \mathcal{S}^{-1} \circ \mathcal{I} \circ \psi(O_{\mathcal{K}})$. From this we obtain a $(2, 2)$-isogeny $\phi := \mathcal{C}_U \circ \psi : \mathcal{K} \to \mathcal{K}' = \mathcal{K}/G_{i,j}$, where $\mathcal{K}$ and $\mathcal{K}'$ are fast Kummer surfaces.

Example 3.1. Consider the $(2, 2)$-subgroup

$$G_{1,2} = \{(a\colon b\colon c\colon d), (a\colon b\colon -c\colon -d), (a\colon -b\colon c\colon -d), (a\colon -b\colon c\colon -d)\}.$$

Here $\mathbf{A} = I_4$ and the $(2, 2)$-isogeny is given by

$$(X_1\colon X_2\colon X_3\colon X_4) \mapsto$$
$$\left(\frac{X_1^2 + X_2^2 + X_3^2 + X_4^2}{A}\colon \frac{X_1^2 + X_2^2 - X_3^2 - X_4^2}{B}\colon \right.$$
$$\left. \colon \frac{X_1^2 - X_2^2 + X_3^2 - X_4^2}{C}\colon \frac{X_1^2 - X_2^2 - X_3^2 + X_4^2}{D}\right)$$

We call this the *distinguised kernel*: the kernel of the first half of doubling. Indeed, $U = (A : B : C : D)$ and we recover the first three steps $\mathcal{C}_{\mathcal{I}((\mathcal{A}:\mathcal{B}:\mathcal{C}:\mathcal{D}))} \circ \mathcal{H} \circ \mathcal{S}$ of the doubling map on fast Kummer surfaces.

Remark 3.2. Though the formulæ for $(2,2)$-isogenies on fast Kummer surfaces are extremely compact, we remark that the final scaling requires the computation of square roots in $\overline{\Bbbk}$. If R, S are the 2-torsion points generating the isogeny, let P, $Q \in \mathcal{K}$ be such that $P = [2]_* R$, $Q = [2]_* S$. The coordinates of P, Q contain the square roots needed for the final scaling (up to a projective factor). Extracting these square roots from the coordinates of the 4-torsion points lying above R, S remains an open problem.

3.2. **The general case: odd N.** From this point forward, we suppose N is an odd prime. Since N is odd, the (N,N)-isogeny Φ restricts to an isomorphism of 2-torsion subgroups $\mathcal{J}[2] \to \mathcal{J}'[2]$. Furthermore, since Φ is an isogeny of p.p.a.v. it is compatible with the N-Weil pairing by definition, and so it maps the symplectic structure on $\mathcal{J}[2]$ associated with the fast Kummer $\mathcal{K}$ onto a symplectic 2-torsion structure on $\mathcal{J}'[2]$, which is associated with a fast Kummer $\mathcal{K}'$. The isogeny Φ therefore descends to a morphism $\phi : \mathcal{K} \to \mathcal{K}'$ of fast Kummers. Our goal is to construct explicit equations for ϕ.

To do so, we follow the strategy taken by Cassels and Flynn [11, §9], Bruin, Flynn and Testa [9], Nicholls [40, §5], and Flynn [27], adapting it to the case of *fast* Kummer surfaces. We will observe that the special forms of the affine translation maps σ_i are very helpful in this setting, and lead to nice results.

In practice, we are given a fast Kummer $\mathcal{K}$ and the image $\pi(G)$ of an (N,N)-subgroup G of $\mathcal{J}$ in $\mathcal{K}$. There exists a fast Kummer $\mathcal{K}' \cong (\mathcal{J}'/G)/\langle \pm 1 \rangle$, and our goal is to find $\mathcal{K}'$ and the map $\phi : \mathcal{K} \to \mathcal{K}'$ induced by the quotient (N,N)-isogeny Φ with kernel G. Crucially, ϕ "commutes" with the action by 2-torsion points, in the sense of the following definition.

Definition 3.3. An *isogeny of fast Kummer surfaces* is a morphism $\phi : \mathcal{K} \to \mathcal{K}'$ induced by an isogeny $\Phi : \mathcal{J} \to \mathcal{J}'$ such that when lifted to a map on the ambient space, we have

$$\phi \circ U_i^{\mathcal{K}} = U_i^{\mathcal{K}'} \circ \phi \quad \text{and} \quad \phi \circ V_i^{\mathcal{K}} = V_i^{\mathcal{K}'} \circ \phi \quad \text{for } i = 1, 2\,,$$

where U_i and V_i are as defined in Section 2.4.

We want to compute $\phi : \mathcal{K} \to \mathcal{K}'$, but $\mathcal{K}'$ is unknown. However, as $\mathcal{K}$ and $\mathcal{K}'$ are both embedded in $\mathbb{P}^3$, ϕ must be defined by forms of degree N, and it must commute with the actions of the U_i and V_i. This imposes heavy constraints on the shape of the forms defining ϕ, and we can hope to interpolate them using linear algebra given the action of G, and therefore to interpolate the image Kummer $\mathcal{K}'$ by pushing the theta constants $(a\colon b\colon c\colon d)$ through the isogeny.

Let $\mathcal{K}[N]$ be the image of $\mathcal{J}[N]$ on $\mathcal{K}$ and fix $R, S \in \mathcal{K}[N]$. From this point forward, we write $\langle R, S \rangle \subset \mathcal{K}[N]$ for the image of the subgroup G of $\mathcal{J}[N]$ generated by the preimages of R, S. By abuse of notation, we say that R, S are N-torsion points on $\mathcal{K}$.

The first step is to compute two sets of homogenous forms of degree N in the coordinates of $\mathcal{K}$ that are invariant under translation by R and by S. The following lemma, due to Nicholls [40, §5.8.4], describes how we can use the biquadratic forms associated to the Kummer surface introduced in Section 2.3 to construct these homogenous forms.

Lemma 3.4. *Fix Kummer surface $\mathcal{K}$ with coordinates X_1, X_2, X_3, X_4 and associated biquadratic forms $B_{i,j}$ for $1 \le i, j \le 4$. Let N be an odd prime number, and fix a point $R \in \mathcal{K}[N]$ of order N.*

We denote by $I \in \{1,2,3,4\}^N$ a list of indices $I = (i_1, \ldots, i_N)$, where $i_j \in \{1,2,3,4\}$ for all $j = 1, \ldots, N$. Letting τ be a permutation of $\{1, \ldots, N\}$, we write

$$\tau(I) = \tau((i_1, \ldots, i_N)) := (i_{\tau(1)}, \ldots, i_{\tau(N)}).$$

Then, for each $I \in \{1,2,3,4\}^N$ we define

$$F_I := \sum_{\tau \in C_N} X_{i_{\tau(1)}} \cdot \prod_{k=1}^{(N-1)/2} B_{i_{\tau(2k)}, i_{\tau(2k+1)}}(X_1, X_2, X_3, X_4;\ kR)\,,$$

where C_N is the cyclic group of order N. Then, the set $\mathcal{F}_R := \{F_I\}$ is the set of the homogenous forms of degree N invariant under translation by R.

Applying the lemma above to N-torsion points R and S, we obtain the two sets $\mathcal{F}_R$ and $\mathcal{F}_S$. The homogeneous forms of degree N in each set will generate a space of dimension $m \ge 4$. The next step is to compute a basis for these two spaces, say $F_1^R, \ldots, F_m^R$ is a basis for the space generated by the homogenous forms in $\mathcal{F}_R$, and $F_1^S, \ldots, F_m^S$ a basis for the space generated by $\mathcal{F}_S$.

The intersection of these spaces contains homogenous forms of degree N that are invariant under translation by any point in the kernel G of our (N,N)-isogeny. The intersection will be of dimension 4, and a basis for this intersection gives an (N,N)-isogeny $\psi : \mathcal{K} \to \widetilde{\mathcal{K}}$. Explicitly, the third step is to compute a basis of this intersection, say f_1, f_2, f_3, f_4. Then, our (N,N)-isogeny is given by $\psi = (f_1 : f_2 : f_3 : f_4)$.

We note that $\widetilde{\mathcal{K}}$ may not be in the correct form given by Equation (1). When computing chains of isogenies, however, it is important to ensure that our (N,N)-isogenies have domain and image in the same form. Therefore, the last step is to apply a linear transformation $\mathbf{M} : \widetilde{\mathcal{K}} \to \mathcal{K}'$, where $\mathcal{K}'$ is a fast Kummer surface. Post-composing the map ψ with this linear transformation gives a $\overline{\mathbb{k}}$-rational (N,N)-isogeny $\phi : \mathcal{K} \to \mathcal{K}'$ between fast Kummer surfaces generated by kernel $G = \langle R, S \rangle$.

Remark 3.5. The compactness and efficiency of our isogeny formulae is determined by the choice of basis we make for the spaces generated by the forms in $\mathcal{F}_R$ and $\mathcal{F}_S$. An open question that arises from this work, therefore, is finding a solution to the following problem: let $f_1, \ldots, f_n \in \mathbb{Q}(a_1, \ldots, a_k)[x_1, \ldots, x_m]$ be a basis of polynomials defined over a function field. Find a "nice" basis $g_1, \ldots, g_n$ where $g_1, \ldots, g_n$ are $\mathbb{Q}(a_1, \ldots, a_k)$-linear combinations of the f_i.

4. Explicit $(3,3)$-isogenies on fast Kummers

We now specialise the discussion in Section 3 to $N = 3$ to construct $(3,3)$-isogenies between fast Kummer surfaces.

Let $\mathcal{J} = \mathrm{Jac}(C)$ be the Jacobian of a genus 2 curve C defined over $\overline{\mathbb{k}}$. Suppose we have a $(3,3)$-subgroup of $\mathcal{J}[3]$, which induces an isogeny ϕ on the corresponding fast Kummer surface $\mathcal{K} = \mathcal{J}/\{\pm\}$ with kernel $G = \langle R, S \rangle$ for some $R, S \in \mathcal{K}[3]$ (i.e., G is the image of the $(3,3)$-subgroup in $\mathcal{K}$).

