Supersingular Isogeny Key Encapsulation

Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev

November 14
ECC 2017
Nijmegen, The Netherlands

Supersingular I… Key Encapsulation

Reza Azarderakhsh, M… w Campagna, Craig Costello, Lu… e Feo, Basil Hess, Brian Koziel,
Brian LaMacchi… ick Longa, Michael Naehrig, Joos… s, Vladimir Soukharev

amazon
UNIVERSITY OF WATERLOO
UNIVERSIT… VERSAI… ST-QUENTI… LINES
TEXAS INSTRUMENTS
Microsoft Resear…
Radboud University
FAU FLORIDA ATLANTIC UNIVERSITY
INFOSEC GLOBAL

2017
Nijmege… e Netherlands

Part 1: Quick re-motivation

Part 2: Quick tutorial recap

Part 3: SIKE

Quantum computers ↔ Cryptopocalypse

- Quantum computers break elliptic curves, finite fields, factoring, everything currently used for PKC

NIST
National Institute
of Standards
and Technology

- NIST calls for quantum-secure key exchange and signatures. Deadline Nov 30, 2017.

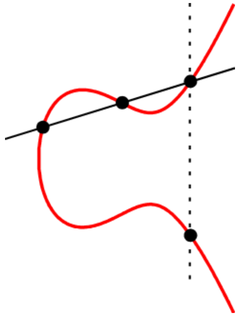
Diffie-Hellman instantiations

$\mathbb{Z}_q$

$g^a \bmod q$

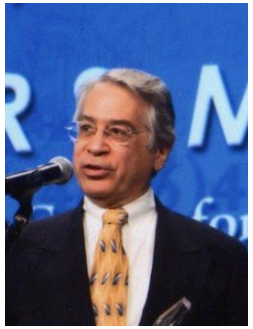
$g^b \bmod q$

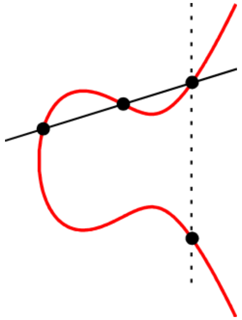
$\mathbb{Z}_q$

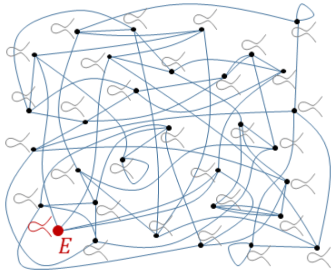


$[a]P$

$[b]P$

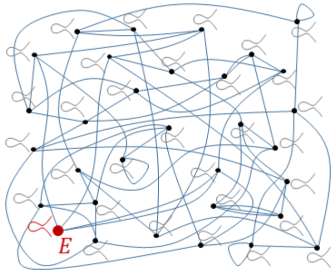






$\phi_A(E)$

$\phi_B(E)$



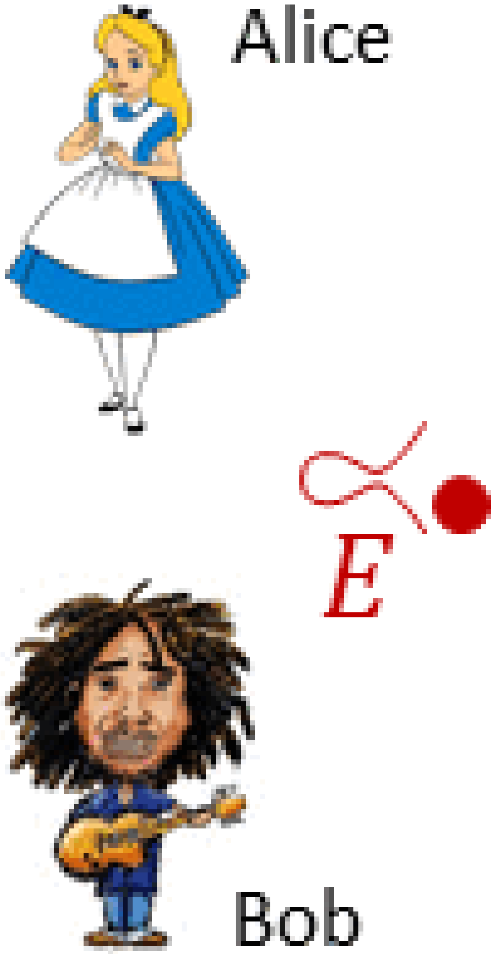
Diffie-Hellman instantiations

	DH	ECDH	SIDH
Elements	integers g modulo prime	points P in curve group	curves E in isogeny class
Secrets	exponents x	scalars k	isogenies ϕ
computations	$g, x \mapsto g^x$	$k, P \mapsto [k]P$	$\phi, E \mapsto \phi(E)$
hard problem	given g, g^x find x	given $P, [k]P$ find k	given $E, \phi(E)$ find ϕ

Part 1: Quick re-motivation

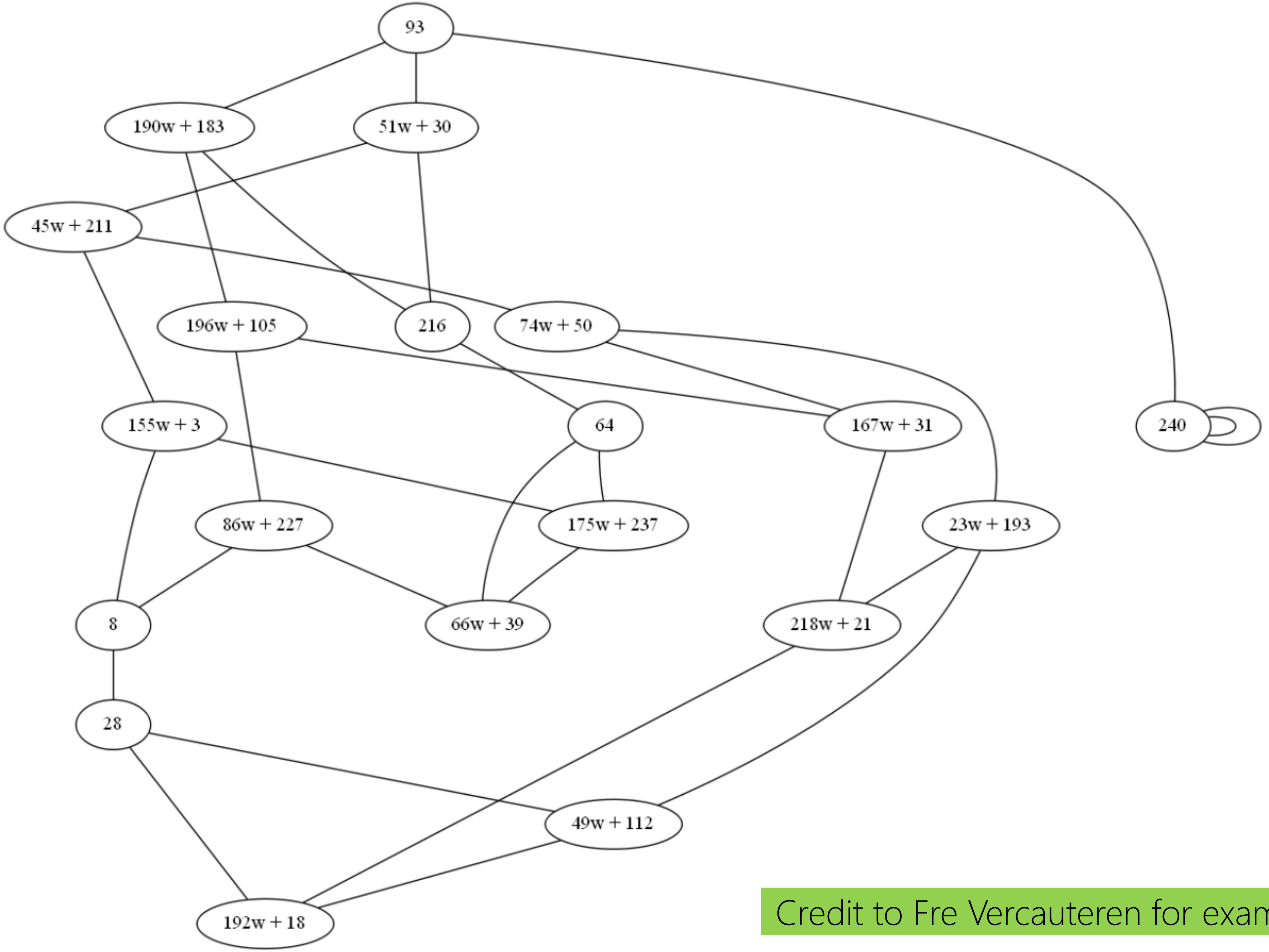
Part 2: Quick tutorial recap

Part 3: SIKE



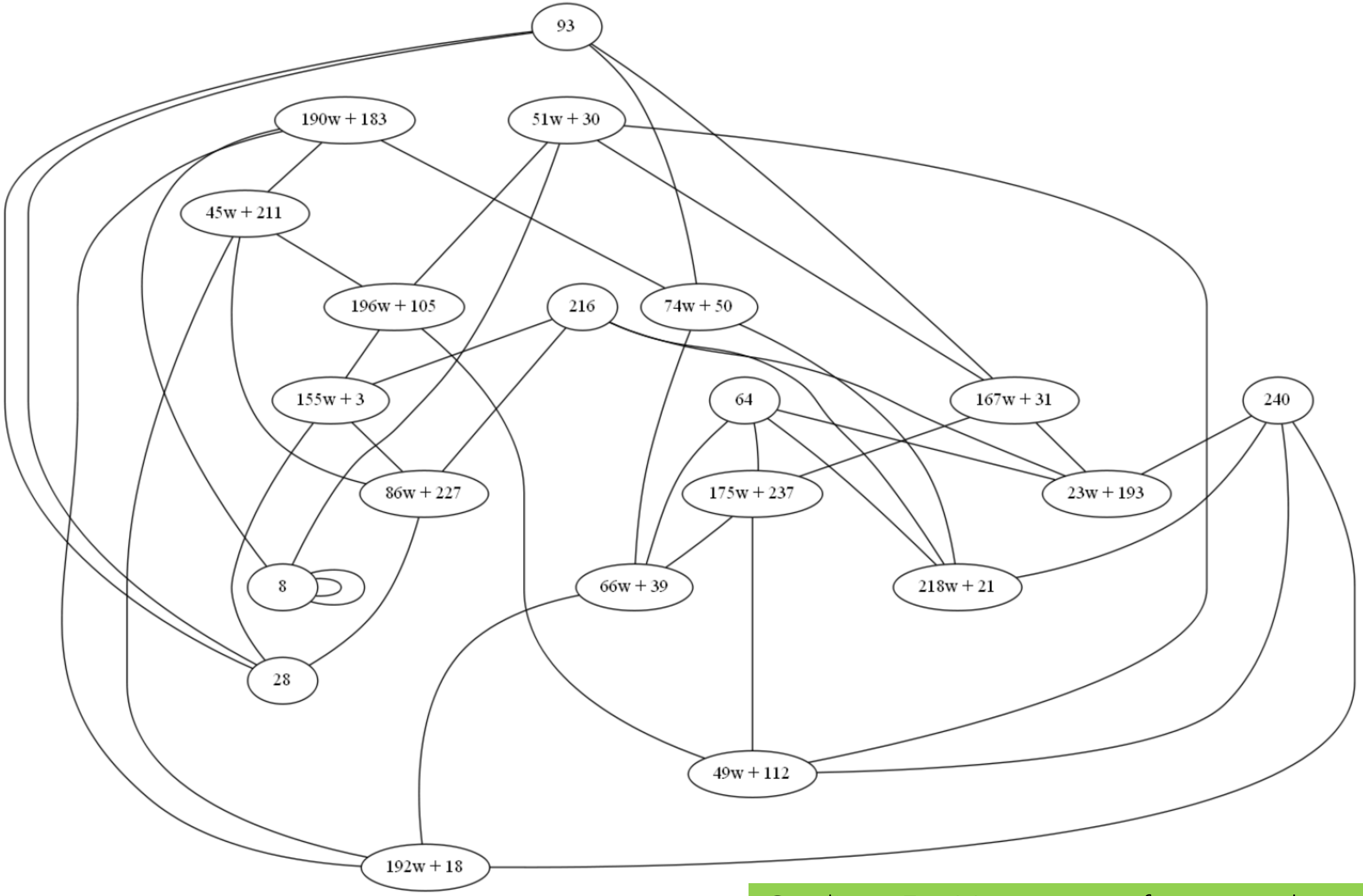
W. Castryck (GIF): "Elliptic curves are dead: long live elliptic curves" https://www.esat.kuleuven.be/cosic/?p=7404

