Chasing Rabbits Through Hypercubes: Better algorithms for higher dimensional 2-isogeny computations

Pierrick Dartois and Max Duparc

2026, July 9

HD isogenies in cryptography

- Since 2022, an overwhelming majority of new isogeny based schemes use 2D isogenies.
- Some of them require 4D isogenies (SQIsignHD, qt-Pegasis, MIKE).
- We need fast & constant time formulae to compute these isogenies.

HD isogenies in cryptography

- Since 2022, an overwhelming majority of new isogeny based schemes use 2D isogenies.
- Some of them require 4D isogenies (SQIsignHD, qt-Pegasis, MIKE).
- We need fast & constant time formulae to compute these isogenies.

✓ Possible using theta coordinates [Mum66].

Level 2 theta coordinates for isogeny computations

Symplectic basis

Setup:

- Let k be a field.
- Let A/k be a PPAV of dimension g.

Symplectic basis

Setup:

- Let k be a field.
- Let A/k be a PPAV of dimension g.

Symplectic basis:

- Let $n \in \mathbb{Z}_{>0}$.
- Let $\zeta \in \overline{k}^*$ be a primitive n-th root of unity.
- Let $e_n : A[n] \times A[n] \to \overline{k}^*$ be the n-th Weil pairing.

Symplectic basis

Setup:

- Let k be a field.
- Let A/k be a PPAV of dimension g.

Symplectic basis:

- Let $n \in \mathbb{Z}_{>0}$.
- Let $\zeta \in \overline{k}^*$ be a primitive n-th root of unity.
- Let $e_n : A[n] \times A[n] \to \overline{k}^*$ be the n-th Weil pairing.
- A (ζ-)symplectic basis $\mathscr{B} := (S_1, \cdots, S_g, T_1, \cdots, T_g)$ of $A[n] \subseteq A(\overline{k})$ satisfies:

$$\forall l, m \in [\![1 \ ; \ g]\!], \quad e_n(S_l, S_m) = e_n(T_l, T_m) = 1,$$
$$\text{and} \quad e_n(S_l, T_m) = \zeta^{\delta_{l,m}},$$

where $\delta_{l,m} = 1$ if $l = m$ and 0 otherwise.

Definition: level 2 theta structures

Definition

Let A/k be a PPAV of dimension g. A *theta structure* of level 2 is a map

$$\begin{array}{rcl} \theta^A : A & \longrightarrow & \mathbb{P}^{2^g-1} \\ P & \longmapsto & \theta^A(P) = (\theta^A_{\mathsf{i}}(P))_{\mathsf{i}\in(\mathbb{Z}/2\mathbb{Z})^g} \end{array}$$

along with a symplectic basis $\mathscr{B} := (S_1, \cdots, S_g, T_1, \cdots, T_g)$ of $A[2]$ satisfying the *theta group action relation*:

$$\theta^A_{\mathsf{i}}(P + S_m) = \theta^A_{\mathsf{i}+\mathsf{e}_m}(P) \quad \text{and} \quad \theta^A_{\mathsf{i}}(P + T_m) = (-1)^{i_m}\theta^A_{\mathsf{i}}(P),$$

for all $P \in A$, $\mathsf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$ and $m \in [\![1\ ;\ g]\!]$, where $\mathsf{e}_m := (\delta_{l,m})_{1\le l\le g}$.

Properties of level 2 theta structures

Theorem (Mumford, 1966)

A level 2 theta structure $\Theta_A := (\theta^A, \mathscr{B})$ is fully determined by a symplectic basis $\mathscr{B}'$ of $A[4]$ such that $[2]\mathscr{B}' = \mathscr{B}$.

Properties of level 2 theta structures

Theorem (Mumford, 1966)

A level 2 theta structure $\Theta_A := (\theta^A, \mathscr{B})$ is fully determined by a symplectic basis $\mathscr{B}'$ of $A[4]$ such that $[2]\mathscr{B}' = \mathscr{B}$.

Theorem

If A is not a (polarised) product, a level 2 theta structure induces an embedding of the Kummer variety:

$$\theta^A : A/\pm \hookrightarrow \mathbb{P}^{2^g-1}$$

- Points are represented up to sign: $\theta^A(P) = \theta^A(-P)$ for all $P \in A$.
- This is the price to pay for minimising the number of coordinates.

Properties of level 2 theta structures

Theorem (Mumford, 1966)

A level 2 theta structure $\Theta_A := (\theta^A, \mathscr{B})$ is fully determined by a symplectic basis $\mathscr{B}'$ of $A[4]$ such that $[2]\mathscr{B}' = \mathscr{B}$.

Theorem

If A is not a (polarised) product, a level 2 theta structure induces an embedding of the Kummer variety:

$$\theta^A : A/\pm \hookrightarrow \mathbb{P}^{2g-1}$$

- Points are represented up to sign: $\theta^A(P) = \theta^A(-P)$ for all $P \in A$.
- This is the price to pay for minimising the number of coordinates.
- We have a differential addition and doubling formulae

$$(\theta^A(P), \theta^A(Q), \theta^A(P-Q)) \mapsto \theta^A(P+Q) \qquad \theta^A(P) \mapsto \theta^A([2]P).$$

Motivation: computing chains of 2-isogenies

Our working example: qt-Pegasis [DEIV25]

Inputs:

- Let $E/\mathbb{F}_p$ be a supersingular elliptic curve (p prime $\equiv 7 \mod 8$).
- Assume E is oriented by $\mathbb{Z}[(\sqrt{-p}+1)/2] \hookrightarrow \mathrm{End}(E)$ where $\sqrt{-p}$ is mapped to $\pi_p : (x, y) \in E \mapsto (x^p, y^p) \in E$.
- Let $\mathfrak{a} \subseteq \mathbb{Z}[(\sqrt{-p}+1)/2]$ be an ideal.

Our working example: qt-Pegasis [DEIV25]

Inputs:

- Let $E/\mathbb{F}_p$ be a supersingular elliptic curve (p prime $\equiv 7 \mod 8$).
- Assume E is oriented by $\mathbb{Z}[(\sqrt{-p}+1)/2] \hookrightarrow \mathrm{End}(E)$ where $\sqrt{-p}$ is mapped to $\pi_p : (x, y) \in E \mapsto (x^p, y^p) \in E$.
- Let $\mathfrak{a} \subseteq \mathbb{Z}[(\sqrt{-p}+1)/2]$ be an ideal.