Exploiting the fact that ϕ is an isogeny of fast Kummer surfaces, we obtain the following lemma, demonstrating that it is determined by five $\overline{\mathbb{k}}$-rational functions in the coordinates of $O_{\mathcal{K}} = (a : b : c : d)$, R and S.

Lemma 4.1. *Let R and S be distinct 3-torsion points on $\mathcal{K}$ generating a $(3,3)$-subgroup $G \subset \mathcal{K}[3]$, and set $O_{\mathcal{K}} = (a\colon b\colon c\colon d)$. The $(3,3)$-isogeny of fast Kummer surfaces $\phi\colon \mathcal{K} \to \mathcal{K}'$ generated by kernel G is in the form*

$$(X_1 : X_2 : X_3 : X_4) \longmapsto (\phi_1(X_1, X_2, X_3, X_4) : \cdots : \phi_4(X_1, X_2, X_3, X_4)) \ ,$$

where

$$\begin{aligned}
\phi_1(X_1, X_2, X_3, X_4) &= X_1 \left(c_1 X_1^2 + c_2 X_2^2 + c_3 X_3^2 + c_4 X_4^2\right) + c_5 X_2 X_3 X_4 \ , \\
\phi_2(X_1, X_2, X_3, X_4) &= X_2 \left(c_2 X_1^2 + c_1 X_2^2 + c_4 X_3^2 + c_3 X_4^2\right) + c_5 X_1 X_3 X_4 \ , \\
\phi_3(X_1, X_2, X_3, X_4) &= X_3 \left(c_3 X_1^2 + c_4 X_2^2 + c_1 X_3^2 + c_2 X_4^2\right) + c_5 X_1 X_2 X_4 \ , \\
\phi_4(X_1, X_2, X_3, X_4) &= X_4 \left(c_4 X_1^2 + c_3 X_2^2 + c_2 X_3^2 + c_1 X_4^2\right) + c_5 X_1 X_2 X_3 \ ,
\end{aligned}$$

with $c_i \in \overline{\Bbbk}[a, b, c, d, r_1, r_2, r_3, r_4, s_1, s_2, s_3, s_4]$.

Proof. The isogeny ϕ is given by cubic forms [9, §5.1], that is, ϕ is defined by polynomials

$$\begin{aligned}
\phi_i = {} & c_{i,1} X_1^3 + c_{i,2} X_1 X_2^2 + c_{i,3} X_1 X_3^2 + c_{i,4} X_1 X_4^2 + c_{i,5} X_2 X_3 X_4 \\
& + c_{i,6} X_2 X_1^2 + c_{i,7} X_2^3 + c_{i,8} X_2 X_3^2 + c_{i,9} X_2 X_4^2 + c_{i,10} X_1 X_3 X_4 \\
& + c_{i,11} X_3 X_1^2 + c_{i,12} X_3 X_2^2 + c_{i,13} X_3^3 + c_{i,14} X_3 X_4^2 + c_{i,15} X_1 X_2 X_4 \\
& + c_{i,16} X_4 X_1^2 + c_{i,17} X_4 X_2^2 + c_{i,18} X_4 X_3^2 + c_{i,19} X_4^3 + c_{i,20} X_1 X_2 X_3 \ ,
\end{aligned}$$

where $c_{i,j}$ are $\overline{\Bbbk}$-rational functions in the coordinates of $O_{\mathcal{K}}$, R, and S for $1 \le i \le 4$ and $1 \le j \le 20$. We are looking for an isogeny of fast Kummer surfaces in the sense of Definition 3.3, and compatibility with the translations by 2-torsion thus forces

$$\sigma_i'((\phi_1 \colon \phi_2 \colon \phi_3 \colon \phi_4)) = \phi(\sigma_i(X_1 \colon X_2 \colon X_3 \colon X_4)) \tag{5}$$

for all $1 \le i \le 15$, where σ_i is the action of the 2-torsion point $T_i \in \mathcal{K}$, and similarly σ_i' is the action of $T_i' \in \mathcal{K}'$ (as defined in Section 2.4). Equation (5) gives rise to relations between the coefficients of the cubic monomials, from which we deduce that ϕ is of the form as in the statement of the lemma. See `section4/lemma-4_1.m` in the code accompanying this paper. Clearing denominators (as our Kummer surfaces lie in $\mathbb{P}^3$), we obtain the $c_i \in \overline{\Bbbk}[a, b, c, d, r_1, r_2, r_3, r_4, s_1, s_2, s_3, s_4]$. □

By Lemma 4.1, to determine explicit formulae for the $(3,3)$-isogeny ϕ generated by kernel $G = \langle R, S \rangle \subset \mathcal{K}$, it suffices to determine the coefficients $c_1, \ldots c_5$. We follow the method given in Section 3 and compute the G-invariant cubic forms. Define

$$\begin{aligned}
B_{ij}^R(X_1, X_2, X_3, X_4) &:= B_{i,j}(X_1, X_2, X_3, X_4; R), \\
B_{ij}^S(X_1, X_2, X_3, X_4) &:= B_{i,j}(X_1, X_2, X_3, X_4; S).
\end{aligned}$$

By Lemma 3.4, the cubic forms invariant under translation by R and S are given by

$$\begin{aligned}
F_{ijk}^R &:= X_i B_{jk}^R + X_j B_{ki}^R + X_k B_{ij}^R, \\
F_{ijk}^S &:= X_i B_{jk}^S + X_j B_{ki}^S + X_k B_{ij}^S,
\end{aligned}$$

respectively, where $1 \le i, j, k \le 4$. Let $\mathcal{F}_R = \{F_{ijk}^R\}_{1 \le i,j,k \le 4}$ and similarly define $\mathcal{F}_S$. The cubic forms in $\mathcal{F}_R$ and $\mathcal{F}_S$ each generate a space of dimension 8, for which

we choose a basis

$$\{F^R_{111},\ F^R_{234},\ F^R_{222},\ F^R_{134},\ F^R_{333},\ F^R_{124},\ F^R_{444},\ F^R_{123}\},$$

and similarly for $\mathcal{F}_S$. These spaces will intersect in a space of dimension 4, which will give a description of the $(3,3)$-isogeny. We compute a basis

$$\begin{aligned} f_1 &:= z_1F^R_{111} + z_2F^R_{234}\,, & f_2 &:= z_3F^R_{222} + z_4F^R_{134}\,,\\ f_3 &:= z_5F^R_{333} + z_6F^R_{124}\,, & f_4 &:= z_7F^R_{444} + z_8F^R_{123} \end{aligned}$$

for the intersection, with $z_1,\dots,z_8 \in \overline{\Bbbk}$ such that there exist $w_1,\dots,w_8 \in \overline{\Bbbk}$ with

$$\begin{aligned} f_1 &= w_1F^S_{111} + w_2F^S_{234}\,, & f_2 &= w_3F^S_{222} + w_4F^S_{134}\,,\\ f_3 &= w_5F^S_{333} + w_6F^S_{124}\,, & f_4 &= w_7F^S_{444} + w_8F^S_{123}\,. \end{aligned}$$

From this, we obtain a $(3,3)$-isogeny $\psi = (f_1 \colon f_2 \colon f_3 \colon f_4) : \mathcal{K} \to \widetilde{\mathcal{K}}$.

To move $\widetilde{\mathcal{K}}$ to the correct form, we first define

$$D_1 := (ab-cd)(ab+cd),\ D_2 := (ac-bd)(ac+bd),\ D_3 := (ad-bc)(ad+bc).$$

For a point $P = (x_1 \colon x_2 \colon x_3 \colon x_4)$, we define

$$\gamma(P) := (D_{23}(x_1x_2ab - x_3x_4cd) + D_{13}(x_1x_3ac - x_2x_4bd) + D_{12}(x_1x_4ad - x_2x_3bc)),$$

and $h_i(P)$ as the coordinates of $\mathcal{H}\circ\mathcal{S}(P)$, for $i = 1,2,3,4$.

Applying a linear transformation $\mathbf{M}$ to $\widetilde{\mathcal{K}}$, where $\mathbf{M}$ is defined as

$$\mathbf{M} := \begin{pmatrix} \alpha_1 & 0 & 0 & 0\\ 0 & \alpha_2/2 & 0 & 0\\ 0 & 0 & \alpha_1/2 & 0\\ 0 & 0 & 0 & 3\alpha_2/2 \end{pmatrix}$$

and where

$$\begin{aligned} \alpha_1 &:= D_3(\gamma(R)(s_4s_3ab - s_1s_2cd) - \gamma(S)(r_4r_3ab - r_1r_2cd)),\\ \alpha_2 &:= D_1(\gamma(R)(s_2s_3ad - s_1s_4bc) - \gamma(S)(r_2r_3ad - r_1r_4bc)), \end{aligned}$$

we get a simple and efficiently computable expression for our $(3,3)$-isogeny $\phi := \mathbf{M}(f_1,f_2,f_3,f_4)^T$, whose image is in the desired form. The formulæ for the intersection and linear transformation can be found and verified in the file `section4/linear-transform.m` in the accompanying code.