Supersingular isogeny graph for $\ell = 2$: $X(S_{241^2}, 2)$



Credit to Fre Vercauteren for example and pictures…

Supersingular isogeny graph for $\ell = 3$: $X(S_{241^2}, 3)$

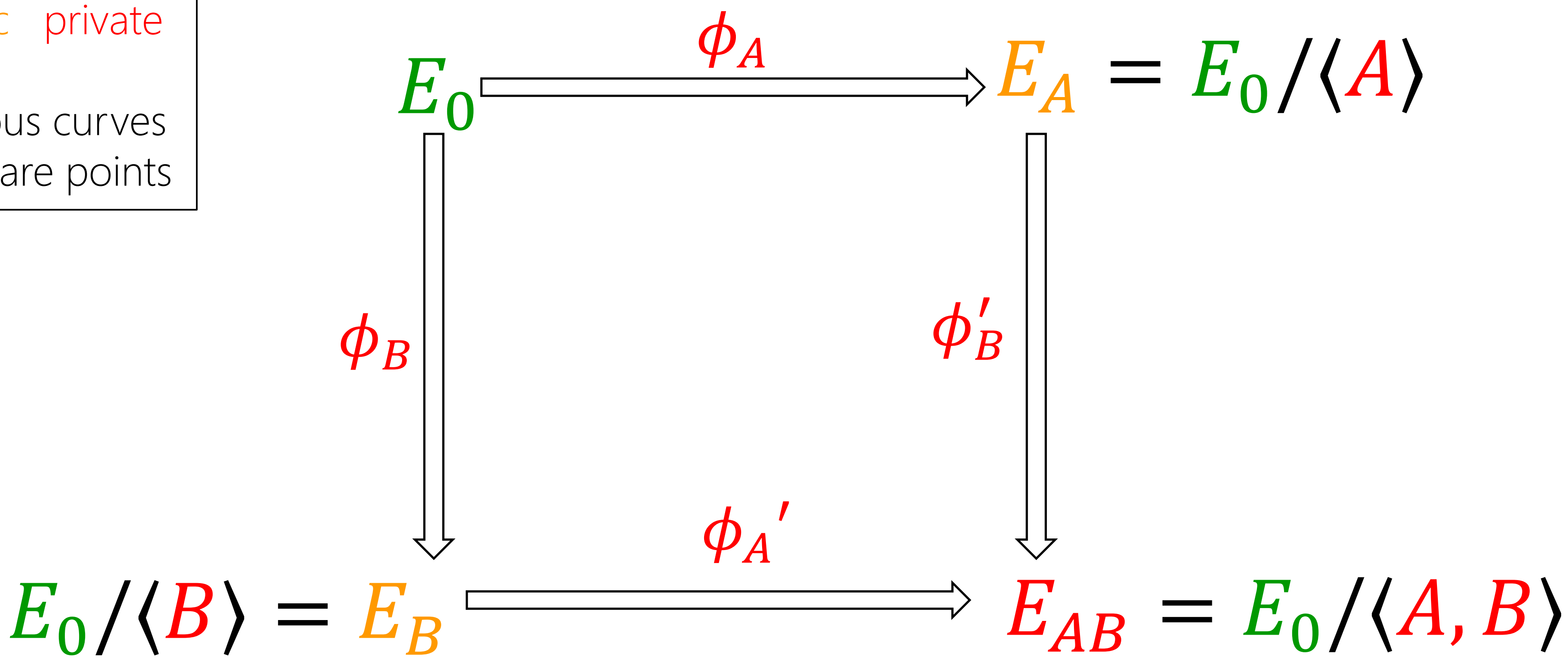


Credit to Fre Vercauteren for example and pictures…

SIDH: in a nutshell

params public private

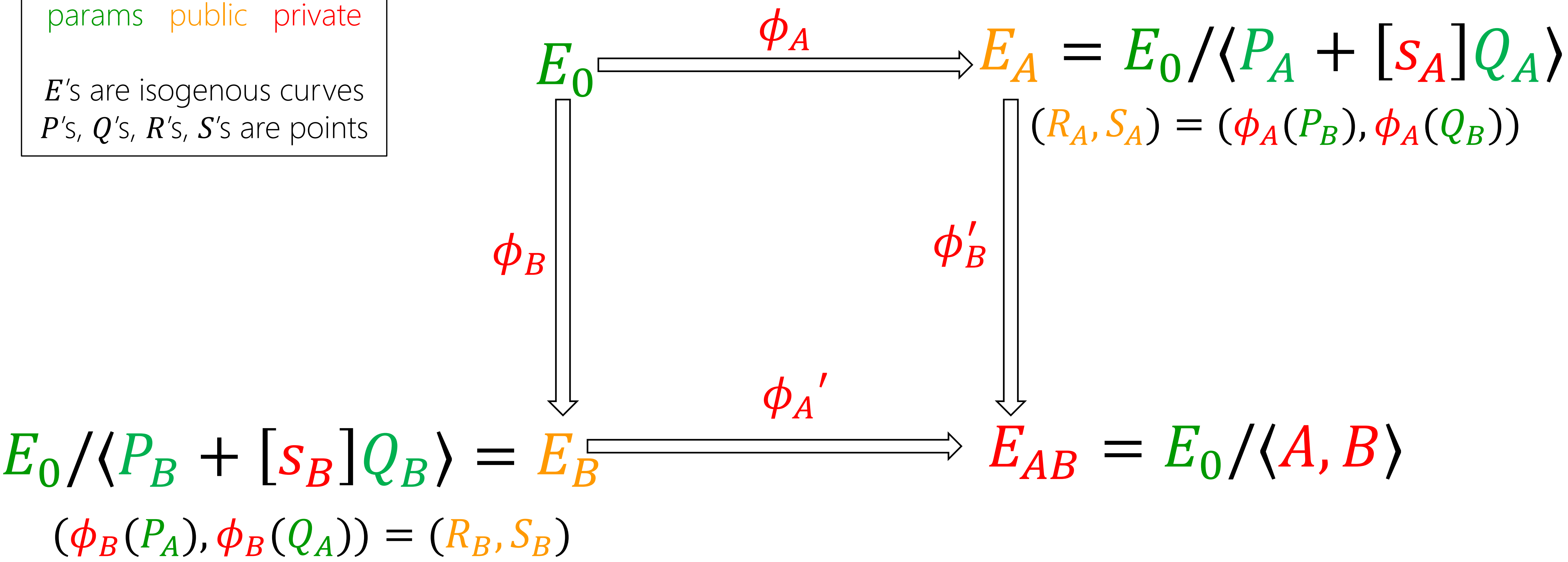
E's are isogenous curves
P's, Q's, R's, S's are points



SIDH: in a nutshell

params public private

E's are isogenous curves
P's, Q's, R's, S's are points



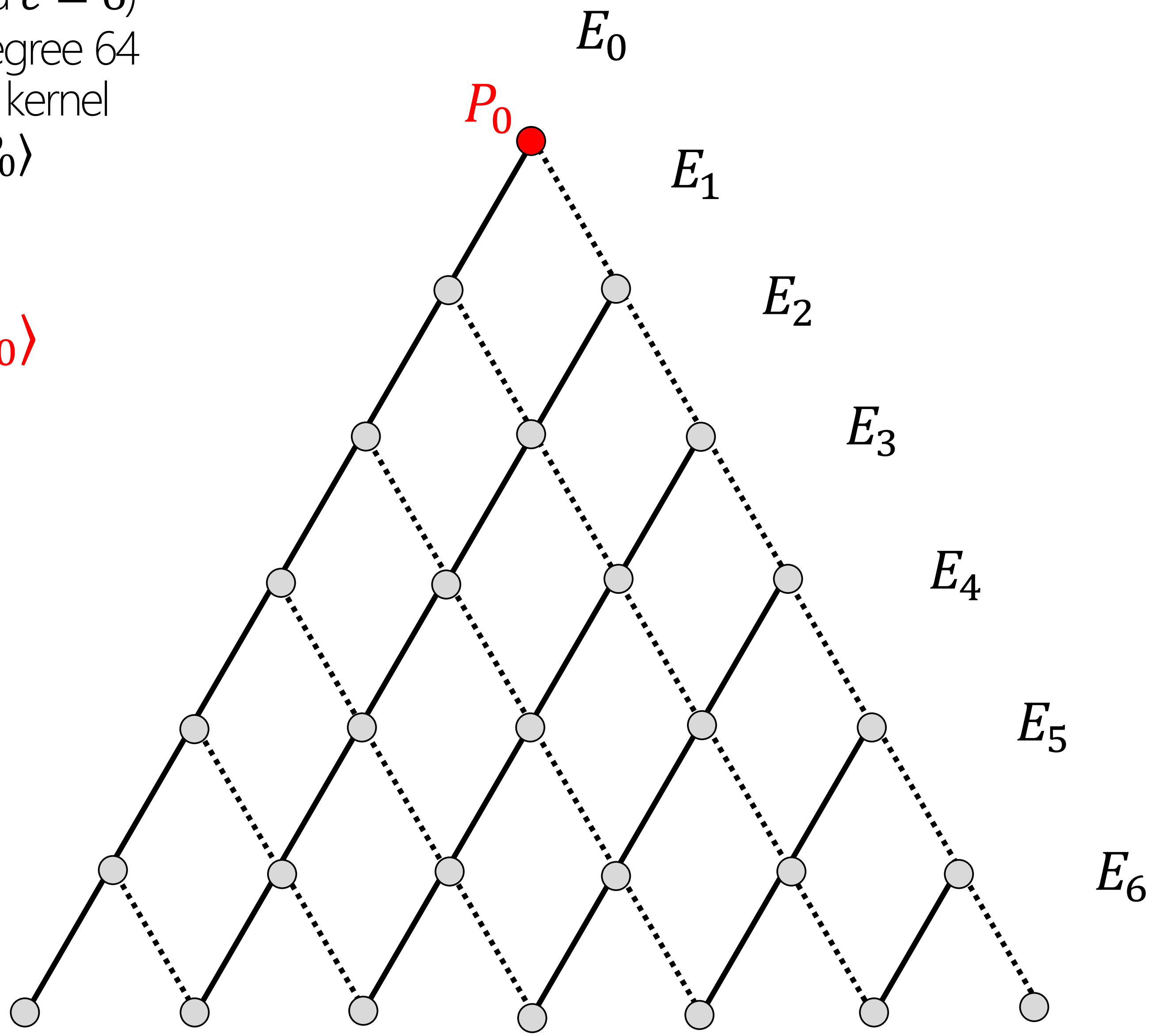
Key: Alice sends her isogeny evaluated at Bob's generators, and vice versa

$$E_A/\langle R_A + [s_B]S_A \rangle \cong E_0/\langle P_A + [s_A]Q_A\,, P_B + [s_B]Q_B \rangle \cong E_B/\langle R_B + [s_A]S_B \rangle$$

Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

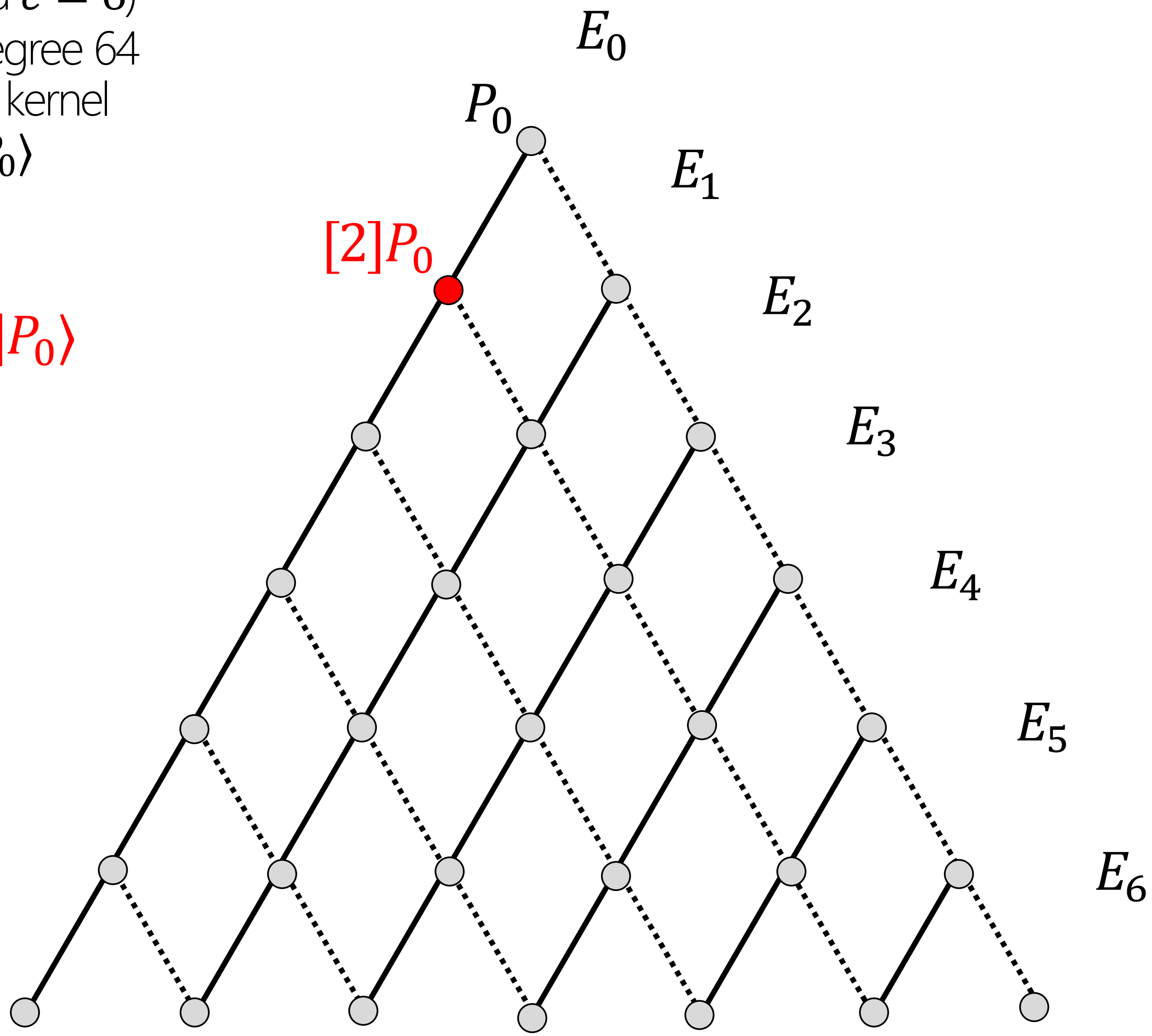
$E_6 = E_0/\langle P_0 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

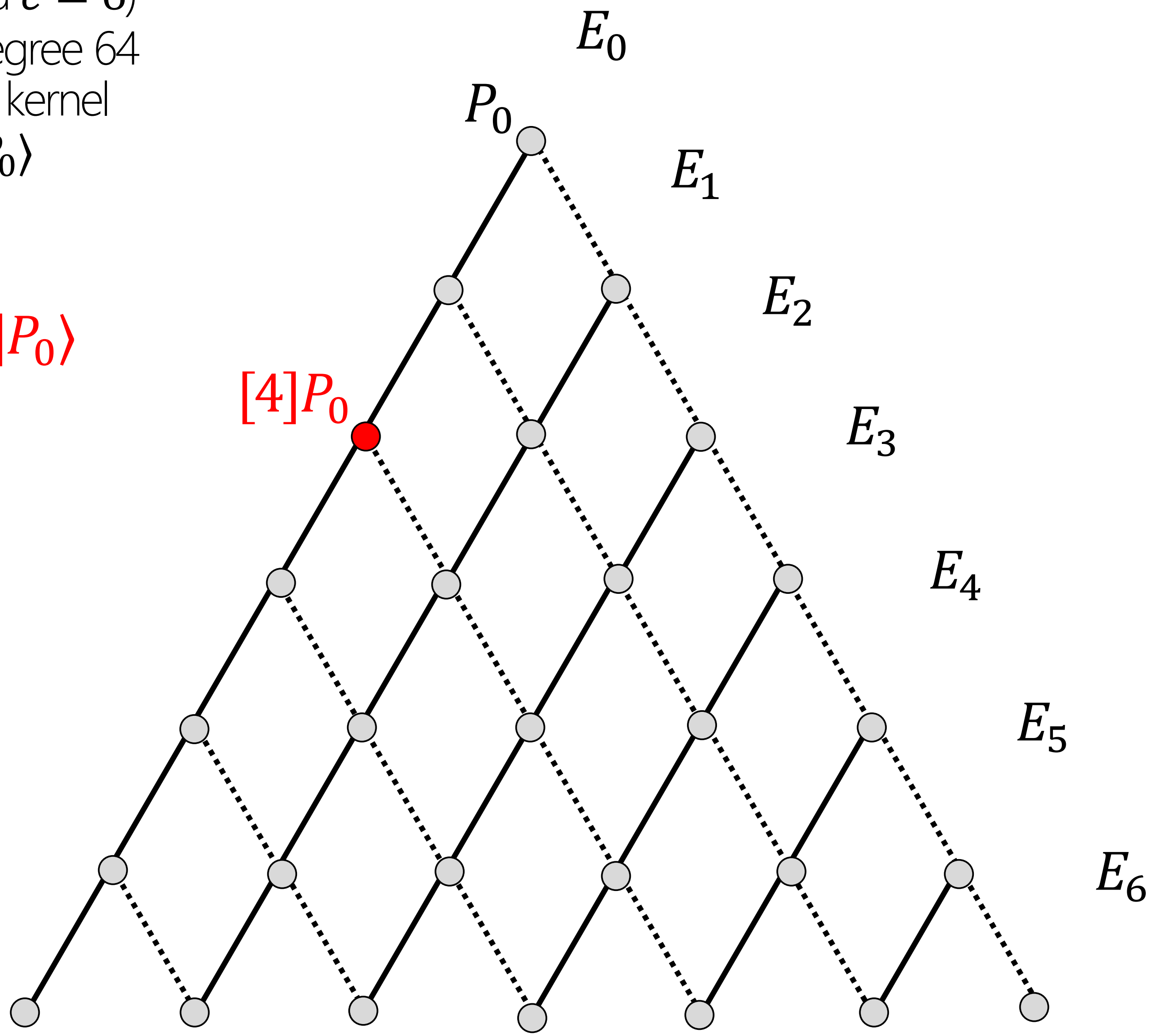
$E_5 = E_0 / \langle [2]P_0 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