Output: Compute $E_\mathfrak{a}$, the result of the action by $\mathfrak{a}$: $E \to E_\mathfrak{a}$.

Our working example: qt-Pegasis [DEIV25]

Inputs:

- Let $E/\mathbb{F}_p$ be a supersingular elliptic curve (p prime $\equiv 7 \mod 8$).
- Assume E is oriented by $\mathbb{Z}[(\sqrt{-p}+1)/2] \hookrightarrow \mathrm{End}(E)$ where $\sqrt{-p}$ is mapped to $\pi_p : (x, y) \in E \mapsto (x^p, y^p) \in E$.
- Let $\mathfrak{a} \subseteq \mathbb{Z}[(\sqrt{-p}+1)/2]$ be an ideal.

Output: Compute $E_\mathfrak{a}$, the result of the action by $\mathfrak{a}$: $E \to E_\mathfrak{a}$.

Solution (qt-Pegasis [DEIV25]):

- Solve a quadratic diophantine equation $\sum_{i=1}^{4} N(\mathfrak{b}_i) = 2^e$.

Our working example: qt-Pegasis [DEIV25]

Inputs:

- Let $E/\mathbb{F}_p$ be a supersingular elliptic curve (p prime $\equiv 7 \mod 8$).
- Assume E is oriented by $\mathbb{Z}[(\sqrt{-p}+1)/2] \hookrightarrow \mathrm{End}(E)$ where $\sqrt{-p}$ is mapped to $\pi_p : (x, y) \in E \mapsto (x^p, y^p) \in E$.
- Let $\mathfrak{a} \subseteq \mathbb{Z}[(\sqrt{-p}+1)/2]$ be an ideal.

Output: Compute $E_{\mathfrak{a}}$, the result of the action by $\mathfrak{a}$: $E \to E_{\mathfrak{a}}$.

Solution (qt-Pegasis [DEIV25]):

- Solve a quadratic diophantine equation $\sum_{i=1}^{4} N(\mathfrak{b}_i) = 2^e$.
- Compute a corresponding 2^e-isogeny in dimension $g = 4$:

$$F : E^4 \to E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S$$

A 2^e-isogeny satisfies $\widetilde{F} \circ F = [2^e]$, where $\widetilde{F} : E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S \to E^4$ is the *contravariant* of F.

Our working example: qt-Pegasis [DEIV25]

Inputs:

- Let $E/\mathbb{F}_p$ be a supersingular elliptic curve (p prime $\equiv 7 \mod 8$).
- Assume E is oriented by $\mathbb{Z}[(\sqrt{-p}+1)/2] \hookrightarrow \mathrm{End}(E)$ where $\sqrt{-p}$ is mapped to $\pi_p : (x, y) \in E \mapsto (x^p, y^p) \in E$.
- Let $\mathfrak{a} \subseteq \mathbb{Z}[(\sqrt{-p}+1)/2]$ be an ideal.

Output: Compute $E_\mathfrak{a}$, the result of the action by $\mathfrak{a}$: $E \to E_\mathfrak{a}$.

Solution (qt-Pegasis [DEIV25]):

- Solve a quadratic diophantine equation $\sum_{i=1}^{4} N(\mathfrak{b}_i) = 2^e$.
- Compute a corresponding 2^e-isogeny in dimension $g = 4$:

$$F : E^4 \to E_\mathfrak{a} \times E_{\overline{\mathfrak{a}}} \times S$$

  A 2^e-isogeny satisfies $\widetilde{F} \circ F = [2^e]$, where $\widetilde{F} : E_\mathfrak{a} \times E_{\overline{\mathfrak{a}}} \times S \to E^4$ is the *contravariant* of F.
- Extract $E_\mathfrak{a}$ from the codomain of F.

Computing a 2^e-isogeny: example of qt-Pegasis

Goal: Compute a 2^e-isogeny $F : A \longrightarrow B$ when given $T_1, \cdots, T_g \in A[2^{e+2}]$ such that $\ker(F) = \langle [4]T_1, \cdots, [4]T_g \rangle$. The computation is done in level 2 theta coordinates:

$$\theta^A(P) \longmapsto \theta^B(F(P))$$

Computing a 2^e-isogeny: example of qt-Pegasis

Goal: Compute a 2^e-isogeny $F : A \longrightarrow B$ when given $T_1, \cdots, T_g \in A[2^{e+2}]$ such that $\ker(F) = \langle [4]T_1, \cdots, [4]T_g \rangle$. The computation is done in level 2 theta coordinates:

$$\theta^A(P) \longmapsto \theta^B(F(P))$$

Method (ex. qt-Pegasis, $g = 4$): Decompose F into a chain of 2-isogenies of length e:

$$A = E^4 \xrightarrow{f_1} A_1 \xrightarrow{f_2} A_2 \quad \cdots \quad A_{e-2} \xrightarrow{f_{e-1}} A_{e-1} \xrightarrow{f_e} B = E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S$$

Gluing — Generic isogenies — Splitting

Computing a 2^e-isogeny: example of qt-Pegasis

$$A = E^4 \xrightarrow{f_1} A_1 \xrightarrow{f_2} A_2 \quad \cdots \quad A_{e-2} \xrightarrow{f_{e-1}} A_{e-1} \xrightarrow{f_e} B = E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S$$

Gluing — Generic isogenies — Splitting

Steps:

1. **Gluing:** Compute the gluing $f_1 : A \to A_1$.
   Using 8-torsion points $[2^{e-1}]T_1, \cdots, [2^{e-1}]T_g$.

Computing a 2^e-isogeny: example of qt-Pegasis

$$A = E^4 \xrightarrow{f_1} A_1 \xrightarrow{f_2} A_2 \quad \cdots \quad A_{e-2} \xrightarrow{f_{e-1}} A_{e-1} \xrightarrow{f_e} B = E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S$$

Gluing — Generic isogenies — Splitting

Steps:

1. **Gluing:** Compute the gluing $f_1 : A \to A_1$.
   Using 8-torsion points $[2^{e-1}]T_1, \cdots, [2^{e-1}]T_g$.

2. **Generic:** Compute generic isogenies $f_i : A_{i-1} \to A_i$ $(2 \le i \le e)$.
   Using 8-torsion points $[2^{e-i}]f_{i-1} \circ \cdots \circ f_1(T_1, \cdots, T_g)$.