4.1. **Explicit formulae for $(3,3)$-isogenies.** We now give the explicit formulae for the isogeny $\phi : \mathcal{K} \to \mathcal{K}'$ generated by kernel $G = \langle R, S\rangle$. Recall from Lemma 4.1 that it suffices to give the explicit formulae for the coefficients $c_i \in \overline{\Bbbk}[a,b,c,d,r_1,r_2,r_3,r_4,s_1,s_2,s_3,s_4]$ for $i \in \{1,2,3,4,5\}$. We set

$$\begin{aligned} \beta_1 &:= D_{23}\big(\gamma(R)\cdot(s_3s_4ab - s_1s_2cd) - \gamma(S)\cdot(r_3r_4ab - r_1r_2cd)\big),\\ \beta_2 &:= h_1(R)\cdot h_2(S) - h_2(R)\cdot h_1(S). \end{aligned}$$

Then, maintaining the notation above and letting $D_{ij} := D_i \cdot D_j$, we find

$$\begin{aligned}
c_1 &= 2\beta_1 h_1(R)h_1(S),\\
c_2 &= \beta_1\big(h_1(R)h_2(S) + h_2(R)h_1(S)\big)\\
&\quad + \beta_2\big(\gamma(R)(s_3s_4ab - s_1s_2cd) + \gamma(S)(r_3r_4ab - r_1r_2cd)\big)D_{23},\\
c_3 &= \beta_1\big(h_1(R)h_3(S) + h_3(R)h_1(S)\big)\\
&\quad + \beta_2\big(\gamma(R)(s_2s_4ac - s_1s_3bd) + \gamma(S)(r_2r_4ac - r_1r_3bd)\big)D_{13},\\
c_4 &= \beta_1\big(h_1(R)h_4(S) + h_4(R)h_1(S)\big)\\
&\quad + \beta_2\big(\gamma(R)(s_2s_3ad - s_1s_4bc) + \gamma(S)(r_2r_3ad - r_1r_4bc)\big)D_{12},\\
c_5 &= 2\beta_2\gamma(S)\gamma(R).
\end{aligned}$$

Note that the $c_1, \ldots, c_5$ are symmetric in R and S, as one would expect.

4.2. **Evaluating points under the $(3,3)$-isogeny.** Consider the $(3,3)$-isogeny $\phi : \mathcal{K} \to \mathcal{K}'$ and assume the coefficients $c_1, \ldots, c_5$ have been computed. Given a point $P = (x_1 : x_2 : x_3 : x_4) \in \mathcal{K}$, the image $\phi(P) = (x_1' : x_2' : x_3' : x_4')$ is given by

$$\begin{aligned}
x_1' &:= x_1(c_1x_1^2 + c_2x_2^2 + c_3x_3^2 + c_4x_4^2) + c_5x_2x_3x_4,\\
x_2' &:= x_2(c_2x_1^2 + c_1x_2^2 + c_4x_3^2 + c_3x_4^2) + c_5x_1x_3x_4,\\
x_3' &:= x_3(c_3x_1^2 + c_4x_2^2 + c_1x_3^2 + c_2x_4^2) + c_5x_1x_2x_4,\\
x_4' &:= x_4(c_4x_1^2 + c_3x_2^2 + c_2x_3^2 + c_1x_4^2) + c_5x_1x_2x_3.
\end{aligned}$$

The fundamental theta constants of the image surface $\mathcal{K}'$ can be computed in the same way, i.e., as $\phi((a : b : c : d))$. Via Equation (2), we can then compute the constants E', F', G', H' defining the equation of the surface $\mathcal{K}'$.

4.3. **Implementation.** We implemented $(3,3)$-isogeny evaluation using the formulæ above. We give explicit operation counts for $\Bbbk = \mathbb{F}_q$, which will be necessary for our cryptographic application in Section 6. To optimise the computation, we implement the following algorithms:

(1) TriplingConstantsFromThetas: given fundamental theta constants $(a : b : c : d)$, compute *tripling constants* consisting of:
   - their inverses $(1/a : 1/b : 1/c : 1/d)$;
   - their squares $(a^2 : b^2 : c^2 : d^2)$;
   - squared dual theta constants $(A^2 : B^2 : C^2 : D^2)$; and
   - their inverses $(1/A^2 : 1/B^2 : 1/C^2 : 1/D^2)$.

   For $\Bbbk = \mathbb{F}_q$, this requires 12M, 4S, and 6a.

(2) Compute33Coefficients: given coordinates of R, S and the tripling constants, compute the coefficients $c_1, \ldots, c_5$ defining the $(3,3)$-isogeny. When $\Bbbk = \mathbb{F}_q$, this requires 76M, 8S, and 97a.

(3) Isogeny33Evaluate: given the coefficients $c_1, \ldots, c_5$, computes the image of a point $P \in \mathcal{K}$ under the corresponding $(3,3)$-isogeny (as explained in Section 4.2). For $\Bbbk = \mathbb{F}_q$, this requires 26M, 4S and 16a.

(4) ComputeImageThetas: given coefficients $c_1, \ldots, c_5$ and the tripling constants, compute the fundamental theta constants defining the image curve. For $\Bbbk = \mathbb{F}_q$, this requires 26M and 16a.

Details of the implementation can be found in the accompanying code.

4.4. **A note on $(5,5)$-isogenies on fast Kummers.** Suppose we now have a $(5,5)$-subgroup of $\mathcal{J}[5]$, which induces a $(5,5)$-isogeny ϕ on the corresponding fast Kummer surface $\mathcal{K} = \mathcal{J}/\{\pm\}$ with kernel $G = \langle R, S\rangle$ for some $R, S \in \mathcal{K}[5]$ (i.e., G is the image of the $(5,5)$-subgroup in $\mathcal{K}$).

Following the method in Section 3, we first compute the G-invariant quintic forms. Let $B^R_{i,j}$ and $B^S_{i,j}$ be as before (where now R, S are the 5-torsion points), and define

$$B^{2R}_{ij}(X_1, X_2, X_3, X_4) := B_{i,j}(X_1, X_2, X_3, X_4; 2R),$$
$$B^{2S}_{ij}(X_1, X_2, X_3, X_4) := B_{i,j}(X_1, X_2, X_3, X_4; 2S).$$

By Lemma 3.4 (and following Flynn [27]), the quintic forms invariant under translation by R are given by

$$\begin{aligned} F^R_{ijklm} := {} & X_i B^R_{jk} B^{2R}_{lm} + X_i B^R_{jl} B^{2R}_{km} + X_i B^R_{jm} B^{2R}_{kl} + \\ & X_i B^R_{kl} B^{2R}_{jm} + X_i B^R_{km} B^{2R}_{jl} + X_i B^R_{lm} B^{2R}_{jk} + \\ & X_j B^R_{ik} B^{2R}_{lm} + X_j B^R_{il} B^{2R}_{km} + X_j B^R_{im} B^{2R}_{kl} + \\ & X_j B^R_{kl} B^{2R}_{im} + X_j B^R_{km} B^{2R}_{il} + X_j B^R_{lm} B^{2R}_{ik} + \\ & X_k B^R_{ji} B^{2R}_{lm} + X_k B^R_{jl} B^{2R}_{im} + X_k B^R_{jm} B^{2R}_{il} + \\ & X_k B^R_{il} B^{2R}_{jm} + X_k B^R_{im} B^{2R}_{jl} + X_k B^R_{lm} B^{2R}_{ji} + \\ & X_l B^R_{jk} B^{2R}_{im} + X_l B^R_{ji} B^{2R}_{km} + X_l B^R_{jm} B^{2R}_{ki} + \\ & X_l B^R_{ki} B^{2R}_{jm} + X_l B^R_{km} B^{2R}_{ji} + X_l B^R_{im} B^{2R}_{jk} + \\ & X_m B^R_{jk} B^{2R}_{li} + X_m B^R_{jl} B^{2R}_{ki} + X_m B^R_{ji} B^{2R}_{kl} + \\ & X_m B^R_{kl} B^{2R}_{ji} + X_m B^R_{ki} B^{2R}_{jl} + X_m B^R_{li} B^{2R}_{jk}, \end{aligned}$$

where $1 \le i, j, k, l, m \le 4$. We similarly define F^S_{ijklm}: the quintic forms invariant under translation by S.

Let $\mathcal{F}_R = \{F^R_{ijklm}\}$ and $\mathcal{F}_R = \{F^S_{ijklm}\}$. The quintic forms in $\mathcal{F}_R$ and $\mathcal{F}_S$ each generate a space of dimension 12, for which we choose a basis

$$\begin{aligned} \Big\{ & F^R_{13344},\ F^R_{14444},\ F^R_{23334},\ F^R_{23444},\ F^R_{22244},\ F^R_{23333},\ F^R_{23344},\ F^R_{24444}, \\ & F^R_{22344},\ F^R_{33333},\ F^R_{33344},\ F^R_{34444},\ F^R_{22444},\ F^R_{33334},\ F^R_{33444},\ F^R_{44444} \Big\}, \end{aligned}$$

and similarly for $\mathcal{F}_S$. These spaces intersect in a space of dimension 4, which gives a description of the $(5,5)$-isogeny.

5. Generating (N^k, N^k)-subgroups

In the remaining two sections, we turn to building a hash function based on the $(3,3)$-isogenies derived in the previous section. The hash function will compute $(3^k, 3^k)$-isogenies as chains of $(3,3)$-isogenies, and will start each such chain by computing a $(3^k, 3^k)$-subgroup on a fast Kummer surface. This section describes how such (N^k, N^k)-subgroups can be computed (for prime number N and integer $k \ge 1$) in a way that is amenable to efficient and secure cryptographic implementations.

We do this in two steps: first, in Section 5.1 we show how to compute a *symplectic* basis for the N^k-torsion on the Jacobian $\mathcal{J}$, which we then push down to the corresponding fast Kummer surface $\mathcal{K} = \mathcal{J}/\{\pm 1\}$; and second, we use this

basis to compute the generators R, S of the (N^k, N^k)-subgroup G using the *three-dimensional differential addition chain* introduced in Section 5.2.

5.1. **Generating a symplectic basis on $\mathcal{K}$.** We first compute a symplectic basis for the N^k-torsion on the Jacobian $\mathcal{J}$ of the genus-2 curve $C_{\lambda,\mu,\nu}$ corresponding to $\mathcal{K}$.

Definition 5.1. A basis $\{Q_1, Q_2, Q_3, Q_4\}$ for $\mathcal{J}[D]$ is *symplectic* with respect to the D-Weil pairing if

$$e_D(Q_1, Q_3) = e_D(Q_2, Q_4) = \zeta$$

where ζ is a primitive D-th root of unity, and $e_D(Q_i, Q_j) = 1$ otherwise.