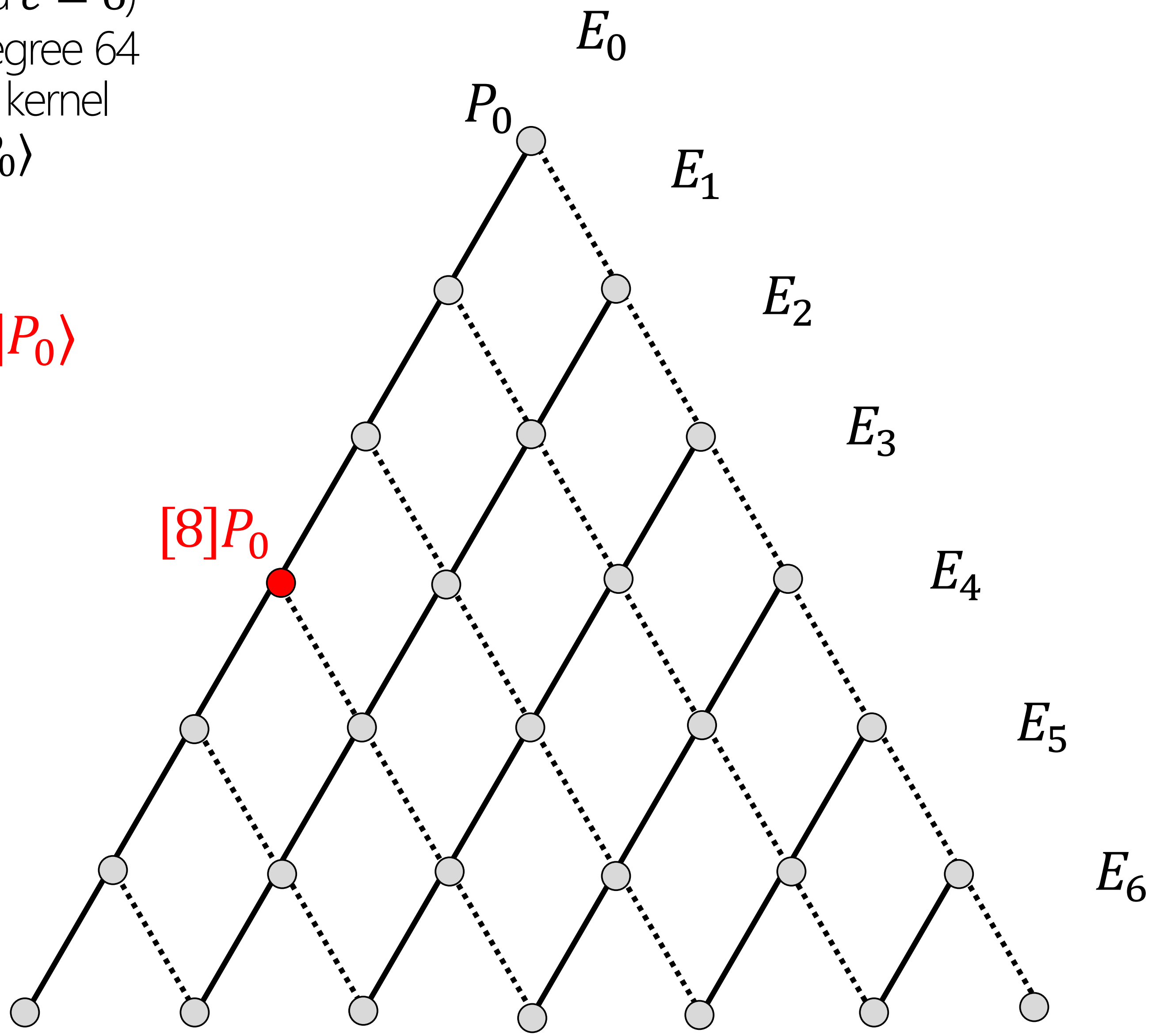
$E_4 = E_0 / \langle [4]P_0 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

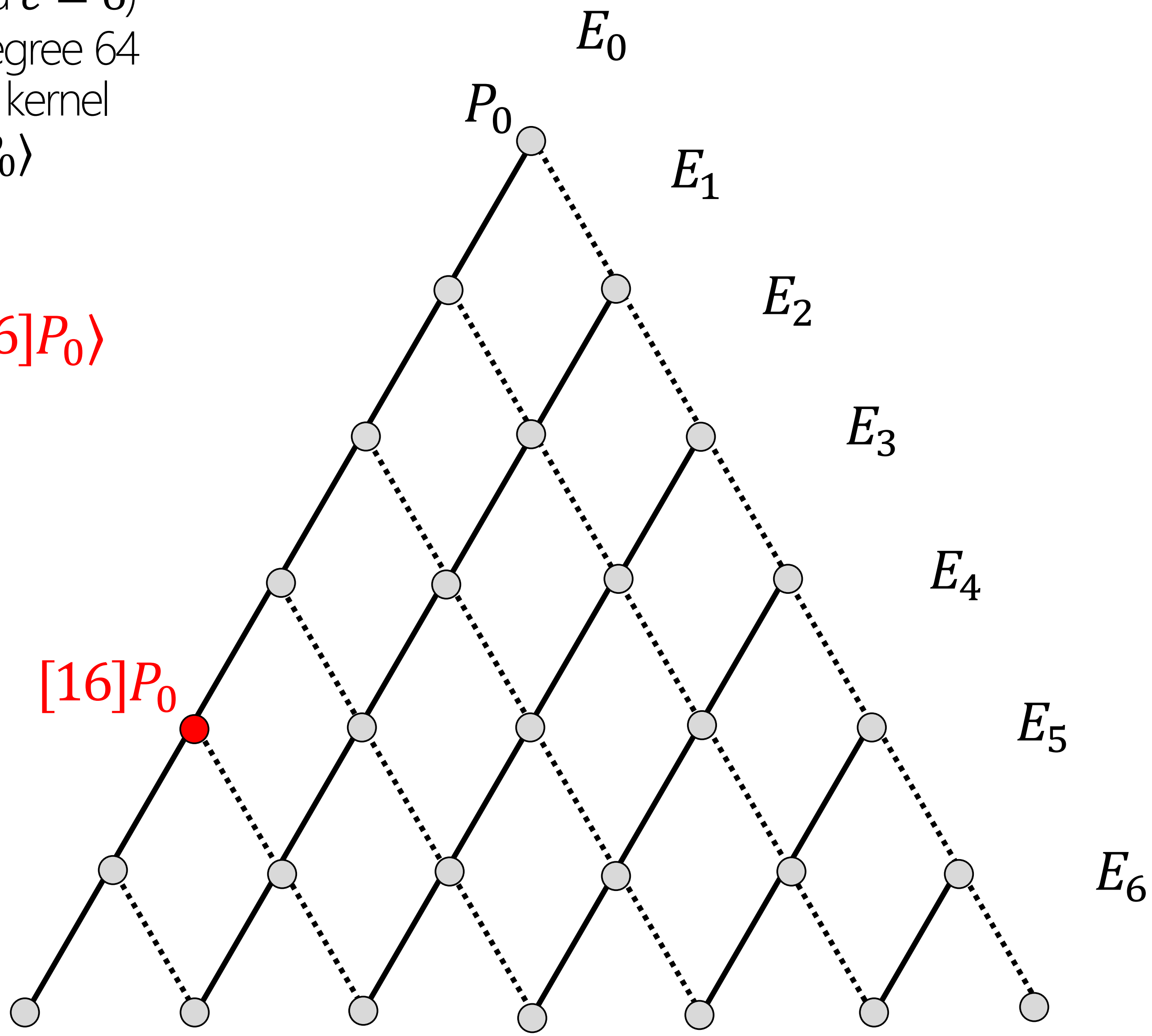
$E_3 = E_0 / \langle [8] P_0 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

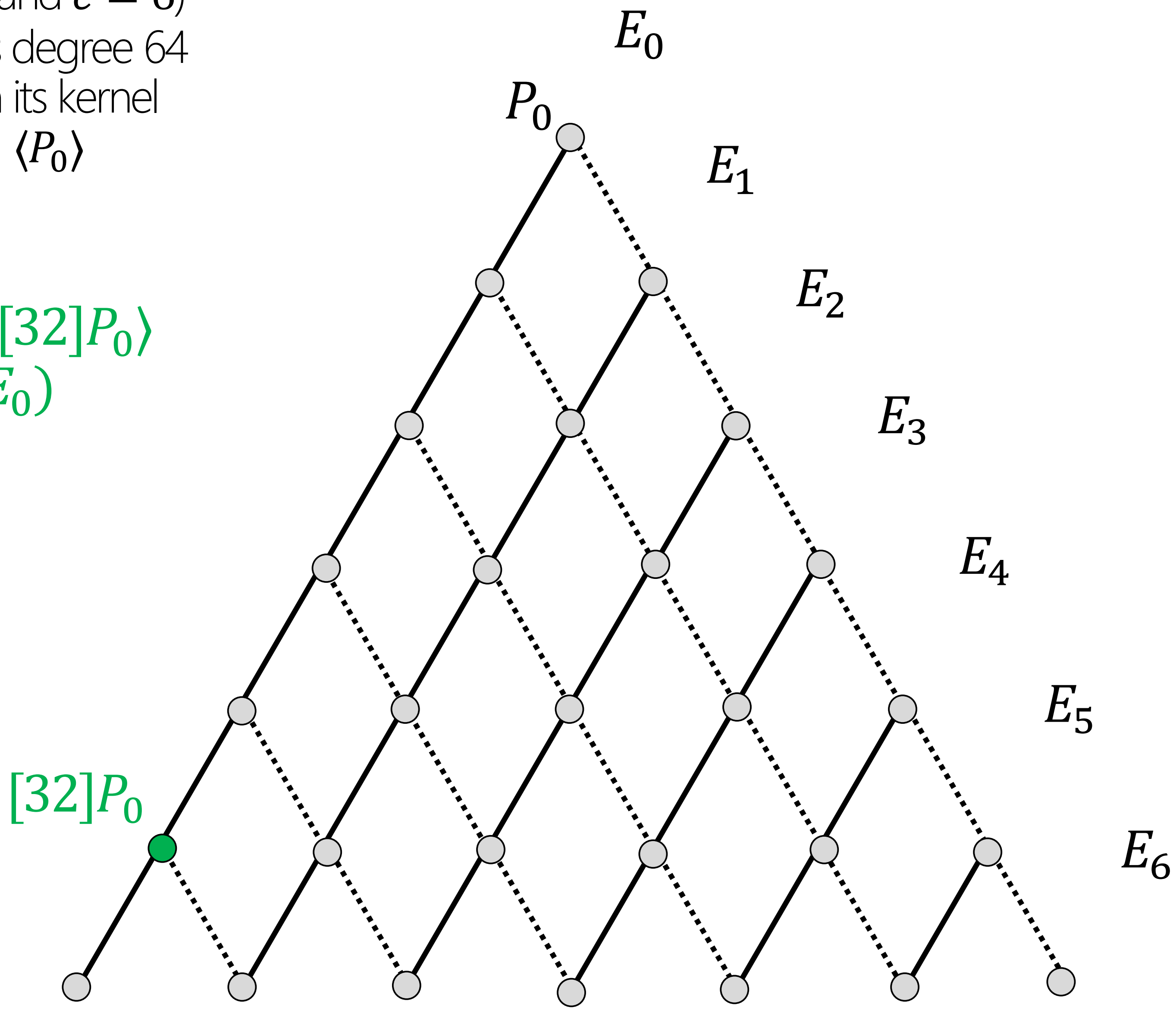
$E_2 = E_0/\langle [16]P_0 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_1 = E_0 / \langle [32]P_0 \rangle$
$= \phi_0(E_0)$