Computing a 2^e-isogeny: example of qt-Pegasis

$$A = E^4 \xrightarrow{f_1} A_1 \xrightarrow{f_2} A_2 \quad \cdots \quad A_{e-2} \xrightarrow{f_{e-1}} A_{e-1} \xrightarrow{f_e} B = E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S$$

Gluing — Generic isogenies — Splitting

Steps:

1. **Gluing:** Compute the gluing $f_1 : A \to A_1$.
   Using 8-torsion points $[2^{e-1}]T_1, \cdots, [2^{e-1}]T_g$.

2. **Generic:** Compute generic isogenies $f_i : A_{i-1} \to A_i$ $(2 \leq i \leq e)$.
   Using 8-torsion points $[2^{e-i}]f_{i-1} \circ \cdots \circ f_1(T_1, \cdots, T_g)$.
   $O(g \cdot e \log(e))$ point duplications and 2-isogeny evaluations needed.

Computing a 2^e-isogeny: example of qt-Pegasis

$$A = E^4 \xrightarrow{f_1} A_1 \xrightarrow{f_2} A_2 \quad \cdots \quad A_{e-2} \xrightarrow{f_{e-1}} A_{e-1} \xrightarrow{f_e} B = E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S$$

Gluing — Generic isogenies — Splitting

Steps:

1. **Gluing:** Compute the gluing $f_1 : A \to A_1$.
   Using 8-torsion points $[2^{e-1}]T_1, \cdots, [2^{e-1}]T_g$.

2. **Generic:** Compute generic isogenies $f_i : A_{i-1} \to A_i$ $(2 \le i \le e)$.
   Using 8-torsion points $[2^{e-i}]f_{i-1} \circ \cdots \circ f_1(T_1, \cdots, T_g)$.
   $O(g \cdot e \log(e))$ point duplications and 2-isogeny evaluations needed.

3. **Splitting:** Split the codomain B (into $E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S$).

Computing generic 2-isogenies

Computing a generic 2-isogeny

Goal: compute a 2-isogeny $f : A \to B$.

Computing a generic 2-isogeny

Goal: compute a 2-isogeny $f : A \to B$.

Inputs:

- 8-torsion points $T_1, \cdots, T_g$ such that $\ker(f) = \langle [4]T_1, \cdots, [4]T_g \rangle$.

Computing a generic 2-isogeny

Goal: compute a 2-isogeny $f : A \to B$.

Inputs:

- 8-torsion points $T_1, \cdots, T_g$ such that $\ker(f) = \langle [4]T_1, \cdots, [4]T_g \rangle$.
- Consider a symplectic basis $\mathscr{B} := (S_1, \cdots, S_g, T_1, \cdots, T_g)$ of $A[8]$.
- A level 2-theta structure Θ_A on A **adapted to** f *i.e.* induced by the symplectic basis $[2]\mathscr{B}$ of $A[4]$.

Computing a generic 2-isogeny

Goal: compute a 2-isogeny $f : A \to B$.

Inputs:

- 8-torsion points $T_1, \cdots, T_g$ such that $\ker(f) = \langle [4]T_1, \cdots, [4]T_g \rangle$.
- Consider a symplectic basis $\mathscr{B} := (S_1, \cdots, S_g, T_1, \cdots, T_g)$ of $A[8]$.
- A level 2-theta structure Θ_A on A **adapted to** f *i.e.* induced by the symplectic basis $[2]\mathscr{B}$ of $A[4]$.

Output:

- A level 2-theta structure Θ_B on B induced by the symplectic basis of $B[4]$:

$$f_*(\mathscr{B}) := ([2]f(S_1), \cdots, [2]f(S_g), f(T_1), \cdots, f(T_g)).$$

- This provides a way to evaluate f easily $\theta^A(P) \mapsto \theta^B(f(P))$.

Computing a generic 2-isogeny

Computation:

- Compute the **theta null-point** $\theta^B(0_B)$ from $\theta^A(T_1), \cdots, \theta^A(T_g)$.
- $\theta^B(0_B)$ determines (B, Θ_B) and $f : A \to B$.

Computing a generic 2-isogeny

Computation:

- Compute the **theta null-point** $\theta^B(0_B)$ from $\theta^A(T_1),\cdots,\theta^A(T_g)$.
- $\theta^B(0_B)$ determines (B,Θ_B) and $f : A \to B$.

Evaluation: Use the **duplication formula**:

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P)^{\odot 2}),$$

with the notations:

- $x \odot y = (x_i \cdot y_i)_{i\in(\mathbb{Z}/2\mathbb{Z})^g}$ (dot product).
- $H(x) = \left(\sum_{j\in(\mathbb{Z}/2\mathbb{Z})^g}(-1)^{\langle i|j\rangle}x_j\right)_{i\in(\mathbb{Z}/2\mathbb{Z})^g}$ (Hadamard).
- $U^B(P) = H(\theta^B(P))$ (dual theta coordinates).

Computing a generic 2-isogeny

Computation:

- Compute the **theta null-point** $\theta^B(0_B)$ from $\theta^A(T_1),\cdots,\theta^A(T_g)$.
- $\theta^B(0_B)$ determines (B,Θ_B) and $f : A \to B$.

Evaluation: Use the **duplication formula**:

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P)^{\odot 2}),$$

with the notations:

- $x \odot y = (x_i \cdot y_i)_{i\in(\mathbb{Z}/2\mathbb{Z})^g}$ (dot product).
- $H(x) = \left(\sum_{j\in(\mathbb{Z}/2\mathbb{Z})^g}(-1)^{\langle i|j\rangle}x_j\right)_{i\in(\mathbb{Z}/2\mathbb{Z})^g}$ (Hadamard).
- $U^B(P) = H(\theta^B(P))$ (dual theta coordinates).

Evaluation algorithm:

$$\theta^A(P) \xrightarrow{-^{\odot 2}} * \xrightarrow{H} * \xrightarrow{-\odot U^B(0_B)^{\odot -1}} * \xrightarrow{H} \theta^B(f(P)).$$

Inverse dual theta null-point computation

Goal: Compute $U^B(0_B)^{\odot -1}$.

Inverse dual theta null-point computation

Goal: Compute $U^B(0_B)^{\odot -1}$.

Method:

- For all $\mathrm{j} \in (\mathbb{Z}/2\mathbb{Z})^g$, let $T_{\mathrm{j}} := \sum_{m=1}^{g} [j_m]\, T_m$.