We compute the symplectic basis for $\mathcal{J}[N^k]$ by generating N^k-torsion points on $\mathcal{J}$ until the N^k-Weil pairing condition given in Definition 5.1 is satisfied.[2] These points are then pushed down to $\mathcal{K}$ via

$$\begin{aligned} &\pi\colon \mathcal{J} \to \mathcal{K}, \\ &\quad P \mapsto \left(\mathcal{C}_{\mathcal{I}(\mathcal{O}_\mathcal{K})} \circ \mathcal{H} \circ \mathcal{C}_{(\mathcal{I}\circ\mathcal{H}\circ\mathcal{S})(\mathcal{O}_\mathcal{K})}\mathcal{S} \circ \mathcal{H} \circ \kappa\right)(P). \end{aligned}$$

Here, $\mathcal{C}$, $\mathcal{H}$, $\mathcal{S}$ and $\mathcal{I}$ are the Kummer operations defined in Section 2.5 and κ is a map taking points $P = (x^2 + u_1 x + u_0, v_1 x + v_0)$ on $\mathcal{J}$ in Mumford coordinates to the *squared* Kummer surface $\mathcal{K}^{\mathrm{Sqr}}$, as follows

$$\begin{aligned} &\kappa\colon \mathcal{J} \mapsto \mathcal{K}^{\mathrm{Sqr}}, \\ &\quad (x^2 + u_1 x + u_0, v_1 x + v_0) \mapsto (X_1 \colon X_2 \colon X_3 \colon X_4)\,, \end{aligned}$$

with

$$\begin{aligned} X_1 &= a^2(u_0(\mu - u_0)(\lambda + u_1 + \nu) - v_0^2)\,, \quad X_2 = b^2(u_0(\nu\lambda - u_0)(1 + u_1 + \mu) - v_0^2)\,, \\ X_3 &= c^2(u_0(\nu - u_0)(\lambda + u_1 + \mu) - v_0^2)\,, \quad X_4 = d^2(u_0(\mu\lambda - u_0)(1 + u_1 + \nu) - v_0^2)\,. \end{aligned}$$

The map κ is due to Bisson, Cosset and Robert [5], while the subsequent operations that map from $\mathcal{K}^{\mathrm{Sqr}}$ to $\mathcal{K}$ appear in Renes and Smith [45, Section 4.3]. Note that the map π corresponds to a $(2,2)$-isogeny, which will not affect the order of the basis points if N is coprime to 2. If, however, $2 \mid N$ then one must additionally check the order of the image points on $\mathcal{K}$.

Remark 5.2. For applications where N, k, and the domain Kummer $\mathcal{K}$ are fixed, the symplectic basis for $\mathcal{J}[N^k]$ can be computed as part of the set-up once and for all, and the image of these basis points (under π) can be hardcoded as system parameters. For instance, this will be in the case for our cryptographic application in Section 6. In these scenarios, optimising the efficiency of the operations in this subsection is not a priority; the important goal is to optimise the efficiency of the *online* part of the (N^k, N^k)-subgroup generation procedure, which amounts to optimising the three-dimensional differential addition chain in the following subsection.

[2] In our implementation, we compute the N^k-Weil pairing of points on $\mathcal{J}[N^k]$ using `MAGMA`'s in-built functionality.

5.2. **Three-dimensional differential addition chains.** Let $Q_1, Q_2, Q_3, Q_4 \in \mathcal{K}$ be the images of a symplectic basis for $\mathcal{J}[N^k]$ under the map π described above. In this subsection we show how to use this basis to compute the two generators R and S of our (N^k, N^k)-subgroup.

As a first simplification, we restrict to (N^k, N^k)-subgroups with generators of the form

$$\begin{aligned} R &= Q_1 + [\alpha]Q_3 + [\beta]Q_4, \\ S &= Q_2 + [\beta]Q_3 + [\gamma]Q_4, \end{aligned} \tag{6}$$

where $\alpha, \beta, \gamma \in \mathbb{Z}/N^k\mathbb{Z}$. There are N^{3k} such (N^k, N^k)-subgroups (for example, see [35, Table 1]), and there are $O(N^{3k-1})$ subgroups that we lose by imposing this restriction [35, Def. 3]. In other words, at least half of the (N^k, N^k)-subgroups can be obtained with kernel generators of the form in (6), so in a cryptographic context we lose at most one bit of security by simplifying in this manner.

The remainder of this subsection presents the 3DAC algorithm that allows us to compute the kernel generators R and S via Equation (6). Our task is to define an algorithm that computes $P_1 + [\beta]P_2 + [\gamma]P_3$ for given scalars $\beta, \gamma \in \mathbb{Z}/N^k\mathbb{Z}$ and for the points P_1, P_2 and P_3 on $\mathcal{K}$. The analogous computation on $\mathcal{J}$ could utilise a straightforward 3-way multiexponentation algorithm, but on the Kummer surface we need a three-dimensional *differential addition chain*. Such an addition chain is only allowed to include the computation of the sum $Q+R$ if the difference $\pm(Q-R)$ has already been computed at a previous stage.

Three-dimensional differential addition chains have been studied previously by Rao [42] and more generally by Hutchinson and Karabina [32]. However, both of those works study the general scenario whereby three scalars are in play. Viewing Equation (6), we see that there is no scalar multiplication of the point P_1 in our case, which allows for some convenient simplifications. Moreover, the chain due to Rao [42] is non-uniform and the chain due to Hutchinson and Karabina [32] is not fixed length unless the input scalars are. Our algorithm 3DAC satisfies both of these properties regardless of the input scalars, making it secure for use in cryptographic applications, such as the hash function we present in Section 6. We remark that these properties would not be necessary for a hash function involving only public data, but for other cryptographic applications where inputs to the hash function (or, more broadly, the scalars β and γ) are secret, such as key derivation functions, these properties are an imperative first-step towards protecting the secret data from side-channel attacks.

We derived the 3DAC chain by extending the two-dimensional differential addition chain due to Bernstein [3], the fastest known two-dimensional differential addition chain that is fixed length and uniform. Beyond the points P_1, P_2, P_3, 3DAC also needs seven additional combinations of sums and/or differences of these three points. We specify the full ten-tuple of inputs as

$$\mathcal{D} := \big(P_1, P_2, P_3, P_2{+}P_3, P_2{-}P_3, P_1{-}P_2, P_1{-}P_2, [2](P_2{+}P_3), P_1{+}P_2{+}P_3, P_1{-}P_2{-}P_3\big),$$

for points P_1, P_2, $P_3 \in \mathcal{K}$. Note that, in line with Remark 5.2, these additional sums and differences can be (pre)computed on $\mathcal{J}$ and their image in $\mathcal{K}$ specified as part of the system parameters.

Given the tuple $\mathcal{D}$ above and the two scalars $\beta, \gamma \in \mathbb{Z}/N^k\mathbb{Z}$, our 3DAC algorithm computes the point $P_1 + [\beta]P_2 + [\gamma]P_3$ on $\mathcal{K}$ using $3\ell - 2$ pseudo-additions and $\ell - 1$

pseudo-doublings on $\mathcal{K}$, where ℓ is the bitlength of N^k. See Appendix B for the full description of the algorithm.

6. A hash function from $(3,3)$-isogenies

Isogenies between *superspecial* Jacobians of genus 2 curves have been proposed for use in post-quantum isogeny-based cryptography (e.g., [13, 29]). We follow suit and henceforth restrict our attention superspecial Jacobians defined over $\overline{\Bbbk} = \overline{\mathbb{F}}_p$.

Definition 6.1. We say that the Jacobian $\mathcal{J}$ of a genus 2 curve is *superspecial* if the Hasse–Witt matrix $M \in \mathbb{F}_p^{2\times 2}$ vanishes identically. We say that a Kummer surface $\mathcal{K} = \mathcal{J}/\{\pm 1\}$ is *superspecial* if the corresponding Jacobian $\mathcal{J}$ is superspecial.

It can be shown that every superspecial $\mathcal{J}/\overline{\mathbb{F}}_p$ is $\overline{\mathbb{F}}_p$-isomorphic to a Jacobian defined over $\mathbb{F}_{p^2}$. Similarly, the corresponding superspecial Kummer surface $\mathcal{K}/\overline{\mathbb{F}}_p$ is $\overline{\mathbb{F}}_p$-isomorphic to a Kummer surface with model defined over $\mathbb{F}_{p^2}$.

As an application to exhibit our algorithms, we construct a fundamental cryptographic primitive: a hash function. The first isogeny-based hash function was introduced by Charles, Goren and Lauter who use isogenies between supersingular ellipitic curves [14]. The use of higher dimensional isogenies between superspecial Jacobians of genus-2 curves to construct a variant of the Charles–Goren–Lauter (CGL) hash function was previously explored by Castryck, Decru and Smith [13] using $(2,2)$-isogenies. They argue that although the computation of higher dimensional isogenies is more expensive, breaking the security of the hash function requires $\widetilde{O}(p^{3/2})$ time, rather than $\widetilde{O}(p^{1/2})$ time as in the CGL hash function. Therefore, smaller parameters can be used to obtain the same security. Following this, Castryck and Decru [12] use multiradical formulae for $(3,3)$-isogenies to construct such a hash function and obtain an asymptotic speed-up of around a factor of 9. Later work by Decru and Kunzweiler [23] construct a hash function using $(3,3)$-isogenies between general Kummer surfaces by improving on the formulae given by Bruin, Flynn and Testa [9].

In this section, we describe a variant of the CGL hash function [14], called KuHash, that uses the formulae introduced in Section 4 to compute chains of $(3,3)$-isogenies between fast Kummer surfaces. We obtain a speed-up of around $1.4 - 2.3\mathtt{x}$, and around $27.0 - 28.3\mathtt{x}$ compared to the Castryck–Decru and Decru–Kunzweiler hash functions, respectively, for security levels $\lambda = 128, 192$ and 256.