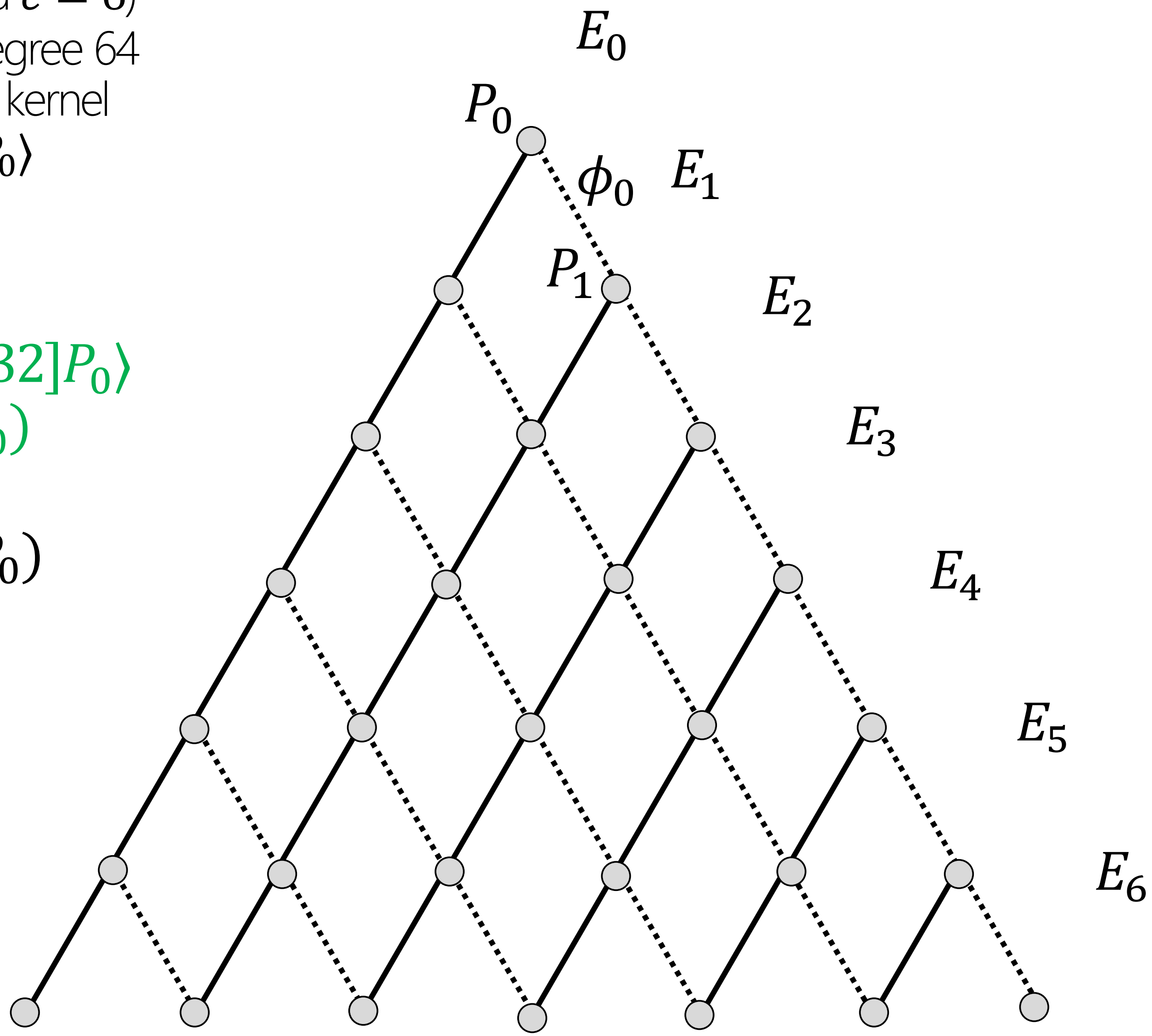


Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$$E_1 = E_0/\langle [32]P_0 \rangle = \phi_0(E_0)$$

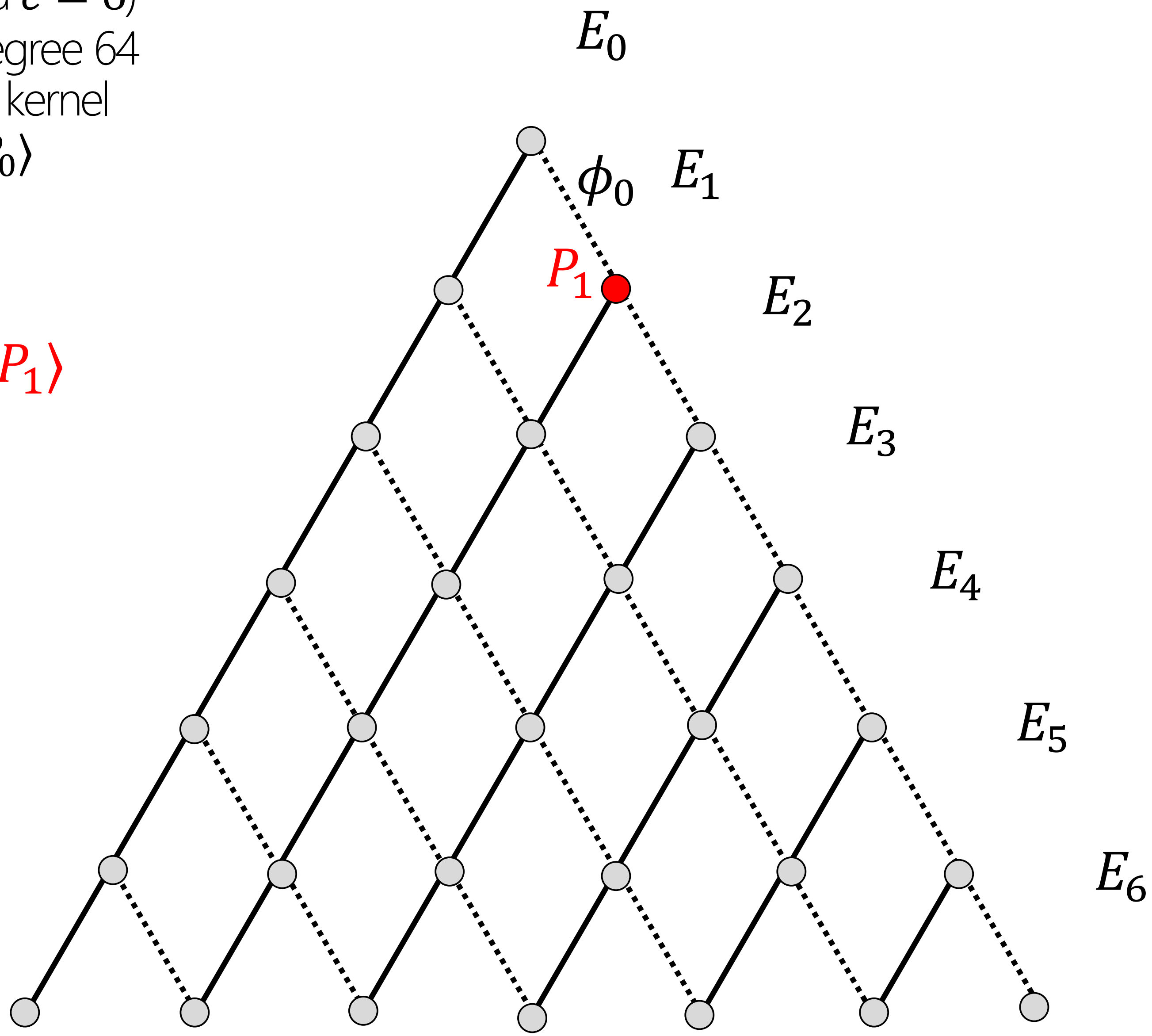
$$P_1 = \phi_0(P_0)$$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

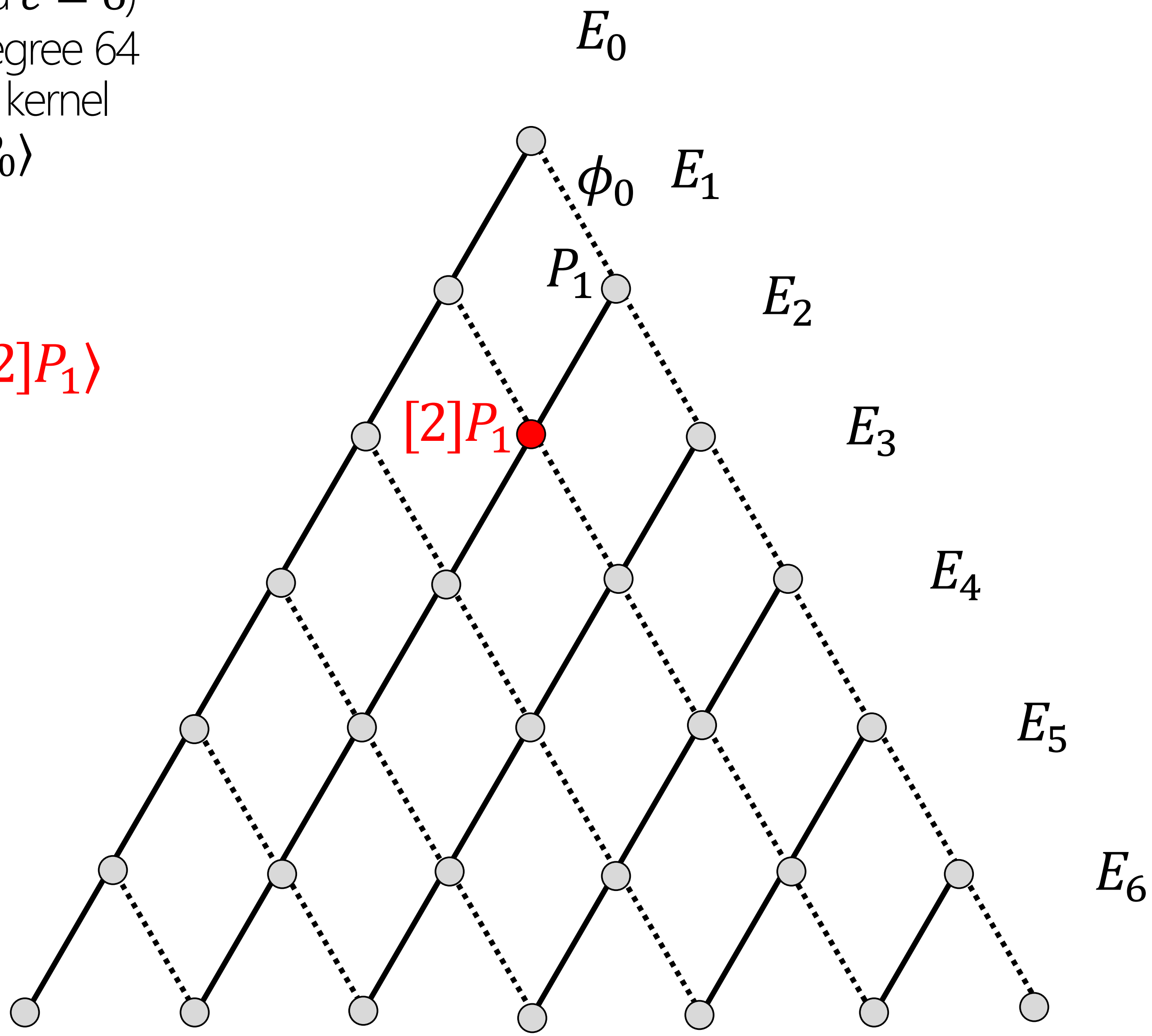
$E_6 = E_1 / \langle P_1 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

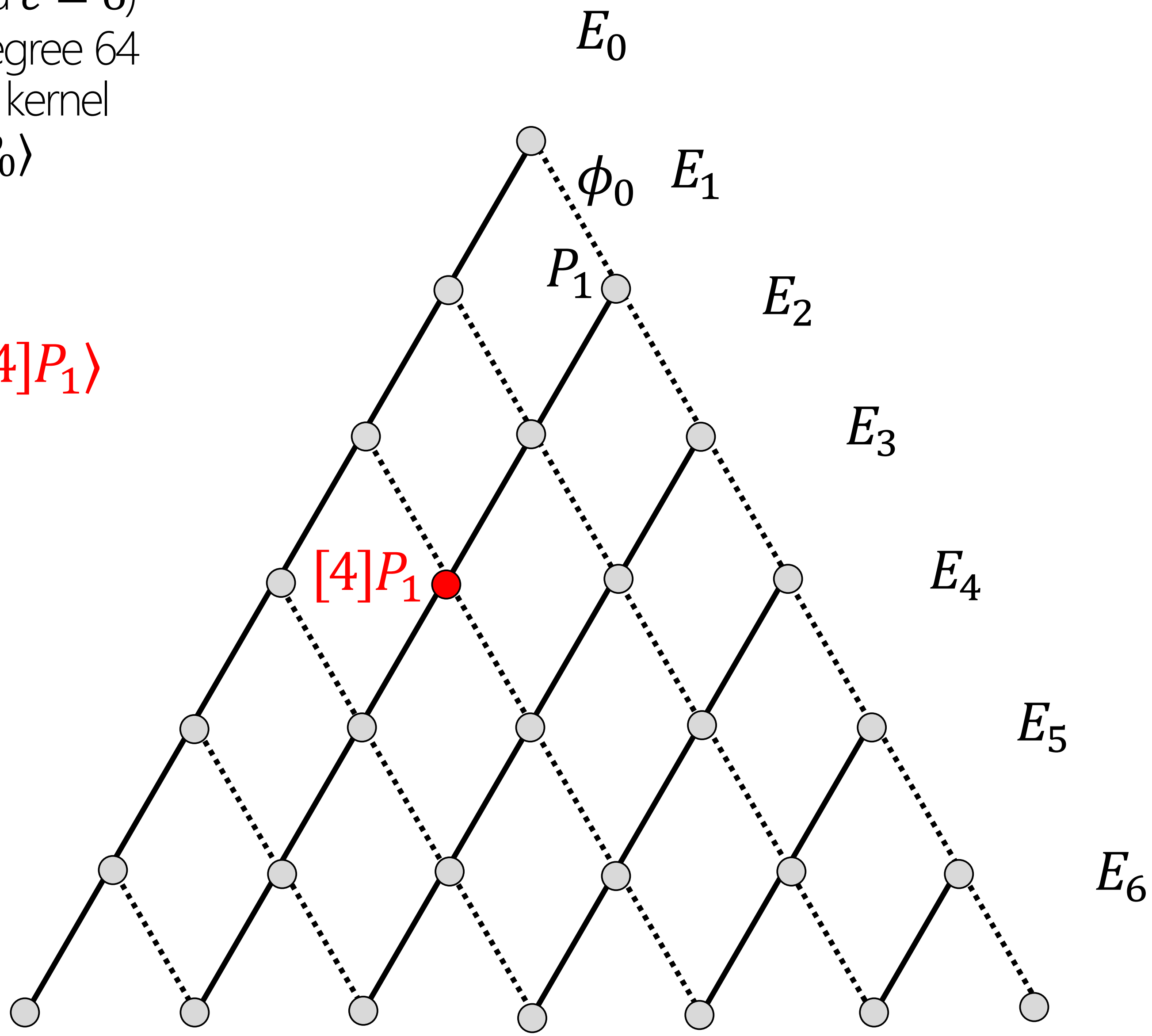
$E_5 = E_1/\langle [2]P_1 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