Inverse dual theta null-point computation

Goal: Compute $U^B(0_B)^{\odot -1}$.

Method:

- For all $\mathsf{j} \in (\mathbb{Z}/2\mathbb{Z})^g$, let $T_\mathsf{j} := \sum_{m=1}^{g} [j_m] T_m$.
- From $\theta^A(T_\mathsf{j})$, we know for all $\mathsf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$,

$$\pi_{\mathsf{i},\mathsf{j}} := H(\theta^A(T_\mathsf{j})^{\odot 2})_\mathsf{i} \quad \text{and} \quad \pi_{\mathsf{i}+\mathsf{j},\mathsf{j}} := H(\theta^A(T_\mathsf{j})^{\odot 2})_{\mathsf{i}+\mathsf{j}}.$$

Inverse dual theta null-point computation

Goal: Compute $U^B(0_B)^{\odot -1}$.

Method:

- For all $\mathsf{j} \in (\mathbb{Z}/2\mathbb{Z})^g$, let $T_\mathsf{j} := \sum_{m=1}^{g} [j_m] T_m$.
- From $\theta^A(T_\mathsf{j})$, we know for all $\mathsf{i} \in (\mathbb{Z}/2\mathbb{Z})^g$,

$$\pi_{\mathsf{i},\mathsf{j}} := H(\theta^A(T_\mathsf{j})^{\odot 2})_\mathsf{i} \quad \text{and} \quad \pi_{\mathsf{i}+\mathsf{j},\mathsf{j}} := H(\theta^A(T_\mathsf{j})^{\odot 2})_{\mathsf{i}+\mathsf{j}}.$$

- Propagation: from $U^B_\mathsf{i}(0_B)^{-1}$, we get

$$U^B_{\mathsf{i}+\mathsf{j}}(0_B)^{-1} = \frac{\pi_{\mathsf{i},\mathsf{j}}}{\pi_{\mathsf{i}+\mathsf{j},\mathsf{j}}} \cdot U^B_\mathsf{i}(0_B)^{-1} \quad \text{(by duplication formula)}$$

whenever $\pi_{\mathsf{i}+\mathsf{j},\mathsf{j}} \neq 0 \iff \pi_{\mathsf{i},\mathsf{j}} \neq 0$.

The hypercube inverse interpolation problem (HIIP)

Goal: Compute the projective inverse of $x := U^B(0_B)$.

Inputs:

- **Support** $S \subseteq (\mathbb{Z}/2\mathbb{Z})^g$ of indices of known $\theta^A(T_j)$.
  $S = \{e_1, \cdots, e_g\}$ when $\theta^A(T_1), \cdots, \theta^A(T_g)$ are known.
- Submatrix $(\pi_{i,j})_{i \in (\mathbb{Z}/2\mathbb{Z})^g, j \in S}$.

The hypercube inverse interpolation problem (HIIP)

Goal: Compute the projective inverse of $x := U^B(0_B)$.

Inputs:

- **Support** $S \subseteq (\mathbb{Z}/2\mathbb{Z})^g$ of indices of known $\theta^A(T_j)$.
  $S = \{e_1, \cdots, e_g\}$ when $\theta^A(T_1), \cdots, \theta^A(T_g)$ are known.
- Submatrix $(\pi_{i,j})_{i \in (\mathbb{Z}/2\mathbb{Z})^g, j \in S}$.

Problem (Hypercube inverse interpolation problem - HIIP)

Find $x^{\odot -1} \in \mathbb{P}^{2^g-1}(k)$ *such that:*

$$\forall i \in (\mathbb{Z}/2\mathbb{Z})^g, j \in S, \quad x_i \cdot \pi_{i+j,j} = x_{i+j} \cdot \pi_{i,j}.$$

The hypercube inverse interpolation problem (HIIP)

Goal: Compute the projective inverse of $x := U^B(0_B)$.

Inputs:

- **Support** $S \subseteq (\mathbb{Z}/2\mathbb{Z})^g$ of indices of known $\theta^A(T_j)$.
  $S = \{e_1, \cdots, e_g\}$ when $\theta^A(T_1), \cdots, \theta^A(T_g)$ are known.
- Submatrix $(\pi_{i,j})_{i \in (\mathbb{Z}/2\mathbb{Z})^g, j \in S}$.

Problem (Hypercube inverse interpolation problem - HIIP)

Find $x^{\odot -1} \in \mathbb{P}^{2^g-1}(k)$ *such that:*

$$\forall i \in (\mathbb{Z}/2\mathbb{Z})^g, j \in S, \quad x_i \cdot \pi_{i+j,j} = x_{i+j} \cdot \pi_{i,j}.$$

⚠ Inversions are very costly so we solve the HIIP projectively with multiplications only.

The HIIP: a graph theoretic problem

The graph: $G = (V, E)$.

- Vertices $V := (\mathbb{Z}/2\mathbb{Z})^g$.
- Edges $E := \{(i + j, i) \mid j \in S \quad \text{and} \quad \pi_{i,j} \neq 0\}$.

The HIIP: a graph theoretic problem

The graph: $G = (V, E)$.

- Vertices $V := (\mathbb{Z}/2\mathbb{Z})^g$.
- Edges $E := \{(i + j, i) \mid j \in S \quad \text{and} \quad \pi_{i,j} \neq 0\}$.
- When $S = \{e_1, \cdots, e_g\}$, G is a subgraph of the hypercube.

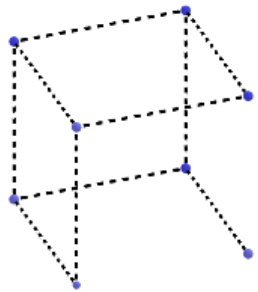


Figure: Subgraph of the (hyper)cube for $g = 3$ (with $\pi_{0,100} = \pi_{100,101} = 0$)

The HIIP: a graph theoretic problem

The graph: $G = (V, E)$.

- Vertices $V := (\mathbb{Z}/2\mathbb{Z})^g$.
- Edges $E := \{(i + j, i) \mid j \in S \quad \text{and} \quad \pi_{i,j} \neq 0\}$.
- When $S = \{e_1, \cdots, e_g\}$, G is a subgraph of the hypercube.