6.1. Chains of $(3,3)$-isogenies.

We first present the Isogeny33Chain routine (Algorithm 2) for computing chains of $(3,3)$-isogenies. Though we use it as a building block for a hash function, we note that it can be also be used in a variety of cryptographic applications. In particular, we have been careful to ensure that each component of the algorithm is amenable to constant-time cryptographic software.

Tripling algorithm. We start by presenting TPL (Algorithm 1), the algorithm to compute $[3]P$ from a point $P \in \mathcal{K}$ and the associated tripling constants (as defined in Section 4.3). This algorithm requires $26\mathtt{M}$, $12\mathtt{S}$ and $32\mathtt{a}$.

Naïve strategies. Given a $(3^k, 3^k)$-subgroup $G = \langle R, S\rangle \subset \mathcal{K}[3^k]$, we use TPL and the algorithms from Section 4.3 to compute an isogeny with kernel $\langle R, S\rangle$ as a chain of $(3,3)$-isogenies of length k. A naïve way of doing so is the following. Set $P_0 := R$, $Q_0 := S$ and $\mathcal{K}_0 := \mathcal{K}$, and then execute the following four steps for $i = 1$ to k:

Algorithm 1 $\mathsf{TPL}(P, \mathtt{TC})$:

Input: Point $P \in \mathcal{K}$ and tripling constants $\mathtt{TC}$ (with i-th entry denoted $\mathtt{TC}_i$)
Output: Point $Q \in \mathcal{K}$ where $Q = [3]P$.

$R \leftarrow \mathcal{S}(P)$
$R \leftarrow \mathcal{H}(R)$
$Q \leftarrow \mathcal{S}(R)$
$Q \leftarrow \mathcal{C}_{\mathtt{TC}_5}(Q)$
$Q \leftarrow \mathcal{H}(Q)$
$Q \leftarrow \mathcal{C}_{\mathtt{TC}_4}(Q)$
$Q \leftarrow \mathcal{S}(Q)$
$Q \leftarrow \mathcal{H}(Q)$
$S \leftarrow \mathcal{C}_{\mathtt{TC}_5}(R)$
$Q \leftarrow \mathcal{C}_S(Q)$
$Q \leftarrow \mathcal{H}(Q)$
$Q \leftarrow \mathcal{C}_{\mathcal{I}(P)}(Q)$
return Q

(1) Compute the tripling constants on $\mathcal{K}_{i-1}$ using TriplingConstantsFromThetas
(2) Compute 3-torsion points $(P_i, Q_i) := (3^{k-i}R, 3^{k-i}S)$ using $k - i$ repeated applications of TPL on R and S.
(3) Compute the $(3,3)$-isogeny $\varphi_i : \mathcal{K}_{i-1} \to \mathcal{K}_i$ with kernel $\langle P_i, Q_i \rangle$, and the images of P_{i-1}, Q_{i-1} under this isogeny using Compute33Coefficients and Isogeny33Evaluate.
(4) Compute the theta constants of the image $\mathcal{K}_i$ of φ_i using ComputeImageThetas.

The $(3^k, 3^k)$-isogeny with kernel G will be given by $\varphi_k \circ \cdots \circ \varphi_1 : \mathcal{K}_0 \to \mathcal{K}_k$.

Optimal strategies. A more efficient way to compute isogeny chains is to use *optimal strategies* [25]. This allow us to reduce the number of executions of TPL needed to compute the kernel at each step in the chain by storing intermediate points obtained during the repeated triples and pushing them through each isogeny. In our case, the cost of tripling is around 1.1x the cost of computing the image of a point under the isogeny, and so we shift the cost in this way to obtain the strategies (see [25] for further details on optimising this approach). We give this algorithm in detail in Algorithm 2, and note that invoking the optimal strategies results in a 3.7–6.6× reduction of the cost to compute a chain of $(3,3)$-isogenies for the set of parameters we specify in Section 6.3.

6.2. **A cryptographic hash function.** We are now ready to present the hash function KuHash that uses chains of $(3,3)$-isogenies between fast Kummer surfaces. For a fixed security parameter λ, the hash function parses the message into three scalars $\alpha, \beta, \gamma \in \mathbb{Z}/3^k\mathbb{Z}$ which are fed into the 3DAC algorithm to compute a $(3^k, 3^k)$-subgroup $G = \langle R, S \rangle$. This is then used to compute the corresponding $(3^k, 3^k)$-isogeny $\varphi : \mathcal{K} \to \mathcal{K}'$, and the output of the hash function is the fundamental theta constants of the image surface $\mathcal{K}'$.

To optimise our hash function, we want to ensure that the $(3^k, 3^k)$-isogeny is $\mathbb{F}_{p^2}$-rational. Given a security parameter λ, we choose a suitably sized prime $p =$

Algorithm 2 $\mathsf{Isogeny33Chain}(k, O_{\mathcal{K}}, R, S, \texttt{strategy})$:

Input: Fundamental theta constants $O_{\mathcal{K}} = (a : b : c : d)$ defining Kummer surface $\mathcal{K}$, generators $R, S \in \mathcal{K}$ of $(3^k, 3^k)$-subgroup for $k \geq 1$, and optimal strategy `strategy`.
Output: Fundamental theta constants defining image Kummer surface $\mathcal{K}'$ of $(3^k, 3^k)$-isogeny $\varphi : \mathcal{K} \to \mathcal{K}'$ with $\ker \varphi = \langle R, S \rangle$.

for $e = d - 1$ to 1 **do**
  $P, Q = R, S$
  `pts` $= [\,]$
  `inds` $= [\,]$
  $i \leftarrow 0$
  `TC` $\leftarrow \mathsf{TriplingConstantsFromThetas}(O_{\mathcal{K}})$
  while $i < d - e$ **do**
    Append $[R, S]$ to `pts`
    Append i to `inds`
    $m \leftarrow$ `strategy`$[d - i - e + 1]$
    for $j = 1$ to m **do**
      $R \leftarrow \mathsf{TPL}(R, \texttt{TC})$
      $S \leftarrow \mathsf{TPL}(S, \texttt{TC})$
    $i \leftarrow i + m$
  `cs` $\leftarrow \mathsf{Compute33Coefficients}(P, Q, \texttt{TC})$
  $O_{\mathcal{K}} \leftarrow \mathsf{ComputingImageThetas}(\texttt{cs})$
  for $[P_1, P_2]$ in `pts` **do**
    $P_1 \leftarrow \mathsf{Isogeny33Evaluate}(P_1, \texttt{cs})$
    $P_2 \leftarrow \mathsf{Isogeny33Evaluate}(P_2, \texttt{cs})$
  if `pts` not empty **then**
    $[R, S] \leftarrow$ `pts`$[-1]$
    $i \leftarrow$ `inds`$[-1]$
    Remove last element from `pts` and `inds`
return $O_{\mathcal{K}}$

$16f \cdot 3^k - 1 \approx 2^\lambda$, where f is a small cofactor, and take a small pseudo-random walk[3] from the superspecial Jacobian of the curve $C_0 : y^2 = x^6 + 1$ to arrive at our starting Jacobian $\mathcal{J} = \mathrm{Jac}(C)$. Since $\mathcal{J}(\mathbb{F}_{p^2}) \cong (\mathbb{Z}/(p+1)\mathbb{Z})^4$, the Rosenhain invariants of C are all rational in $\mathbb{F}_{p^2}$ and we use these to compute the $O_{\mathcal{K}} = (a : b : c : d)$, i.e. the fundamental theta constants of the starting Kummer surface $\mathcal{K} = \mathcal{J}/\{\pm 1\}$. Using the methods described in Section 5.1, we compute a symplectic basis for $\mathcal{J}[3^k]$, which (together with the auxiliary sums and differences defined in Section 5.2) is pushed down to our starting fast Kummer surface $\mathcal{K}$ via $\pi : \mathcal{J} \to \mathcal{K}$ to obtain the two tuples of points $\mathcal{D}_R$ and $\mathcal{D}_S$. The setup routine outputs $\texttt{gens} = \{\mathcal{D}_R,\ \mathcal{D}_S\}$ and $\texttt{data} = \{k,\ \lceil \lambda / \log(3) \rceil,\ O_{\mathcal{K}}\}$.

The hash function takes as input `data`, `gens` and a message[4] $\texttt{msg} = (\alpha, \beta, \gamma)$, where $\alpha, \beta, \gamma \in \mathbb{Z}/3^k\mathbb{Z}$, and the output of the hash function is a tuple of elements

[3] In our implementation, we used `MAGMA`'s inbuilt Richelot isogeny routine to take 20 $(2,2)$-isogenies away from C_0.

[4] In practice, the input message `msg` to the hash function would be a bit string, which would then be parsed into the scalars α, β, γ.

$\mathbb{F}_{p^2}$, namely the fundamental theta constants of the image Kummer surface $\mathcal{K}'$ under the $(3^k, 3^k)$-isogeny defined by scalars (α, β, γ). The output is of size $8\log(p)$ without normalising the theta constants (a', b', c', d'), and of size $6\log(p)$ with normalisation $(a'/d', b'/d', c'/d')$, which comes at a cost of one inversion and 3M. We fully specify KuHash in Algorithm 3.

Algorithm 3 KuHash(msg, data, gens)

Input: A message msg, auxiliary data data and generators gens of $\mathcal{K}[3^k]$.
Output: Fundamental theta constants (a', b', c', d') of image Kummer surface $\mathcal{K}'$

1: Parse msg as α, β, γ.
2: Parse data as $k, \ell, \mathcal{O}_\mathcal{K}$.
3: Parse gens as two sets $\mathcal{D}_R$, $\mathcal{D}_S$ (see Section 5.2).
4: $R \leftarrow \mathsf{3DAC}(\mathcal{D}_R, \alpha, \beta, \ell, O_\mathcal{K})$
5: $S \leftarrow \mathsf{3DAC}(\mathcal{D}_S, \beta, \gamma, \ell, O_\mathcal{K})$
6: $(a' : b : c' : d') \leftarrow \mathsf{Isogeny33Chain}(k, O_\mathcal{K}, R, S)$
7: **return** (a', b', c', d')

6.3. **Implementation.** We implement KuHash and give parameters for security levels $\lambda = 128, 192$, and 256.