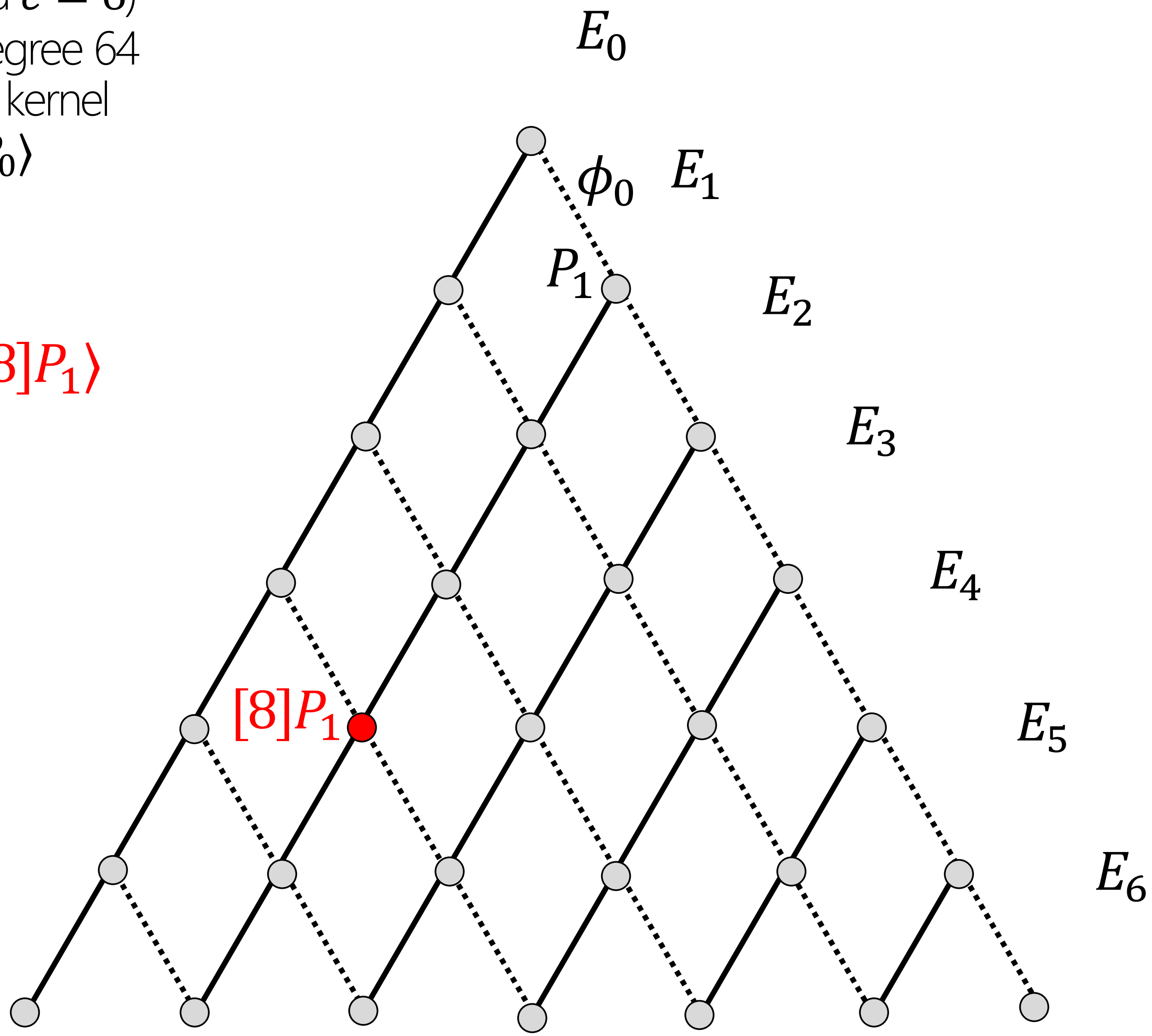
$E_4 = E_1/\langle [4]P_1 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

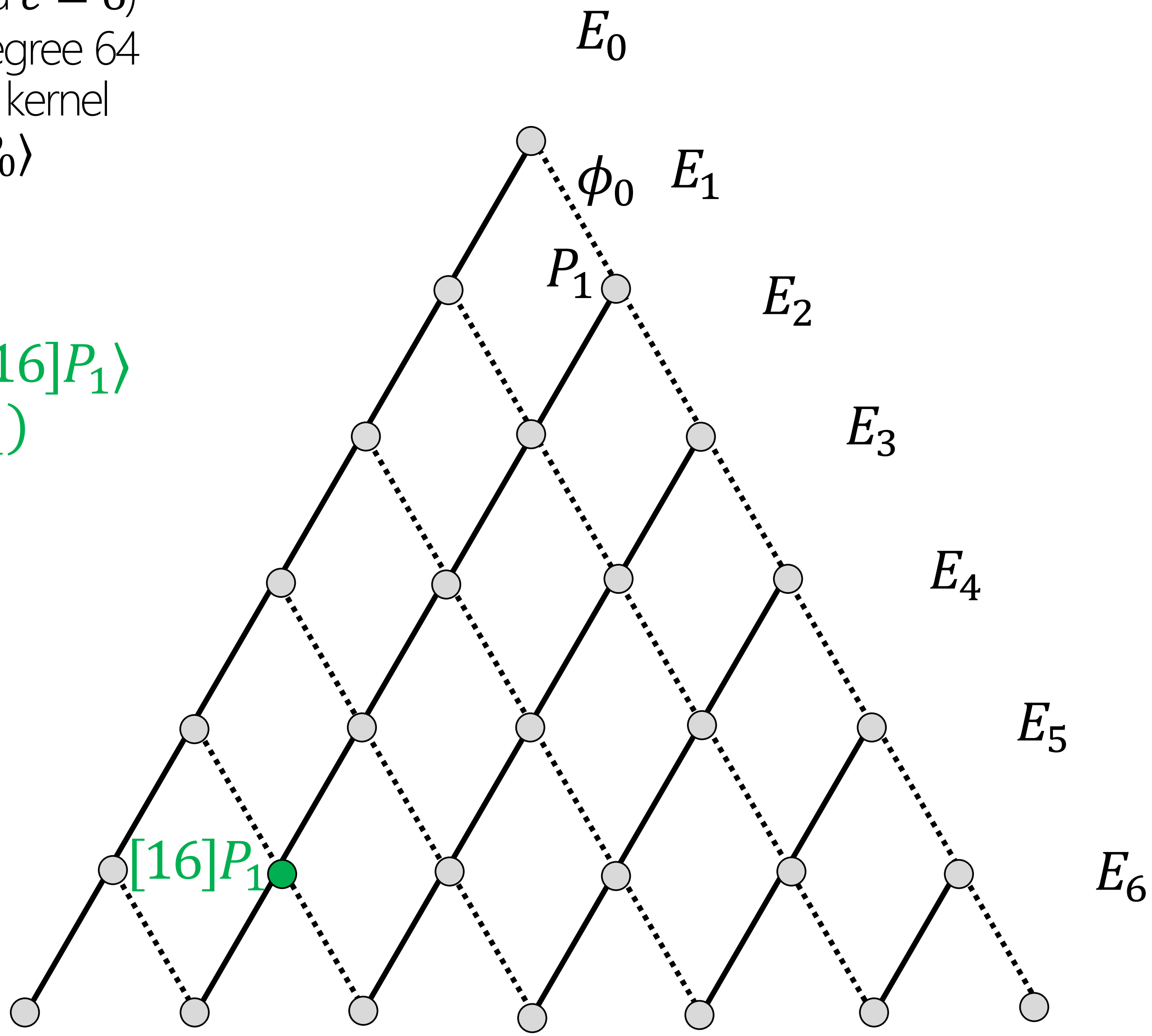
$E_3 = E_1/\langle [8]P_1 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi: E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_2 = E_1/\langle [16]P_1 \rangle$
$= \phi_1(E_1)$

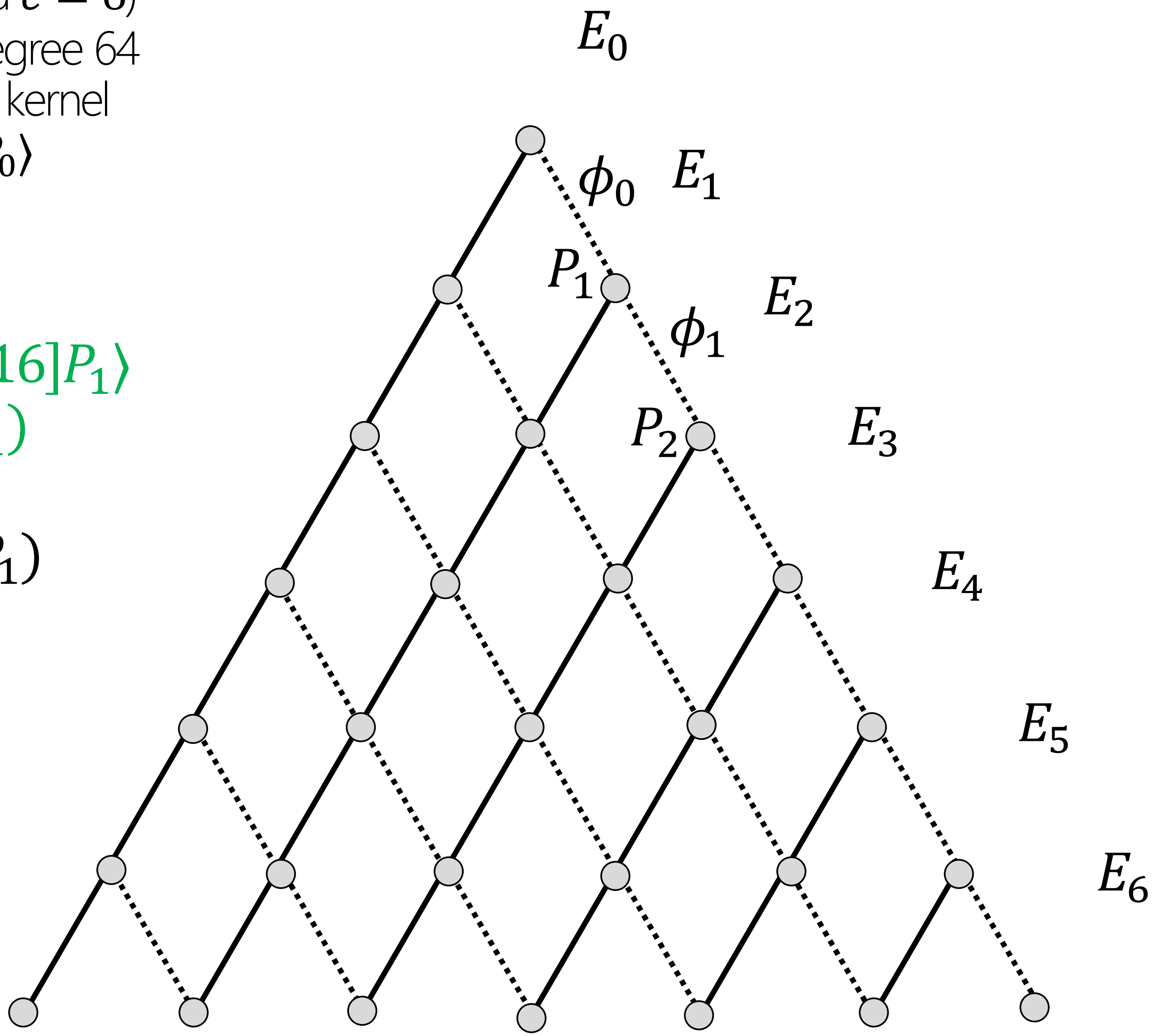


Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$$E_2 = E_1/\langle [16]P_1 \rangle = \phi_1(E_1)$$

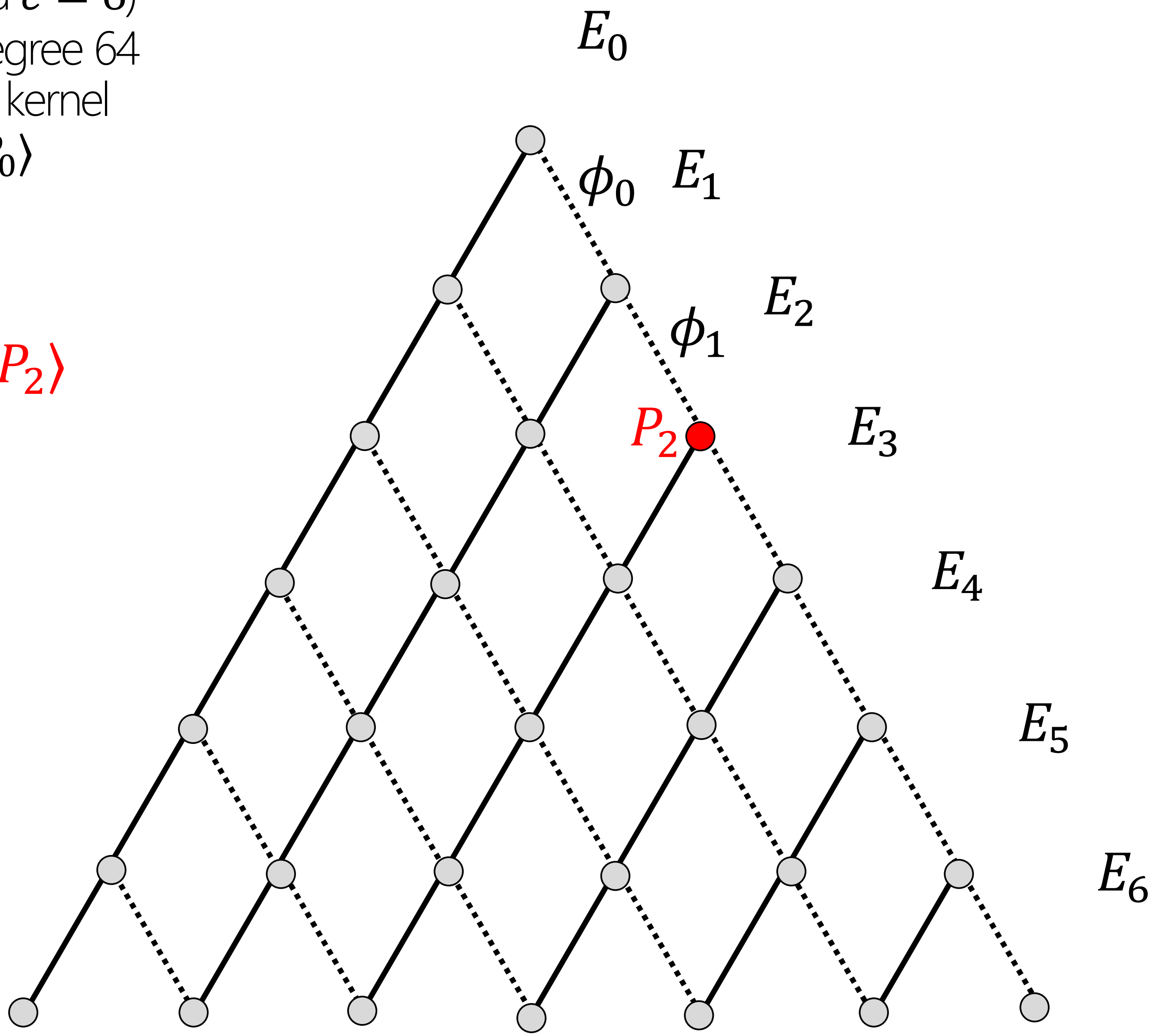
$$P_2 = \phi_1(P_1)$$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi: \; E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