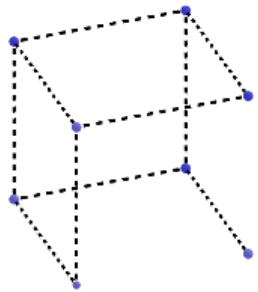


Figure: Subgraph of the (hyper)cube for $g = 3$ (with $\pi_{0,100} = \pi_{100,101} = 0$)

Theorem

The solution to the HIIP is unique iff G is connected.

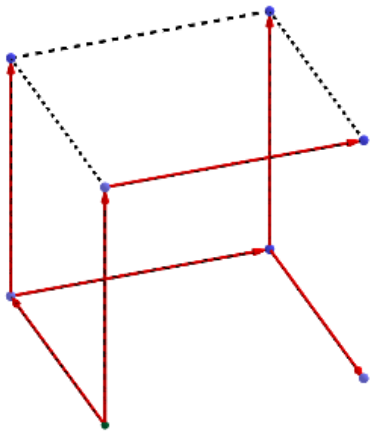
Solving the HIIP

Solution to the HIIP: Use a covering subtree $\mathscr{T} \subseteq G$ to propagate values $x_i \to x_{i+j}$ through edges $(i+j, j) \in E$. We then have an explicit formula to compute $x^{\odot -1}$ with $\mathscr{T}$.

Solving the HIIP

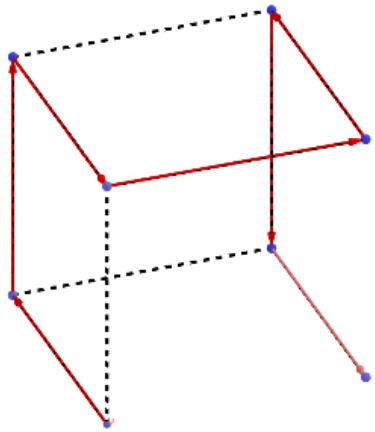
Solution to the HIIP: Use a covering subtree $\mathcal{T} \subseteq G$ to propagate values $x_i \to x_{i+j}$ through edges $(i+j, j) \in E$. We then have an explicit formula to compute $x^{\odot -1}$ with $\mathcal{T}$.

State of the art [Dar25], choose any covering tree $\mathcal{T}$.



✗ Found in variable time.

Our contribution: $\mathcal{T}$ is a Hamiltonian path.



✓ Found in constant time by iterating hypercube automorphisms.

Our contribution: solving the HIIP with a Hamiltonian path

- A Hamiltonian path is given by $\eta : [\![0\ ;\ 2^g-1]\!] \xrightarrow{\sim} (\mathbb{Z}/2\mathbb{Z})^g$ such that $(\eta(i), \eta(i+1)) \in E$ for all $i \in [\![0\ ;\ 2^g-2]\!]$.

$$\begin{array}{ccccccc} X_{\eta(0)} & & X_{\eta(1)} & & X_{\eta(2^g-2)} & & X_{\eta(2^g-1)} \\ \eta(0) & \longrightarrow & \eta(1) & \cdots & \eta(2^g-2) & \longrightarrow & \eta(2^g-1). \end{array}$$

Our contribution: solving the HIIP with a Hamiltonian path

- A Hamiltonian path is given by $\eta : [\![0\ ;\ 2^g-1]\!] \xrightarrow{\sim} (\mathbb{Z}/2\mathbb{Z})^g$ such that $(\eta(i), \eta(i+1)) \in E$ for all $i \in [\![0\ ;\ 2^g-2]\!]$.

$$\begin{array}{ccccccc} x_{\eta(0)} & & x_{\eta(1)} & & x_{\eta(2^g-2)} & & x_{\eta(2^g-1)} \\ \eta(0) & \longrightarrow & \eta(1) & \cdots & \eta(2^g-2) & \longrightarrow & \eta(2^g-1). \end{array}$$

- Projectively, we have:

$$x_{\eta(i)}^{-1} = \left(\prod_{j=0}^{i-1} \pi_{\eta(j),s_j}\right) \cdot \left(\prod_{j=i}^{2^g-2} \pi_{\eta(j+1),s_j}\right) = \rho_L(i) \cdot \rho_R(i)$$

  with $s_j := \eta(j+1) - \eta(j)$.
- With iterative definitions $\rho_L(0) = 1$, $\rho_R(2^g-1) = 1$,

$$\rho_L(i) = \rho_L(i-1) \cdot \pi_{\eta(i-1),s_{i-1}} \quad \text{and} \quad \rho_R(i) = \rho_R(i+1) \cdot \pi_{\eta(i+1),s_i}.$$

Our contribution: solving the HIIP with a Hamiltonian path

- A Hamiltonian path is given by $\eta : [\![0\ ;\ 2^g-1]\!] \xrightarrow{\sim} (\mathbb{Z}/2\mathbb{Z})^g$ such that $(\eta(i), \eta(i+1)) \in E$ for all $i \in [\![0\ ;\ 2^g-2]\!]$.

$$\begin{array}{ccccccc} x_{\eta(0)} & & x_{\eta(1)} & & x_{\eta(2^g-2)} & & x_{\eta(2^g-1)} \\ \eta(0) & \longrightarrow & \eta(1) & \cdots & \eta(2^g-2) & \longrightarrow & \eta(2^g-1). \end{array}$$

- Projectively, we have:

$$x_{\eta(i)}^{-1} = \left(\prod_{j=0}^{i-1} \pi_{\eta(j),s_j}\right) \cdot \left(\prod_{j=i}^{2^g-2} \pi_{\eta(j+1),s_j}\right) = \rho_L(i) \cdot \rho_R(i)$$

  with $s_j := \eta(j+1) - \eta(j)$.
- With iterative definitions $\rho_L(0) = 1$, $\rho_R(2^g-1) = 1$,

$$\rho_L(i) = \rho_L(i-1) \cdot \pi_{\eta(i-1),s_{i-1}} \quad \text{and} \quad \rho_R(i) = \rho_R(i+1) \cdot \pi_{\eta(i+1),s_i}.$$

- Algorithmic cost to solve the HIIP: $3(2^g-2)\mathrm{M}$ vs $(6 \cdot 2^g - 9)\mathrm{M}$ state-of-the-art.