Security. Let λ be the security parameter and $p \approx 2^\lambda$. We follow the discussion in [13, §7.4] to determine the security of the hash function KuHash. In particular, the security of our hash function is not affected by taking $N = 3$ rather that $N = 2$, and as a result the security of our hash function relies on similar problems, namely Problem 6.2 and Problem 6.3 below.

Problem 6.2. *Given two superspecial genus 2 curves C_1, C_2 defined over $\mathbb{F}_{p^2}$ find a $(3^k, 3^k)$-isogeny between $\mathrm{Jac}(C_1)$ and $\mathrm{Jac}(C_2)$.*

Problem 6.3. *Given a superspecial genus 2 curves C_1 defined over $\mathbb{F}_{p^2}$, find*

- *a curve C_2 and a $(3^k, 3^k)$-isogeny $\mathrm{Jac}(C_1) \to \mathrm{Jac}(C_2)$,*
- *a curve C_2' and a $(3^{k'}, 3^{k'})$-isogeny $\mathrm{Jac}(C_1) \to \mathrm{Jac}(C_2')$,*

such that $\mathrm{Jac}(C_2)$ and $\mathrm{Jac}(C_2')$ are $\overline{\mathbb{F}}_p$-isomorphic. Here, we can have $k = k'$, but the kernels of the corresponding isogenies must be different.

Previous works take the general Pollard-ρ attack to be the best classical attack against these problems, which runs in $\widetilde{O}(p^{3/2})$. We take a more conservative approach and consider the Costello–Smith algorithm [20] to be the best classical attack, which runs in $\widetilde{O}(p)$. The best quantum attack is based on Grover's claw-finding algorithm and runs in $\widetilde{O}(p^{1/2})$ [20, Theorem 2].

Considering these attacks, we obtain the following parameters:

- $\lambda = 128$: $p = 5 \cdot 2^4 \cdot 3^k - 1$ with $k = 75$;
- $\lambda = 192$: $p = 37 \cdot 2^4 \cdot 3^k - 1$ with $k = 115$;
- $\lambda = 256$: $p = 11 \cdot 2^4 \cdot 3^k - 1$ with $k = 154$.

Cost Metric. To benchmark KuHash, we count $\mathbb{F}_p$-operations. Indeed, our implementation in Python/SageMath will call underlying $\mathbb{F}_p$-operations to compute $\mathbb{F}_{p^2}$-operations. For simplicity, our cost metric will take $\mathtt{M} = \mathtt{S}$ and ignore $\mathtt{a}$, as additions have only a very minor impact on performance. Note that it is relatively straightforward to convert this cost into a more fine-grained metric (e.g., bit operations, cycle counts, etc.).

Results. We ran KuHash in SageMath version 10.1 using Python 3.11.1 and record the cost, as per the cost metric above, averaging over 100 random inputs for each prime size. We present the results in Table 1. Taking the $\mathbb{F}_{p^2}$-operation count from [23, §3.2] and using our cost metric, the cost of computing the coefficients of Decru and Kunzweiler's $(3,3)$-isogeny is 6702 (assuming 1 $\mathbb{F}_{p^2}$-multiplication is equivalent to 3 $\mathbb{F}_p$-multiplications). We use this to obtain a lower bound on the cost of the Decru–Kunzweiler hash function. Though this is a lower bound, we see in Table 1 that the cost already exceeds the total cost of KuHash.

The codebase for the other $(3,3)$-isogeny CGL hash variant by Castryck and Decru [12] uses Gröbner basis calculations, which is not realistic to convert to a fixed number of $\mathbb{F}_p$-operations. Therefore, for fair comparison, we ran all three hash functions in `MAGMA` V2.25-6 on Intel(R) Core™ i7-1065G7 CPU @ 1.30GHx × 8 with 15.4 GiB memory, and record the time taken to run the hash functions for the different λ in Table 1. We again average over 100 random inputs for each prime size.

	KuHash		[23]		[12]
Prime	Cost	Time (s)	Cost	Time (s)	Time (s)
$\lambda = 128$	177956	0.22	> 536160	5.93	0.50
$\lambda = 192$	286636	0.35	> 703710	9.73	0.60
$\lambda = 256$	396942	0.61	> 1065618	17.31	0.88

TABLE 1. Comparison of cost using cost metric and time taken to run KuHash and hash functions in [12] and [23]. All results are averaged over 100 runs with random inputs. We remark that the cost of KuHash is the same for all runs because it is uniform.

For security levels $\lambda = 128, 192$ and 256, we obtain a speed-up of around 1.4 – 2.3x compared to the Castryck–Decru hash function, and around 27.0 – 28.3x compared to the Decru–Kunzweiler hash function. For a precise comparison between implementations, however, exact $\mathbb{F}_p$-operation counts of the Castryck–Decru and Decru–Kunzweiler hash functions are required. We note that an advantage of the algorithms developed with our approach is that we do not rely on in-built functionality and all our algorithms are uniform. Therefore, we are able to give precise $\mathbb{F}_p$-operation counts for KuHash.

References

[1] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and P. Schwabe. "Kummer strikes back: new DH speed records". In: *ASIACRYPT 2014*. Springer. 2014, pp. 317–337.

[2] D. J. Bernstein, L. De Feo, A. Leroux, and B. Smith. "Faster computation of isogenies of large prime degree". In: *Open Book Series* 4.1 (2020), pp. 39–55.

[3] Daniel J Bernstein. *Differential addition chains*. 2006. URL: http://cr.yp.to/ecdh/diffchain-20060219.pdf.

[4] G. Bisson. "Endomorphism rings in cryptography". PhD thesis. Institut National Polytechnique de Lorraine-INPL; Technische Universiteit Eindhoven, 2011.

[5] G. Bisson, R. Cosset, and D. Robert. AVIsogenies – a library for computing isogenies between abelian varieties. URL: http://avisogenies.gforge.inria.fr. Nov. 2012.

[6] J. W. Bos, C. Costello, H. Hisil, and K. Lauter. "Fast cryptography in genus 2". In: *Journal of Cryptology* 29 (2016), pp. 28–60.

[7] W. Bosma, J. Cannon, and C. Playoust. "The Magma algebra system. I. The user language". In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory, pp. 235–265.

[8] J.-B. Bost and J.-F. Mestre. "Moyenne arithmético-géométrique et périodes des courbes de genre 1 et 2". In: *Gaz. Math.* 38 (1988), pp. 36–64.

[9] N. Bruin, E. V. Flynn, and D. Testa. "Descent via $(3, 3)$-isogeny on Jacobians of genus 2 curves". In: *arXiv preprint arXiv:1401.0580* (2014).

[10] D. G. Cantor. "Computing in the Jacobian of a hyperelliptic curve". In: *Mathematics of computation* 48.177 (1987), pp. 95–101.

[11] J. W. S. Cassels and E. V. Flynn. *Prolegomena to a middlebrow arithmetic of curves of genus 2*. Vol. 230. Cambridge University Press, 1996.

[12] W. Castryck and T. Decru. "Multiradical isogenies". In: *Arithmetic, Geometry, Cryptography, and Coding Theory* 779 (2021), pp. 57–89.

[13] W. Castryck, T. Decru, and B. Smith. "Hash functions from superspecial genus-2 curves using Richelot isogenies". In: *Journal of Mathematical Cryptology* 14.1 (2020), pp. 268–292.

[14] D. X. Charles, K. E. Lauter, and E. Z. Goren. "Cryptographic Hash Functions from Expander Graphs". In: *J. Cryptol.* 22.1 (2009), pp. 93–113.

[15] D. V. Chudnovsky and G. V. Chudnovsky. "Sequences of numbers generated by addition in formal groups and new primality and factorization tests". In: *Advances in Applied Mathematics* 7.4 (1986), pp. 385–434.

[16] R. Cosset. "Applications des fonctions thêta à la cryptographie sur courbes hyperelliptiques". PhD thesis. Université Henri Poincaré - Nancy, 2011. URL: https://theses.hal.science/tel-00642951.

[17] R. Cosset. "Factorization with genus 2 curves". In: *Mathematics of Computation* 79.270 (2010), pp. 1191–1208.

[18] R. Cosset and D. Robert. "An algorithm for computing (l, l)-isogenies in polynomial time on Jacobians of hyperelliptic curves of genus 2". In: *in Mathematics of computation* (2013), p. 2.

[19] C. Costello and H. Hisil. "A simple and compact algorithm for SIDH with arbitrary degree isogenies". In: *ASIACRYPT 2017*. Springer. 2017, pp. 303–329.