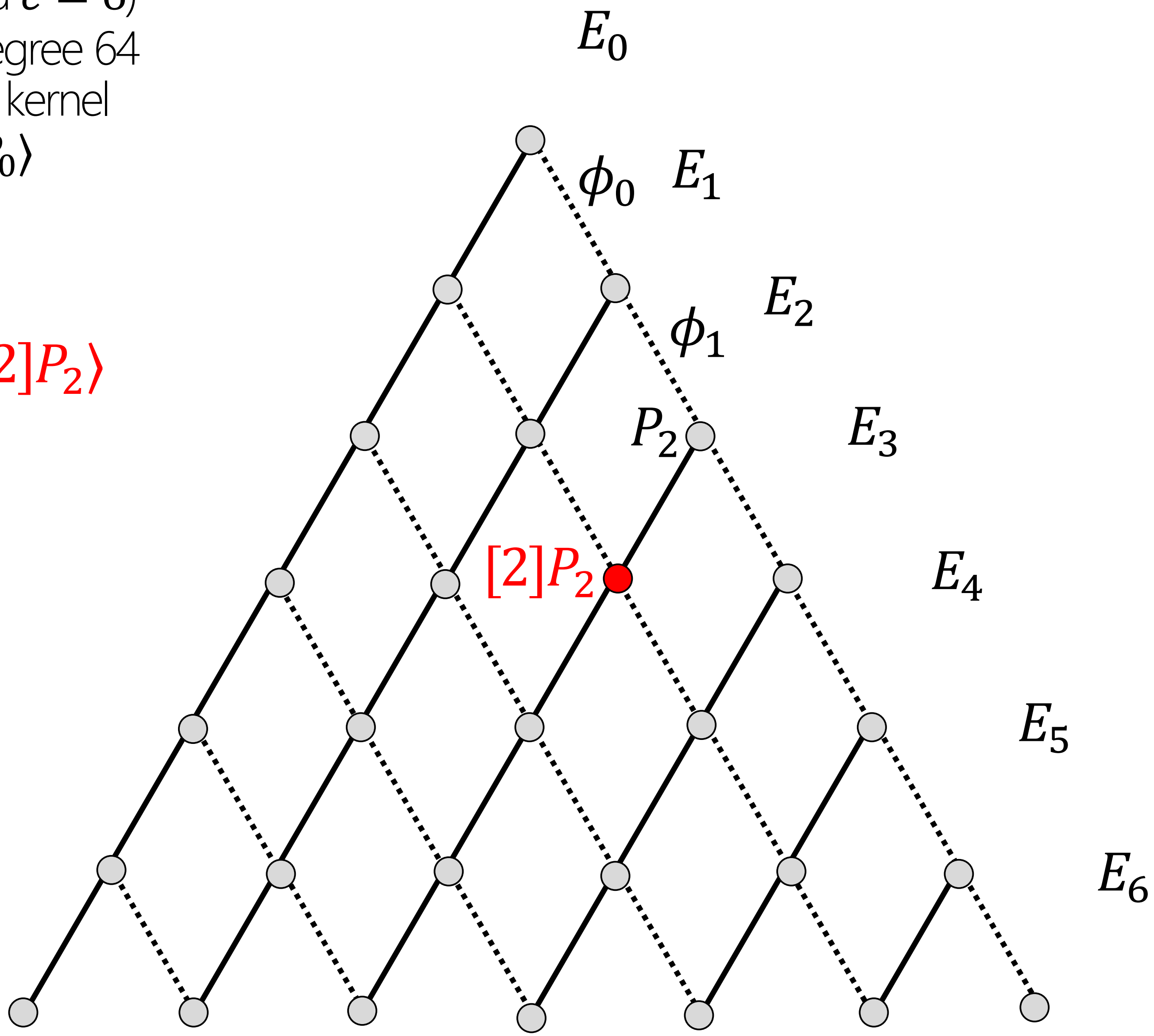
$E_6 = E_2/\langle P_2 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

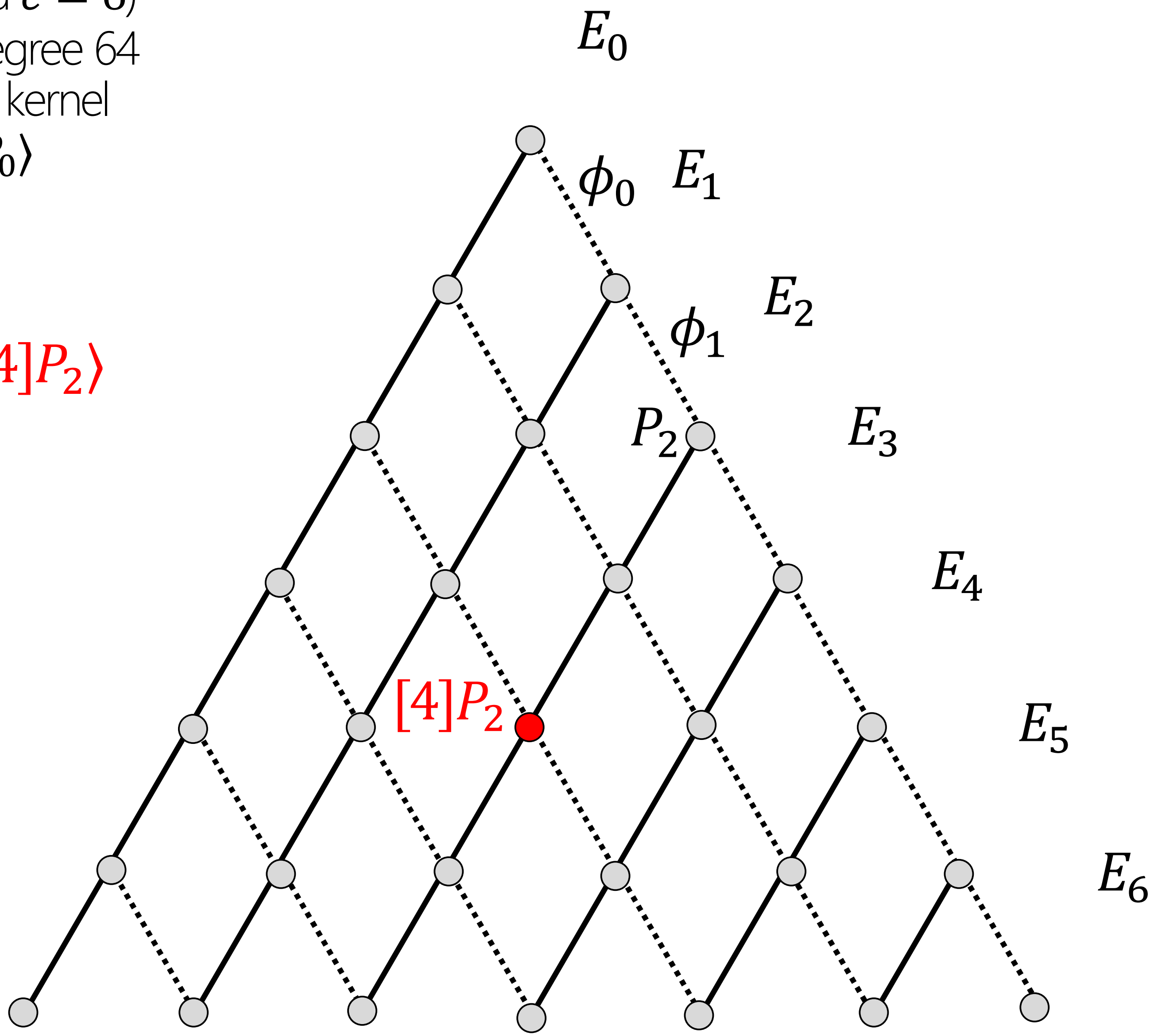
$E_5 = E_2/\langle [2]P_2 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

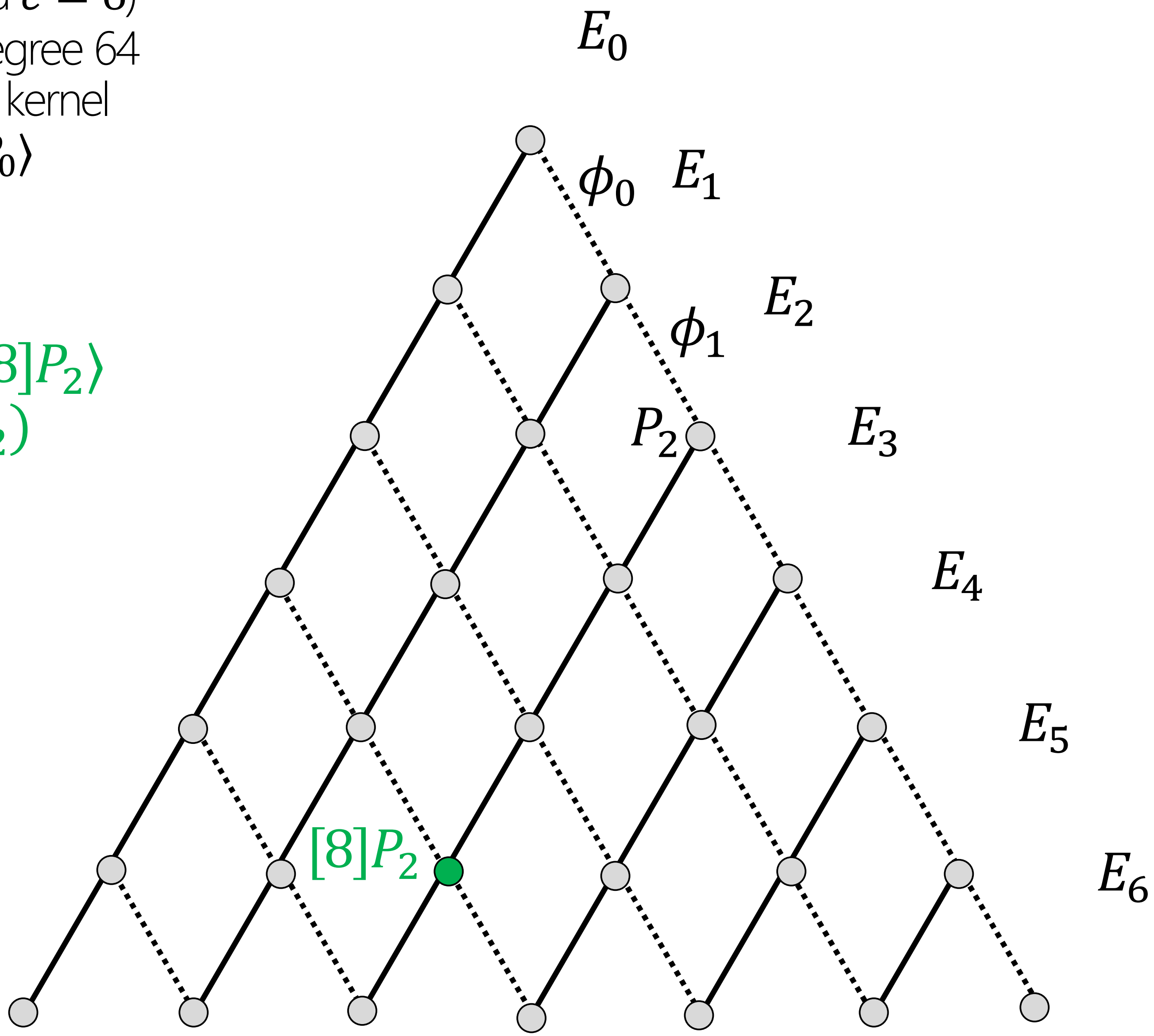
$E_4 = E_2 / \langle [4]P_2 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_3 = E_2/\langle [8]P_2 \rangle$
$= \phi_2(E_2)$

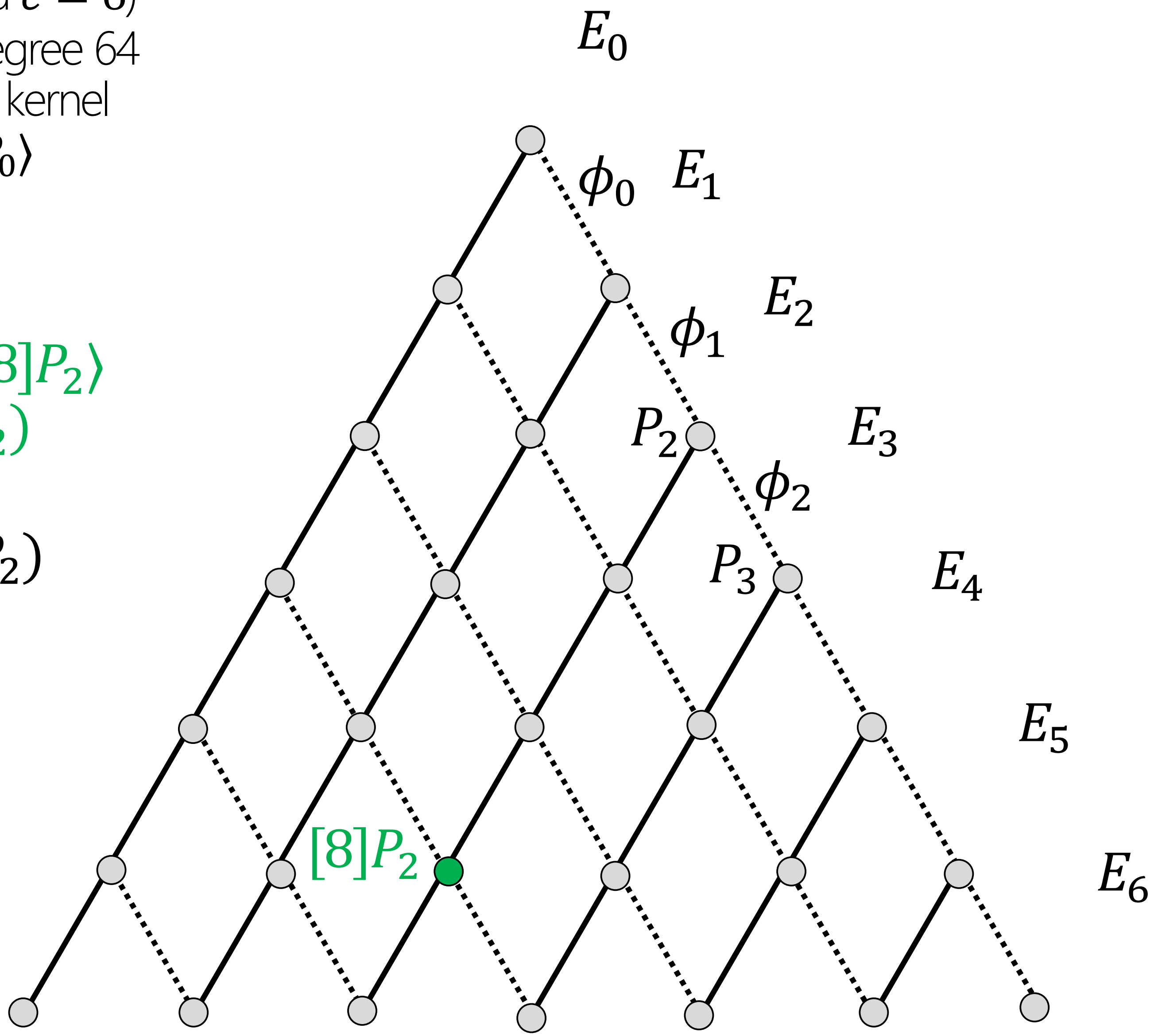


Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : \ E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_3 = E_2/\langle [8]P_2 \rangle$
$= \phi_2(E_2)$