Down the rabbit hole of gluing 2-isogenies

Onto gluing

$$\underbrace{E^4 = A \xrightarrow{f_1} A_1}_{\text{Gluing}} \underbrace{\xrightarrow{f_2} A_2 \xrightarrow{f_2} \cdots \xrightarrow{f^{e-1}} A_{e-1}}_{\text{Generic}} \underbrace{\xrightarrow{f_e} B = E_{\mathfrak{a}} \times E_{\overline{\mathfrak{a}}} \times S}_{\text{Spliting}}$$

Onto gluing

$$\underbrace{E^4 = A \xrightarrow{f_1} A_1}_{\text{Gluing}} \underbrace{\xrightarrow{f_2} A_2 \xrightarrow{f_2} \cdots \xrightarrow{f^{e-1}} A_{e-1}}_{\text{Generic}} \underbrace{\xrightarrow{f_e} B = E_a \times E_{\overline{a}} \times S}_{\text{Spliting}}$$

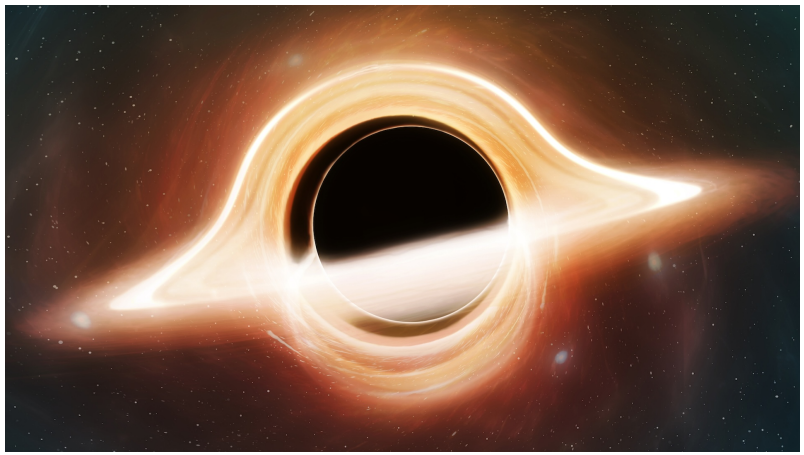


Figure: Artist view of a gluing isogeny

Image credit: Mark Garlick/Science Photo Library/Getty Images

Onto gluing

$$\underbrace{E^4 = A \xrightarrow{f_1} A_1}_{\text{Gluing}} \underbrace{\xrightarrow{f_2} A_2 \xrightarrow{f_2} \cdots \xrightarrow{f^{e-1}} A_{e-1}}_{\text{Generic}} \underbrace{\xrightarrow{f_e} B = E_a \times E_{\overline{a}} \times S}_{\text{Spliting}}$$

Gluing is a rabbit-hole of technicalities.

- Lots of additional technical problems.

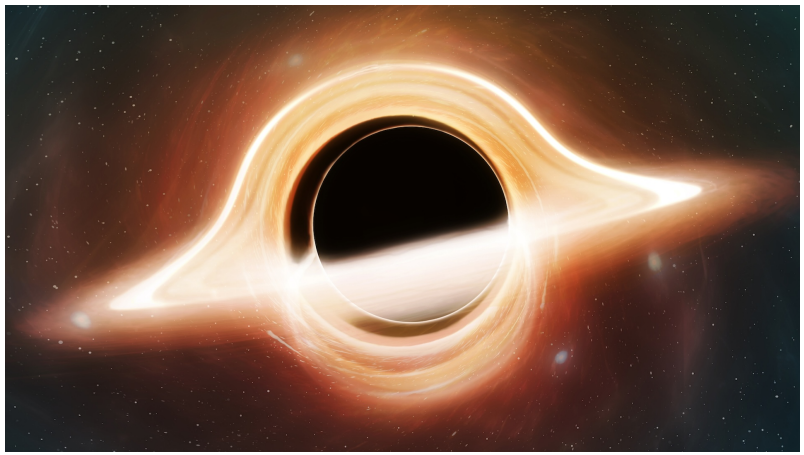


Figure: Artist view of a gluing isogeny

Image credit: Mark Garlick/Science Photo Library/Getty Images

Onto gluing

$$\underbrace{E^4 = A \xrightarrow{f_1} A_1}_{\text{Gluing}} \underbrace{\xrightarrow{f_2} A_2 \xrightarrow{f_2} \cdots \xrightarrow{f^{e-1}} A_{e-1}}_{\text{Generic}} \underbrace{\xrightarrow{f_e} B = E_a \times E_{\overline{a}} \times S}_{\text{Spliting}}$$

Gluing is a rabbit-hole of technicalities.

- Lots of additional technical problems.
- Can be solved thanks to the graph theoretical PoV.

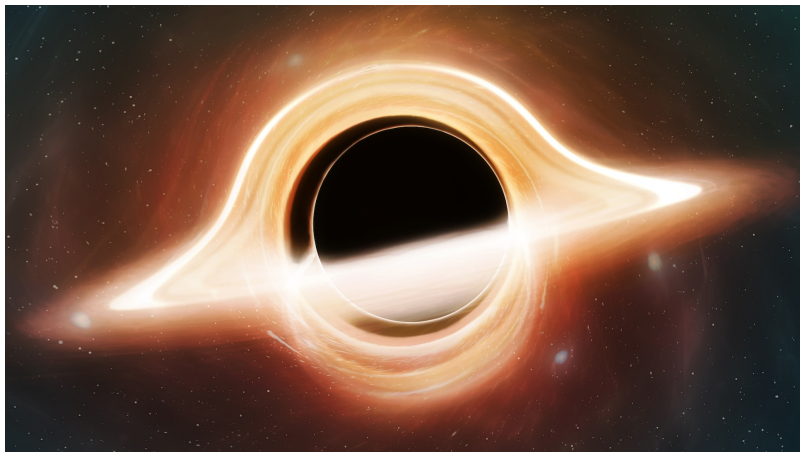


Figure: Artist view of a gluing isogeny

Image credit: Mark Garlick/Science Photo Library/Getty Images

On the generic covering tree computation

Goal: Finding valid Hamiltonian path

On the generic covering tree computation

Goal: Finding valid Hamiltonian path

Finding $\mathscr{T} \subset G$

1. Take Gray path $\mathscr{P}$

$v \in \mathbb{Z}_2^4$ and $i \circ \sigma := (i_{\sigma(m)})_{1 \leq m \leq 4}, \sigma \in \mathfrak{S}_4$