[20] C. Costello and B. Smith. "The Supersingular Isogeny Problem in Genus 2 and Beyond". In: *PQCrypto*. Vol. 12100. Lecture Notes in Computer Science. Springer, 2020, pp. 151–168.
[21] J. Couveignes and T. Ezome. "Computing functions on Jacobians and their quotients". In: *LMS Journal of Computation and Mathematics* 18.1 (2015), pp. 555–577.
[22] P. Dartois, L. Maino, G. Pope, and D. Robert. *An Algorithmic Approach to $(2,2)$-isogenies in the Theta Model and Applications to Isogeny-based Cryptography*. Cryptology ePrint Archive, Paper 2023/1747. 2023. URL: https://eprint.iacr.org/2023/1747.
[23] T. Decru and S. Kunzweiler. "Efficient Computation of $(3^n, 3^n)$-Isogenies". In: *AFRICACRYPT 2023*. Vol. 14064. Lecture Notes in Computer Science. Springer, 2023, pp. 53–78.
[24] I. Dolgachev and D. Lehavi. "On isogenous principally polarized abelian surfaces". In: *Curves and abelian varieties* 465 (2008), pp. 51–69.
[25] L. De Feo, D. Jao, and J. Plût. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *Journal of Mathematical Cryptology* 8.3 (2014), pp. 209–247.
[26] E. V. Flynn. "Curves of genus 2". PhD thesis. University of Cambridge, 1989.
[27] E. V. Flynn. "Descent via $(5,5)$-isogeny on Jacobians of genus 2 curves". In: *Journal of Number Theory* 153 (2015), pp. 270–282.
[28] E. V. Flynn. "The Jacobian and Formal Group of a Curve of Genus 2 over an Arbitrary Ground Field". In: *Math. Proc. Cambridge Philos. Soc.* 107.3 (1990), pp. 425–441. DOI: 10.1017/S0305004100068729.
[29] E. V. Flynn and Y. B. Ti. "Genus Two Isogeny Cryptography". In: *PQCrypto*. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 286–306.
[30] P. Gaudry. "Fast genus 2 arithmetic based on Theta functions". In: *J. Math. Cryptol.* 1.3 (2007), pp. 243–265.
[31] D. Grant. "Formal groups in genus two." In: (1990).
[32] A. Hutchinson and K. Karabina. "Constructing multidimensional differential addition chains and their applications". In: *J. Cryptogr. Eng.* 9.1 (2019), pp. 1–19.
[33] H. W. Lenstra Jr. "Factoring integers with elliptic curves". In: *Annals of mathematics* (1987), pp. 649–673.
[34] D. R. Kohel. *Endomorphism rings of elliptic curves over finite fields*. University of California, Berkeley, 1996.
[35] S. Kunzweiler, Y. B. Ti, and C. Weitkämper. "Secret Keys in Genus-2 SIDH". In: *SAC 2021*. Vol. 13203. Lecture Notes in Computer Science. Springer, 2021, pp. 483–507.
[36] D. Lubicz and D. Robert. "Arithmetic on abelian and Kummer varieties". In: *Finite Fields Their Appl.* 39 (2016), pp. 130–158.
[37] D. Lubicz and D. Robert. "Fast change of level and applications to isogenies". In: *ANTS-XV*. 2022.
[38] D. Moody and D. Shumow. "Analogues of Vélu's formulas for isogenies on alternate models of elliptic curves". In: *Mathematics of Computation* 85.300 (2016), pp. 1929–1951.
[39] D. Mumford. "Tata lectures on theta II". In: *Progress in mathematics* 43 (1984), pp. 243–265.

[40] C. Nicholls. “Descent methods and torsion on Jacobians of higher genus curves”. PhD thesis. University of Oxford, 2018.
[41] R. Ohashi. “On the Rosenhain forms of superspecial curves of genus two”. In: *arXiv preprint arXiv:2308.11963* (2023).
[42] S. R. S. Rao. “Three Dimensional Montgomery Ladder, Differential Point Tripling on Montgomery Curves and Point Quintupling on Weierstrass and Edwards Curves”. In: *AFRICACRYPT 2016*. Vol. 9646. Lecture Notes in Computer Science. Springer, 2016, pp. 84–106.
[43] J. Renes. “Computing isogenies between Montgomery curves using the action of $(0, 0)$”. In: *PQCrypto*. Springer. 2018, pp. 229–247.
[44] J. Renes, P. Schwabe, B. Smith, and L. Batina. “μKummer: efficient hyperelliptic signatures and key exchange on microcontrollers”. In: *CHES*. Springer. 2016, pp. 301–320.
[45] J. Renes and B. Smith. “qDSA: Small and Secure Digital Signatures with Curve-Based Diffie-Hellman Key Pairs”. In: *ASIACRYPT 2017*. 2017, pp. 273–302.
[46] F. J. Richelot. “De transformatione integralium Abelianorum primi ordinis commentatio. Caput secundum. De computatione integralium Abelianorum primi ordinis.” In: (1837).
[47] D. Robert. “Efficient algorithms for abelian varieties and their moduli spaces”. PhD thesis. Université de Bordeaux (UB), 2021. URL: https://hal.science/tel-03498268.
[48] D. Robert. “Theta functions and cryptographic applications”. PhD thesis. Université Henri Poincaré - Nancy, 2010.
[49] B. Smith. “Computing low-degree isogenies in genus 2 with the Dolgachev-Lehavi method”. In: *Arithmetic, geometry, cryptography and coding theory* 574 (2012), pp. 159–170.
[50] Benjamin Smith. “Explicit endomorphisms and correspondences”. PhD thesis. The University of Sydney, 2006.
[51] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.2)*. `https://www.sagemath.org`. 2021.
[52] J. Vélu. “Isogénies entre courbes elliptiques”. In: *Comptes-Rendus de l’Académie des Sciences* 273 (1971), pp. 238–241.

Appendix A. Explicit $(2,2)$-isogenies on fast Kummers

The 15 unique $(2,2)$-subgroups are given by

$$
\begin{aligned}
G_{1,2} &:= \{(a\colon b\colon c\colon d),\ (a\colon b\colon -c\colon -d),\ (a\colon -b\colon c\colon -d),\ (a\colon -b\colon -c\colon d)\},\\
G_{1,4} &:= \{(a\colon b\colon c\colon d),\ (a\colon b\colon -c\colon -d),\ (b\colon a\colon d\colon c),\ (b\colon a\colon -d\colon -c)\},\\
G_{1,6} &:= \{(a\colon b\colon c\colon d),\ (a\colon b\colon -c\colon -d),\ (b\colon -a\colon d\colon -c),\ (b\colon -a\colon -d\colon c)\},\\
G_{2,8} &:= \{(a\colon b\colon c\colon d),\ (a\colon -b\colon c\colon -d),\ (c\colon d\colon a\colon b),\ (c\colon -d\colon a\colon -b)\},\\
G_{2,9} &:= \{(a\colon b\colon c\colon d),\ (a\colon -b\colon c\colon -d),\ (c\colon d\colon -a\colon -b),\ (c\colon -d\colon -a\colon b)\},\\
G_{3,12} &:= \{(a\colon b\colon c\colon d),\ (a\colon -b\colon -c\colon d),\ (d\colon c\colon b\colon a),\ (d\colon -c\colon -b\colon a)\},\\
G_{3,14} &:= \{(a\colon b\colon c\colon d),\ (a\colon -b\colon -c\colon d),\ (d\colon c\colon -b\colon -a),\ (d\colon -c\colon b\colon -a)\},\\
G_{4,8} &:= \{(a\colon b\colon c\colon d),(b\colon a\colon d\colon c),(c\colon d\colon a\colon b),(d\colon c\colon b\colon a)\},\\
G_{4,9} &:= \{(a\colon b\colon c\colon d),(b\colon a\colon d\colon c),(c\colon d\colon -a\colon -b),(d\colon c\colon -b\colon -a)\},\\
G_{5,10} &:= \{(a\colon b\colon c\colon d),(b\colon a\colon -d\colon -c),(c\colon -d\colon a\colon -b),(d\colon -c\colon -b\colon a)\},\\
G_{5,11} &:= \{(a\colon b\colon c\colon d),(b\colon a\colon -d\colon -c),(c\colon -d\colon -a\colon b),(d\colon -c\colon b\colon -a)\},\\
G_{6,8} &:= \{(a\colon b\colon c\colon d),(b\colon -a\colon d\colon -c),(c\colon d\colon a\colon b),(d\colon -c\colon b\colon -a)\},\\
G_{6,9} &:= \{(a\colon b\colon c\colon d),(b\colon -a\colon d\colon -c),(c\colon d\colon -a\colon -b),(d\colon -c\colon -b\colon a)\},\\
G_{7,10} &:= \{(a\colon b\colon c\colon d),(b\colon -a\colon -d\colon c),(c\colon -d\colon a\colon -b),(d\colon c\colon -b\colon -a)\},\\
G_{7,11} &:= \{(a\colon b\colon c\colon d),(b\colon -a\colon -d\colon c),(c\colon -d\colon -a\colon b),(d\colon c\colon b\colon a)\}.
\end{aligned}
$$

For each unique $(2,2)$-subgroup, we can associate morphism $\alpha : \mathcal{K} \to \mathcal{K}$ induced by a linear map on $\mathbb{A}^4$ defined by a matrix $\mathbf{A} = (a_{i,j})_{1\le i,j\le 4}$ with $a_{i,j}^2 \in \{0,1\}$, such that the corresponding $(2,2)$-isogeny $\mathcal{K} \to \widetilde{\mathcal{K}}$ is given by

$$\psi_{i,j}(P) := \mathcal{H} \circ \mathcal{S} \circ \alpha(P).$$

Let i be a root of x^2+1 in $\overline{\mathbb{k}}[x]$. The matrices $\mathbf{A}$ for each $(2,2)$-subgroup $G_{i,j}$ above are given by:

$$
G_{1,2} : \mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad
G_{1,4} : \mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}
$$

$$
G_{1,6} : \mathbf{A} = \begin{pmatrix} 1 & i & 0 & 0 \\ 1 & -i & 0 & 0 \\ 0 & 0 & 1 & i \\ 0 & 0 & 1 & -i \end{pmatrix}, \qquad
G_{2,8} : \mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{pmatrix}
$$

$$
G_{2,9} : \mathbf{A} = \begin{pmatrix} 1 & 0 & i & 0 \\ 1 & 0 & -i & 0 \\ 0 & 1 & 0 & i \\ 0 & 1 & 0 & -i \end{pmatrix}, \qquad
G_{3,12} : \mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{pmatrix}
$$

$$
G_{3,14} : \mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & i \\ 1 & 0 & 0 & -i \\ 0 & 1 & i & 0 \\ 0 & 1 & -i & 0 \end{pmatrix}, \qquad
G_{4,8} : \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}
$$

$$G_{4,9} : \mathbf{A} = \begin{pmatrix} 1 & 1 & i & i \\ 1 & 1 & -i & -i \\ 1 & -1 & i & -i \\ 1 & -1 & -i & i \end{pmatrix}, \qquad G_{5,10} : \mathbf{A} = \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

$$G_{5,11} : \mathbf{A} = \begin{pmatrix} 1 & -1 & -i & -i \\ 1 & -1 & i & i \\ 1 & 1 & -i & i \\ 1 & 1 & i & -i \end{pmatrix}, \qquad G_{6,8} : \mathbf{A} = \begin{pmatrix} 1 & i & 1 & i \\ 1 & i & -1 & -i \\ 1 & -i & 1 & -i \\ 1 & -i & -1 & i \end{pmatrix}$$

$$G_{6,9} : \mathbf{A} = \begin{pmatrix} 1 & -i & -i & -1 \\ 1 & -i & i & 1 \\ 1 & i & -i & 1 \\ 1 & i & i & -1 \end{pmatrix}, \qquad G_{7,10} : \mathbf{A} = \begin{pmatrix} 1 & -i & -1 & -i \\ 1 & -i & 1 & i \\ 1 & i & -1 & i \\ 1 & i & 1 & -i \end{pmatrix}$$

$$G_{7,11} : \mathbf{A} = \begin{pmatrix} 1 & i & i & 1 \\ 1 & i & -i & -1 \\ 1 & -i & i & -1 \\ 1 & -i & -i & 1 \end{pmatrix}$$