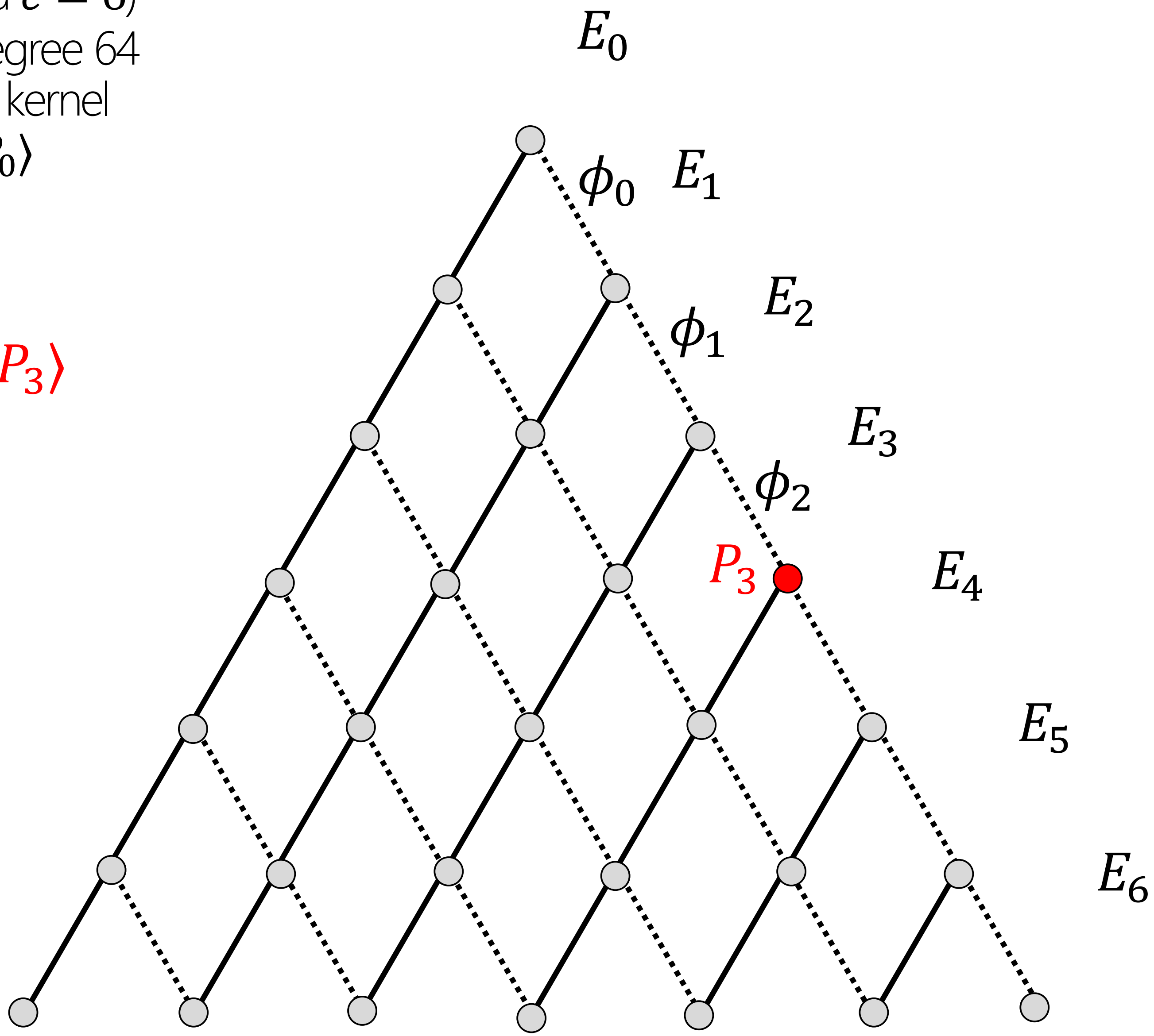
$P_3 = \phi_2(P_2)$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_6 = E_3 / \langle P_3 \rangle$



Computing ℓ^e degree isogenies

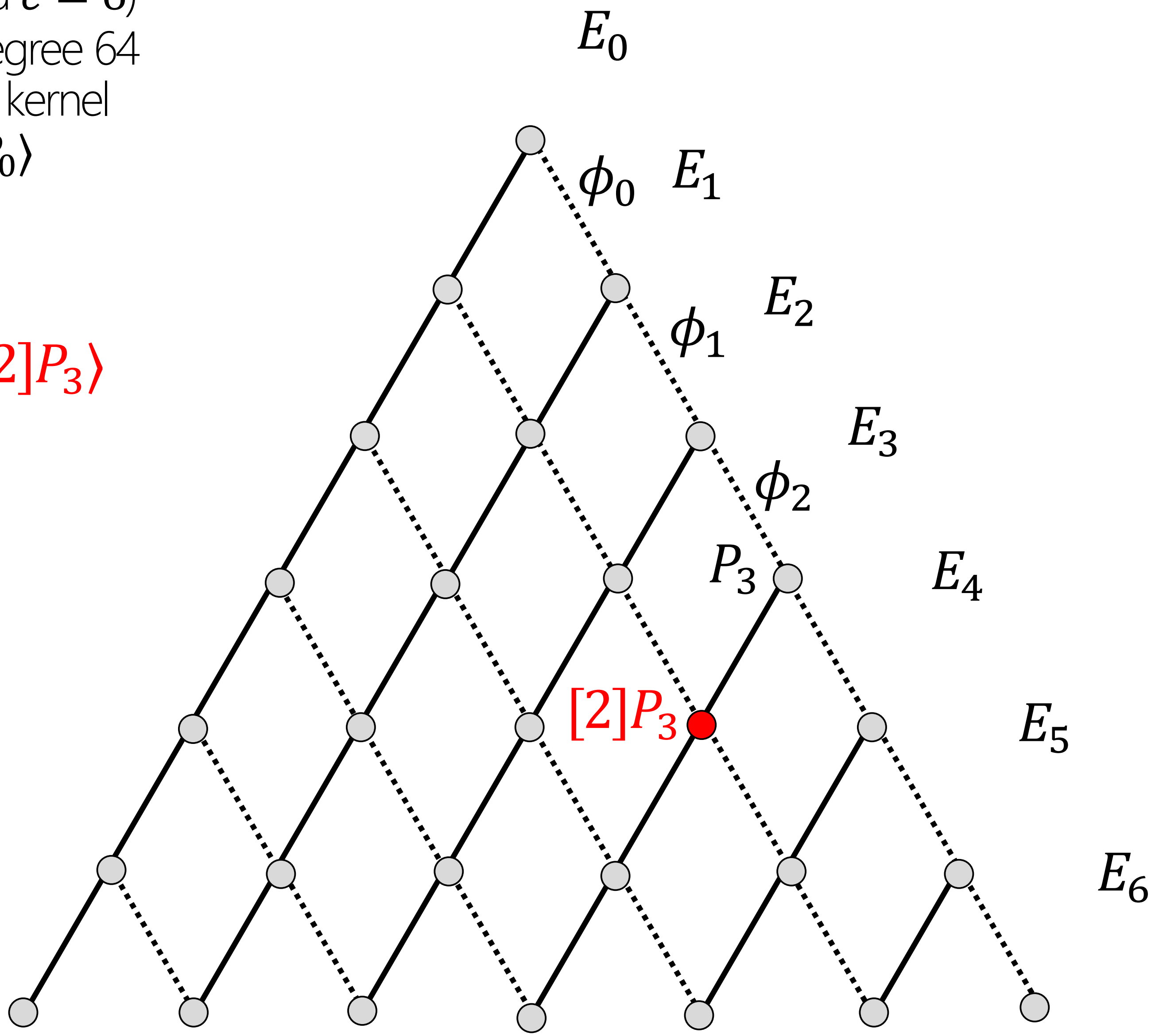
(suppose $\ell = 2$ and $e = 6$)

$\phi: E_0 \to E_6$ is degree 64

64 elements in its kernel

$\ker(\phi) = \langle P_0 \rangle$

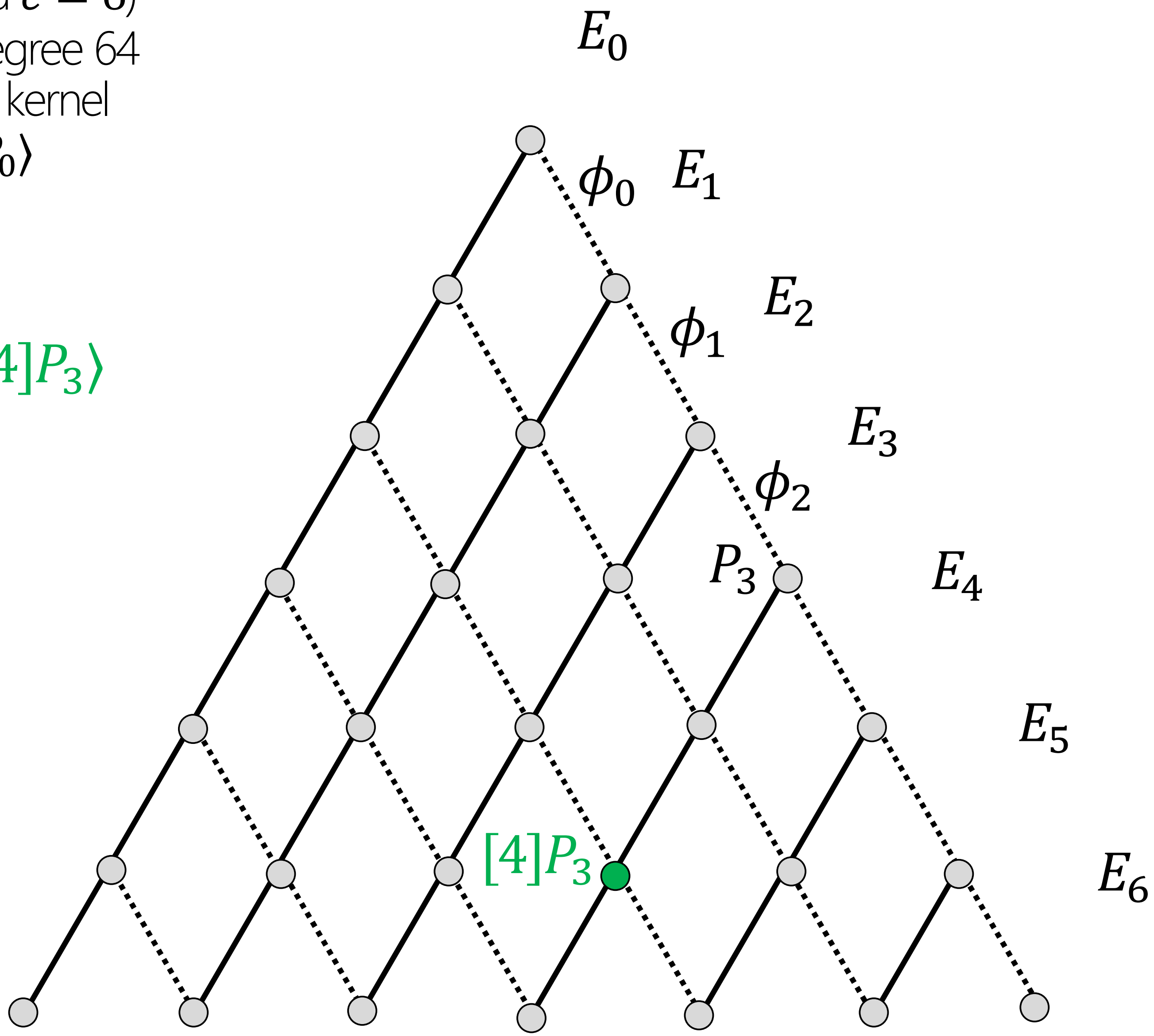
$E_5 = E_3 / \langle [2]P_3 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_4 = E_3/\langle [4]P_3 \rangle$