On the generic covering tree computation

Goal: Finding valid Hamiltonian path

Finding $\mathscr{T} \subset G$

1. Take Gray path $\mathscr{P}$

$v \in \mathbb{Z}_2^4$ and $i \circ \sigma := (i_{\sigma(m)})_{1 \leq m \leq 4}, \sigma \in \mathfrak{S}_4$

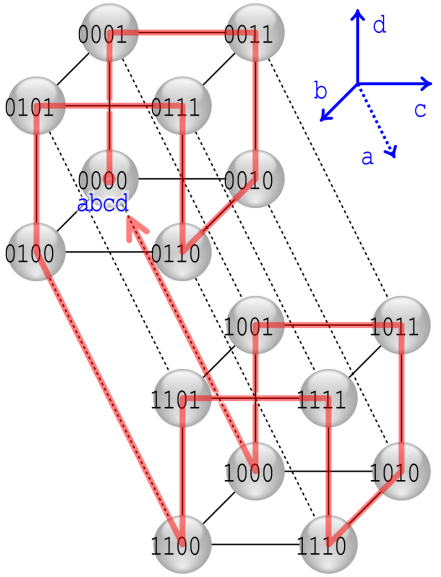


Figure: Gray Path $\mathscr{P}$ in H_4

On the generic covering tree computation

Goal: Finding valid Hamiltonian path

Finding $\mathscr{T} \subset G$

1. Take Gray path $\mathscr{P}$
2. Use $\mathrm{Aut}(H_4)$:

$$\text{Find } \Delta_{\sigma,v} : i \in \mathbb{Z}_2^4 \longmapsto i \circ \sigma + v$$

such that $\Delta_{\sigma,v}(\mathscr{P}) \subset G$.

$v \in \mathbb{Z}_2^4$ and $i \circ \sigma := (i_{\sigma(m)})_{1 \leq m \leq 4}, \sigma \in \mathfrak{S}_4$

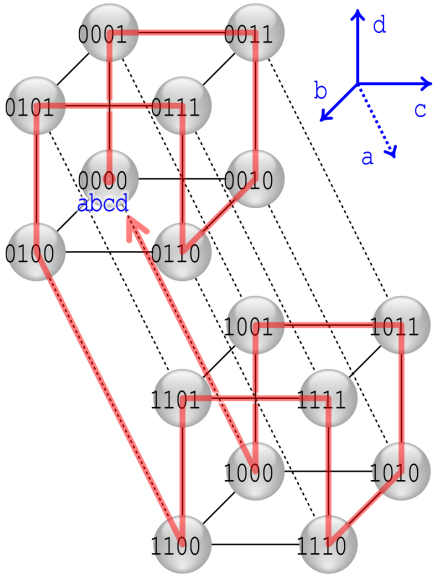


Figure: Gray Path $\mathscr{P}$ in H_4

Gluing isogenies

- Problem: Can not recover all coordinates of $U^B(f(P))$

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P) \odot \theta^A(P))$$

Gluing isogenies

- Problem: Can not recover all coordinates of $U^B(f(P))$

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P) \odot \theta^A(P))$$

  - Recovering lost coordinates doable but technical using theta group action (over $\mathbb{F}_{p^2}$).

Gluing isogenies

- Problem: Can not recover all coordinates of $U^B(f(P))$

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P) \odot \theta^A(P))$$

  - Recovering lost coordinates doable but technical using theta group action (over $\mathbb{F}_{p^2}$).
- Solution: Use T generic.

$$U^B(f(T)) \odot U^B(f(P)) = H(\theta^A(P+T) \odot \theta^A(P-T))$$

Gluing isogenies

- Problem: Can not recover all coordinates of $U^B(f(P))$

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P) \odot \theta^A(P))$$

  - Recovering lost coordinates doable but technical using theta group action (over $\mathbb{F}_{p^2}$).
- Solution: Use T generic.

$$U^B(f(T)) \odot U^B(f(P)) = H(\theta^A(P+T) \odot \theta^A(P-T))$$

  - $\theta^A(P+T)$ and $\theta^A(P-T)$ can be computed simultaneously and efficiently using *barycentric coordinates* [Dup25].

Gluing isogenies

- Problem: Can not recover all coordinates of $U^B(f(P))$

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P) \odot \theta^A(P))$$

  - Recovering lost coordinates doable but technical using theta group action (over $\mathbb{F}_{p^2}$).
- Solution: Use T generic.

$$U^B(f(T)) \odot U^B(f(P)) = H(\theta^A(P+T) \odot \theta^A(P-T))$$

  - $\theta^A(P+T)$ and $\theta^A(P-T)$ can be computed simultaneously and efficiently using *barycentric coordinates* [Dup25].
  - Using compatibility with twist, can be computed over just $\mathbb{F}_p$.

Gluing isogenies

- Problem: Can not recover all coordinates of $U^B(f(P))$

$$U^B(0_B) \odot U^B(f(P)) = H(\theta^A(P) \odot \theta^A(P))$$

  - Recovering lost coordinates doable but technical using theta group action (over $\mathbb{F}_{p^2}$).
- Solution: Use T generic.

$$U^B(f(T)) \odot U^B(f(P)) = H(\theta^A(P+T) \odot \theta^A(P-T))$$

  - $\theta^A(P+T)$ and $\theta^A(P-T)$ can be computed simultaneously and efficiently using *barycentric coordinates* [Dup25].
  - Using compatibility with twist, can be computed over just $\mathbb{F}_p$.

▶ Working with $U^B(f(T))^{\odot -1}$, our gluing graphs are special.

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Figure: A rabbit picture

Image credit: Izuzuki Diver: CC BY-CA 2.5

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Rabbit classification lemma

1. There are 32 rabbits, identical except for fur colour, of which there are 4.

Figure: A rabbit picture

Image credit: Izuzuki Diver: CC BY-CA 2.5

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Rabbit classification lemma

1. There are 32 rabbits, identical except for fur colour, of which there are 4.
2. Can dye rabbits into another colour through bit-swapping the indices.

Figure: A rabbit picture

Image credit: Izuzuki Diver: CC BY-CA 2.5

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Rabbit classification lemma

1. There are 32 rabbits, identical except for fur colour, of which there are 4.
2. Can dye rabbits into another colour through bit-swapping the indices.
3. Rabbits need 5 carrots to be satiated.