Appendix B. Three-dimensional Addition Chain

We describe the three-dimensional addition chain 3DAC needed to compute kernel generators of the form $P_1 + [\beta]P_2 + [\gamma]P_3$, for scalars $\beta, \gamma \in \mathbb{Z}/3^k\mathbb{Z}$ and points $P_1, P_2, P_3 \in \mathcal{K}$. We obtain our three-dimensional differential addition chain – 3DAC – by extending the two-dimensional differential addition chain due to Bernstein [3], the fastest known chain of this kind that is both fixed length and uniform.

The three main subroutines used in 3DAC are DBLTHRICEADD, an encoding algorithm ENCODE and an indexing algorithm IND.

The algorithm DBLTHRICEADD is defined to compute the mapping

$$\big(P, Q, P-Q, R, S, R-S, T, U, T-U\big) \mapsto \big([2]P, P+Q, R+S, T+U\big),$$

for points P, Q, R, S, T, and U in $\mathcal{K}$, using one pseudo-doubling and three pseudo-additions on the Kummer surface $\mathcal{K}$.

The encoding algorithm ENCODE takes as input two ℓ-bit scalars $\beta, \gamma \in \mathbb{Z}3^k\mathbb{Z}$ and outputs a single bit b and four $(\ell-1)$-bit scalars b_i for $i = 0, 1, 2, 3$. The bit b determines one of two possible input combinations that is fed into a differential addition that kickstarts the chain (see Step 5 of Algorithm 6), after which the j-th bits (for $j = 1 \ldots \ell$) of each of the four b_i determine one of 16 input permutations that is fed into the DBLTHRICEADD routine (see Step 10 of Algorithm 6).

The indexing algorithm IND is used to choose one of four points as the last input to the DBLTHRICEADD algorithm.

The full algorithm 3DAC is specified below. 3DAC computes $P_1 + [\beta]P_2 + [\gamma]P_3$, given scalars $\beta, \gamma \in \mathbb{Z}/3^k\mathbb{Z}$, the length ℓ of the chain, the fundamental theta constants $O_{\mathcal{K}}$ of fast Kummer surface $\mathcal{K}$ that we are working on, and a tuple of points

$$\mathcal{D} := \big(P_1, P_2, P_3, P_2+P_3, P_2-P_3, P_1-P_2, P_1-P_2, [2](P_2+P_3), P_1+P_2+P_3, P_1-P_2-P_3\big)$$

on $\mathcal{K}$.

Algorithm 4 $\mathsf{ENCODE}(\beta, \gamma)$:

Input: two ℓ-bit scalars $\beta = (\beta[\ell-1], \ldots, \beta[0])$ and $\gamma = (\gamma[\ell-1], \ldots, \gamma[0])$
Output: a bit $\mathsf{b} \in \{0,1\}$, and four $(\ell-1)$-bit scalars (b_0, b_1, b_2, b_3)

$\mathsf{b} \leftarrow \beta[1]$
for $i = 1$ to $\ell - 1$ **do**
  $b_1[i] \leftarrow \beta[i] \oplus \beta[i+1]$
  $b_0[i] \leftarrow b_1[i] \oplus \gamma[i] \oplus \gamma[i+1]$
  $b_2[i] \leftarrow \beta[i+1] \oplus \gamma[i+1]$
  $b_3[i] \leftarrow \mathsf{b}$
  $\mathsf{b} \leftarrow b_1[i] \oplus (b_0[i] \oplus 1) \otimes \mathsf{b}$
return b, (b_0, b_1, b_2, b_3)

Algorithm 5 $\mathsf{IND}(I)$:

Input: a 6-tuple of integers $I = (I_1, \ldots, I_6)$
Output: an integer index $\mathsf{ind} \in \{1, 2, 3, 4\}$

switch $([I_3 - I_1, I_4 - I_2])$
case $[-1, -1]$**:** $\mathsf{ind} \leftarrow 1$
case $[\ 1,\ \ 1]$**:** $\mathsf{ind} \leftarrow 2$
case $[\ 1, -1]$**:** $\mathsf{ind} \leftarrow 3$
case $[-1,\ \ 1]$**:** $\mathsf{ind} \leftarrow 4$
end switch
return ind

University College London, UK
Email address: maria.santos.20@ucl.ac.uk

Microsoft Research, USA
Email address: craigco@microsoft.com

Inria and École polytechnique, Institut Polytechnique de Paris, Palaiseau, France
Email address: smith@lix.polytechnique.fr

Algorithm 6 $\mathsf{3DAC}(\mathcal{D}, \beta, \gamma, \ell, O_{\mathcal{K}})$:

Input: Scalars $\beta, \gamma \in \mathbb{Z}/3^k\mathbb{Z}$, the length of chain ℓ, fundamental theta constants $O_{\mathcal{K}}$, and a tuple $\mathcal{D}$ of points on $\mathcal{K}$.

Output: $P_1 + [\beta]P_2 + [\gamma]P_3$ where P_1, P_2, P_3 are $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$, respectively.

initialise $P \leftarrow (\mathcal{D}_4, \mathcal{D}_8, \mathcal{D}_4, \mathcal{D}_9)$
initialise $D \leftarrow (\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_5)$
initialise $\Delta \leftarrow (\mathcal{D}_1, \mathcal{D}_{10}, \mathcal{D}_6, \mathcal{D}_7)$
initialise $I \leftarrow (1, 1, 2, 2, 1, 1)$
$\mathsf{b}, (b_0, b_1, b_2, b_3) \leftarrow \mathsf{ENCODE}(\beta, \gamma)$
if $\mathsf{b} = 1$ **then**
    $(P_3, I_6) \leftarrow ((P_3, D_2, D_1), I_6 + 1)$
else
    $(P_3, I_5) \leftarrow ((P_3, D_1, D_2), I_5 + 1)$
switch $((b_0, b_1, b_2, b_3))$
case $(0,0,0,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_3, 2I_4, I_3 + I_5, I_4 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_2, P_1, D_3, P_3, P_2, D_2, P_2, P_4, \Delta_{\mathsf{IND}(I)})$
case $(0,0,0,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_3, 2I_4, I_3 + I_5, I_4 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_2, P_1, D_3, P_3, P_2, D_1, P_2, P_4, \Delta_{\mathsf{IND}(I)})$
case $(0,0,1,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_3, 2I_4, I_3 + I_5, I_4 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_2, P_1, D_4, P_3, P_2, D_2, P_2, P_4, \Delta_{\mathsf{IND}(I)})$
case $(0,0,1,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_3, 2I_4, I_3 + I_5, I_4 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_2, P_1, D_4, P_3, P_2, D_1, P_2, P_4, \Delta_{\mathsf{IND}(I)})$
case $(0,1,0,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_1, 2I_2, I_1 + I_5, I_2 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_1, P_2, D_3, P_3, P_1, D_2, P_1, P_4, \Delta_{\mathsf{IND}(I)})$
case $(0,1,0,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_1, 2I_2, I_1 + I_5, I_2 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_1, P_2, D_3, P_3, P_1, D_1, P_1, P_4, \Delta_{\mathsf{IND}(I)})$
case $(0,1,1,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_1, 2I_2, I_1 + I_5, I_2 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_1, P_2, D_4, P_3, P_1, D_2, P_1, P_4, \Delta_{\mathsf{IND}(I)})$
case $(0,1,1,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_1, 2I_2, I_1 + I_5, I_2 + I_6)$
$(P_2, P_1, P_3, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_1, P_2, D_4, P_3, P_1, D_1, P_1, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,0,0,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_3 + I_5, I_4 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_2, D_2, P_1, P_2, D_3, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,0,0,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_1 + I_5, I_2 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_1, D_1, P_1, P_2, D_3, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,0,1,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_3 + I_5, I_4 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_2, D_2, P_1, P_2, D_4, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,0,1,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_1 + I_5, I_2 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_1, D_1, P_1, P_2, D_4, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,1,0,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_1 + I_5, I_2 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_1, D_2, P_1, P_2, D_3, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,1,0,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_3 + I_5, I_4 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_2, D_1, P_1, P_2, D_3, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,1,1,0)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_1 + I_5, I_2 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_1, D_2, P_1, P_2, D_4, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
case $(1,1,1,1)$**:** $I \leftarrow (I_1 + I_3, I_2 + I_4, 2I_5, 2I_6, I_3 + I_5, I_4 + I_6)$
$(P_2, P_3, P_1, P_4) \leftarrow \mathsf{DBLTHRICEADD}(P_3, P_2, D_1, P_1, P_2, D_4, P_3, P_4, \Delta_{\mathsf{IND}(I)})$
end switch
return P_4