Figure: A rabbit picture

Image credit: Izuzuki Diver: CC BY-CA 2.5

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Rabbit classification lemma

1. There are 32 rabbits, identical except for fur colour, of which there are 4.
2. Can dye rabbits into another colour through bit-swapping the indices.
3. Rabbits need 5 carrots to be satiated.
4. Under X-ray, rabbits are a long skeleton, with two ears on top.

Figure: A rabbit picture

Image credit: Izuzuki Diver: CC BY-CA 2.5

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Rabbit classification lemma

1. All rabbits are isomorphic.
2. Isomorphism are given by Δ_σ with $\sigma \in V_4$ Klein's four-group.
3. Rabbits need 5 points evaluation to be connected.
4. The rabbit body has an Hamiltonian path in his body, with two ears on top.

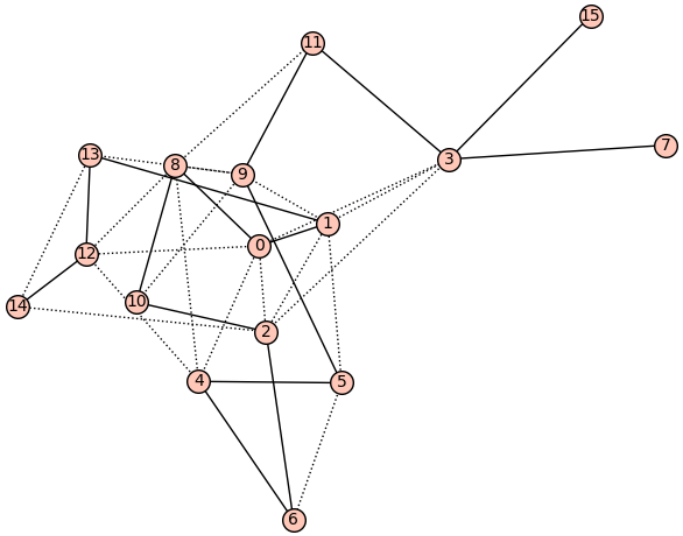


Figure: A rabbit picture

Rabbits

In qt-Pegasis, gluing graphs are rabbits !

Rabbit classification lemma

1. All rabbits are isomorphic.
2. Isomorphism are given by Δ_σ with $\sigma \in V_4$ Klein's four-group.
3. Rabbits need 5 points evaluation to be connected.
4. The rabbit body has an Hamiltonian path in his body, with two ears on top.

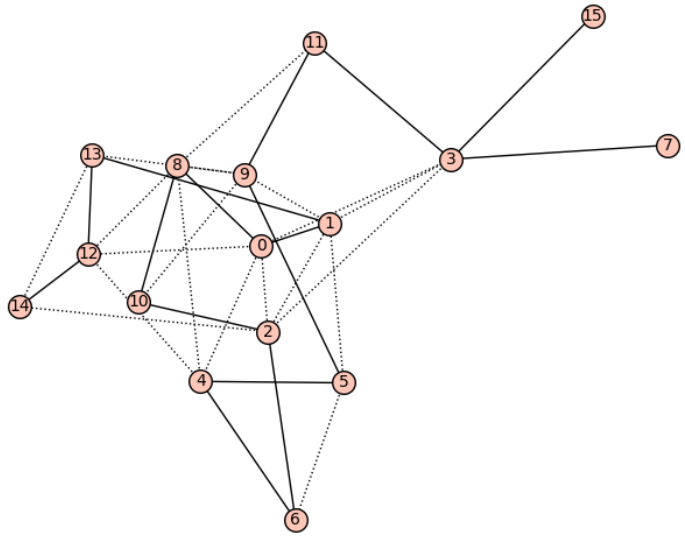


Figure: A rabbit picture

- By interpolating on rabbits, made constant time gluing.

Performance results

Comparison to state of the art

Thanks to our work, we gain:

- A constant time 2-isogeny chain.
- Faster kernel computation and gluing over $\mathbb{F}_p$.
- Faster generic codomain computations.

$\lceil \log_2(p) \rceil$	Old formulae (s)	New formulae (s)	Gain
505	0.757	0.613	19.0 %
1008	2.155	1.889	12.3 %
1525	4.516	4.013	12.3 %
2017	8.152	7.431	11.1 %
4030	41.426	39.073	5.7 %

Table: Comparison between new and old formulae in the SageMath implementation of qt-Pegasis. Average of 100 runs[2].

Timing captured on an AMD Ryzen 77840 U 8 cores CPU at 3.3 GHz, running Ubuntu 24.04.02 LTS, SageMath 10.6 and Python 3.12.11.

Constant time implementation of qt-Pegasis in C (4D chain)

$\lceil \log_2(p) \rceil$	p	ARM[1] (Mcycles)	X86[2] (Mcycles)	Timing[3] (ms)
505	$27 \cdot 2^{500} - 1$	146.95	274.38	23.94
1008	$15 \cdot 2^{1004} - 1$	871.34	1627.05	146.19
1525	$5 \cdot 2^{1522} - 1$	3163.05	5208.81	543.01
2017	$5 \cdot 2^{2014} - 1$	7041.82	11192.45	1222.96
4030	$45 \cdot 2^{4024} - 1$	55228.89	83757.68	10654.93

Table: Performance of the C-implementation of 4-dimensional chain of qt-Pegasis in CPU Mcycles. Results are the average of 100 benchmark runs.

1. Apple M1 Pro 8 cores CPU with nominal clock speed 3.2 GHz, running macOS 15.6.1 and using Clang v16.0.0.
2. Intel Xeon(R) CPU E5-1650 v3 6 cores with nominal clock speed 3.5 GHz, running Manjaro 5.10.237, and using gcc 15.1.1.
3. Apple M4 Pro CPU with nominal clock speed of 4.5 GHz, running macOS 26.2 and using Clang v17.0.0.

Conclusion

1. Dim 4 chains are cryptographically practical!

Conclusion

1. Dim 4 chains are cryptographically practical!
2. Have all tools for generic dimension!

Conclusion

1. Dim 4 chains are cryptographically practical!
2. Have all tools for generic dimension!
3. qt-Pegasis allows for advanced cryptographic applications (PQarrots)!

Conclusion

1. Dim 4 chains are cryptographically practical!
2. Have all tools for generic dimension!
3. qt-Pegasis allows for advanced cryptographic applications (PQarrots)!

Thanks for listening !

Beware of rabbits !

Image credit: Killer rabbit of Caerbannog, taken from the historical documentary "Monty Python and the Holy Grail".