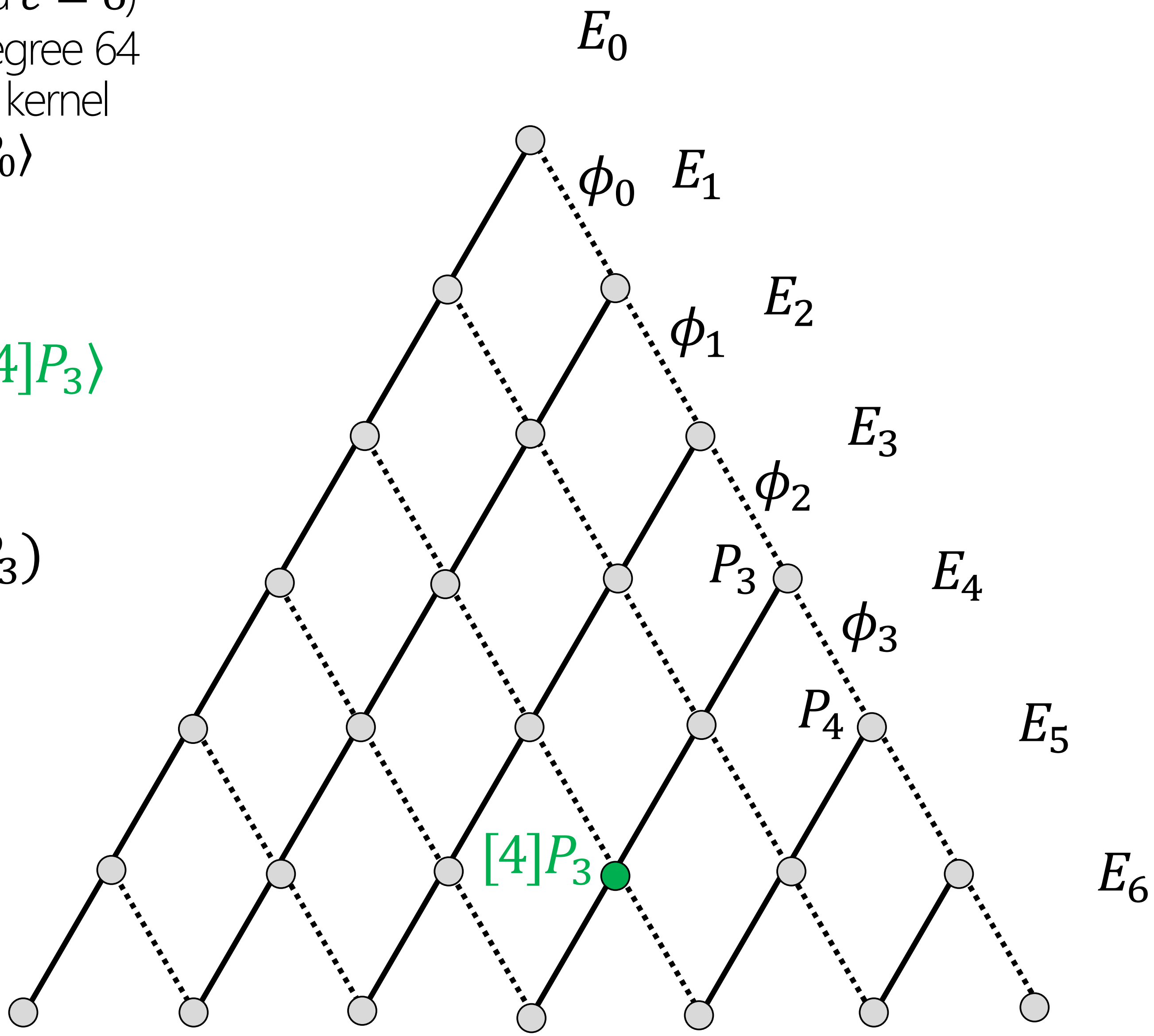


Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \rightarrow E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_4 = E_3/\langle [4]P_3 \rangle$

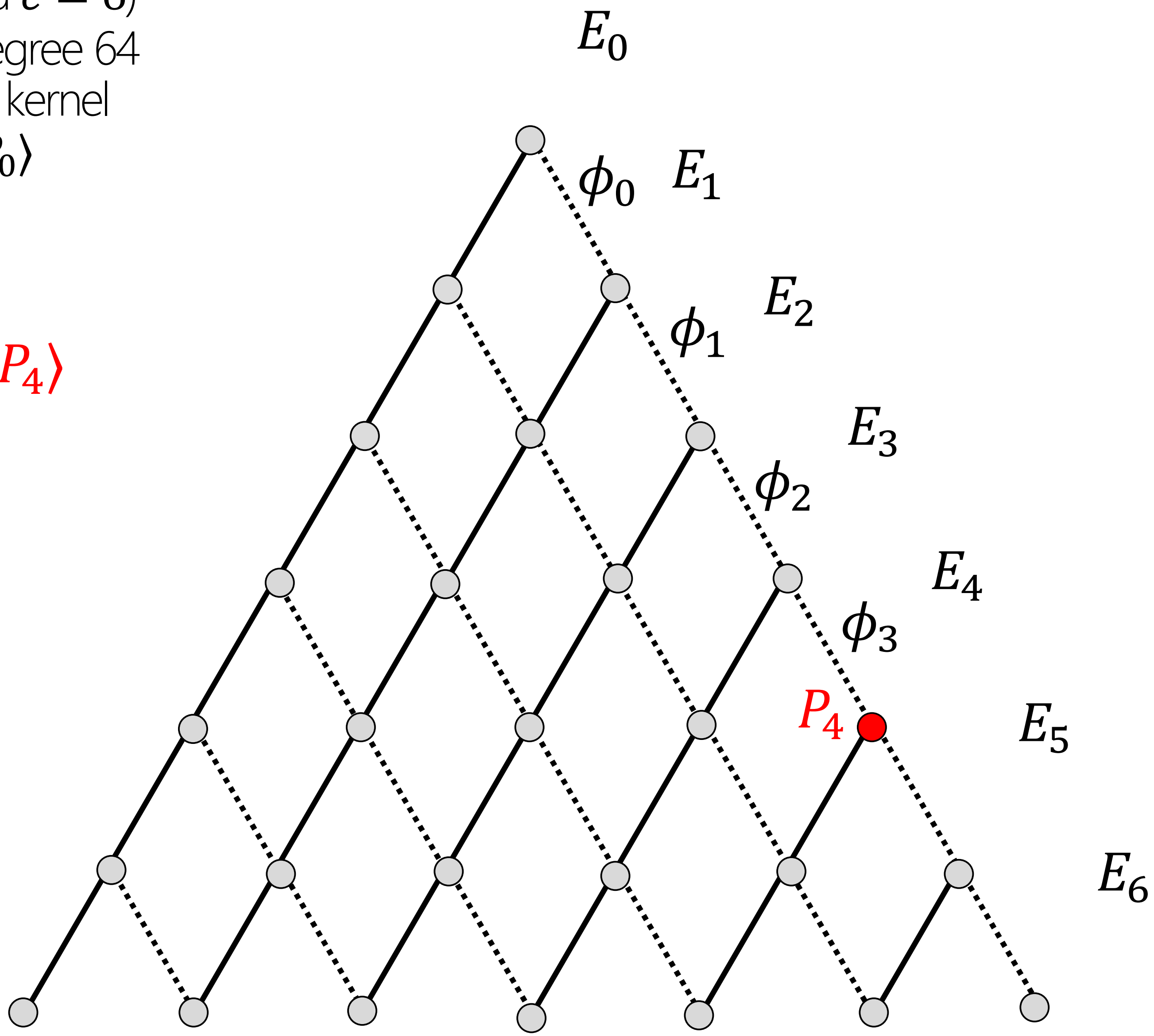
$P_4 = \phi_3(P_3)$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

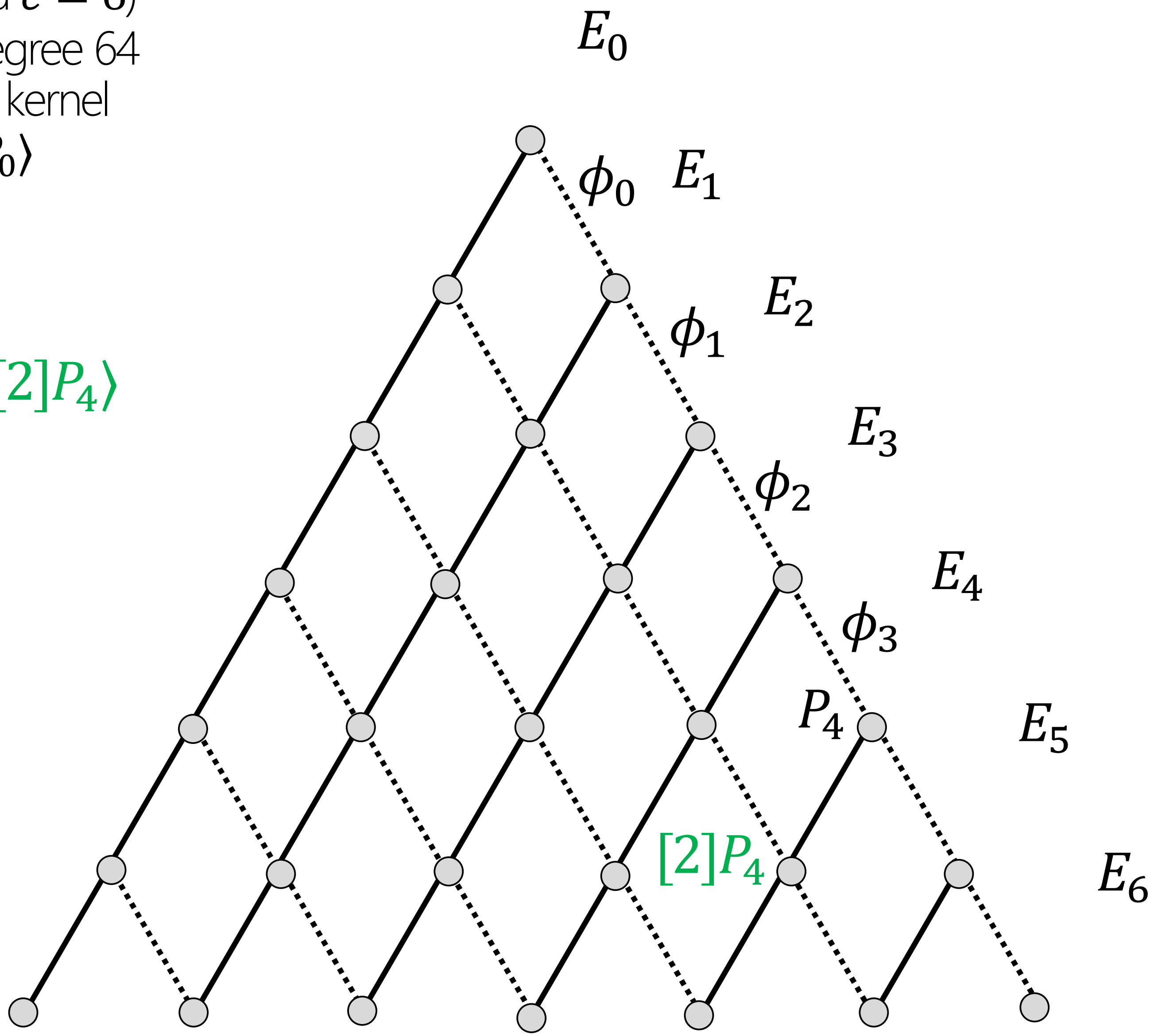
$E_6 = E_4 / \langle P_4 \rangle$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi: E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_5 = E_4 / \langle [2]P_4 \rangle$

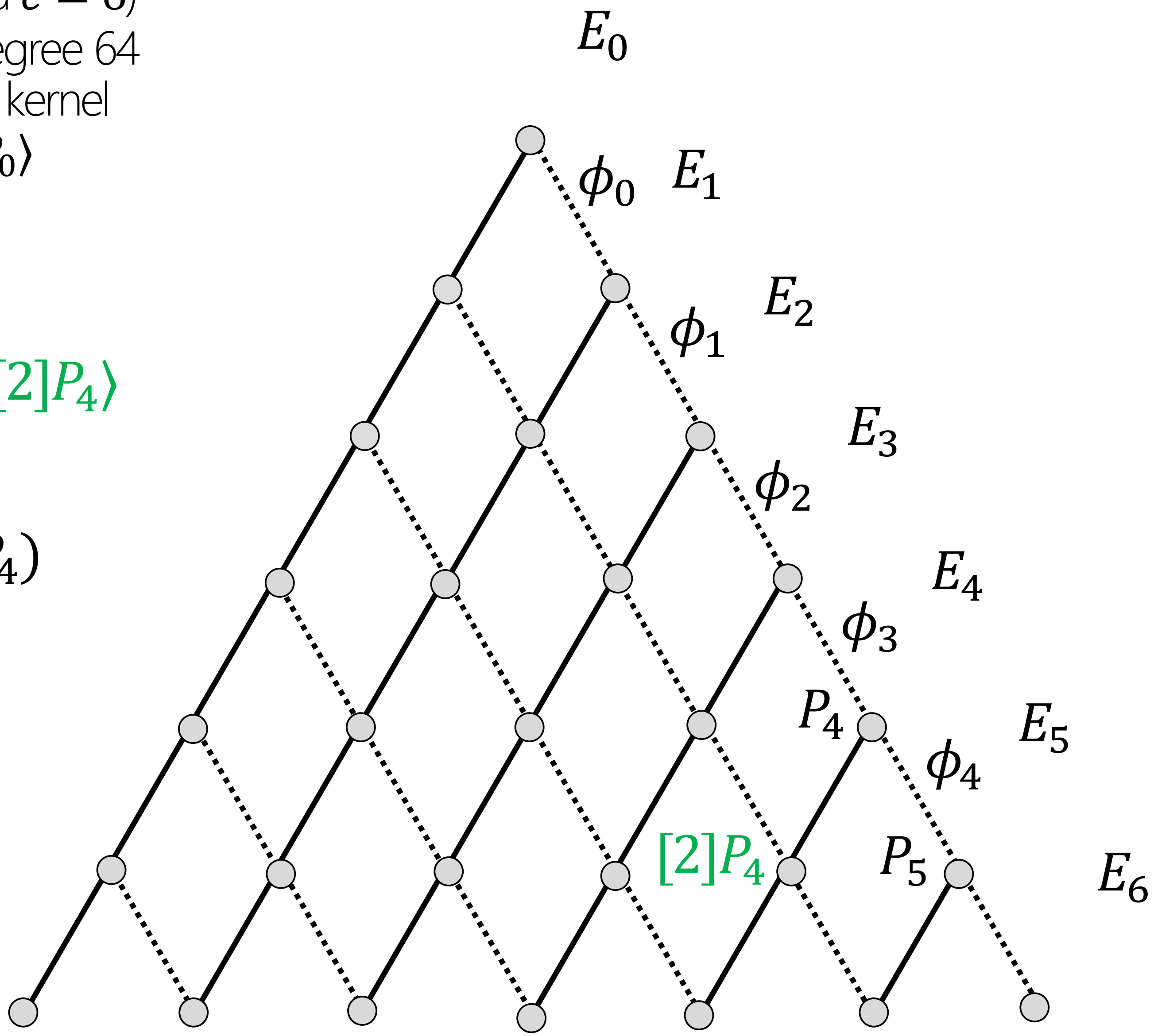


Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : \; E_0 \rightarrow E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

$E_5 = E_4 / \langle [2] P_4 \rangle$

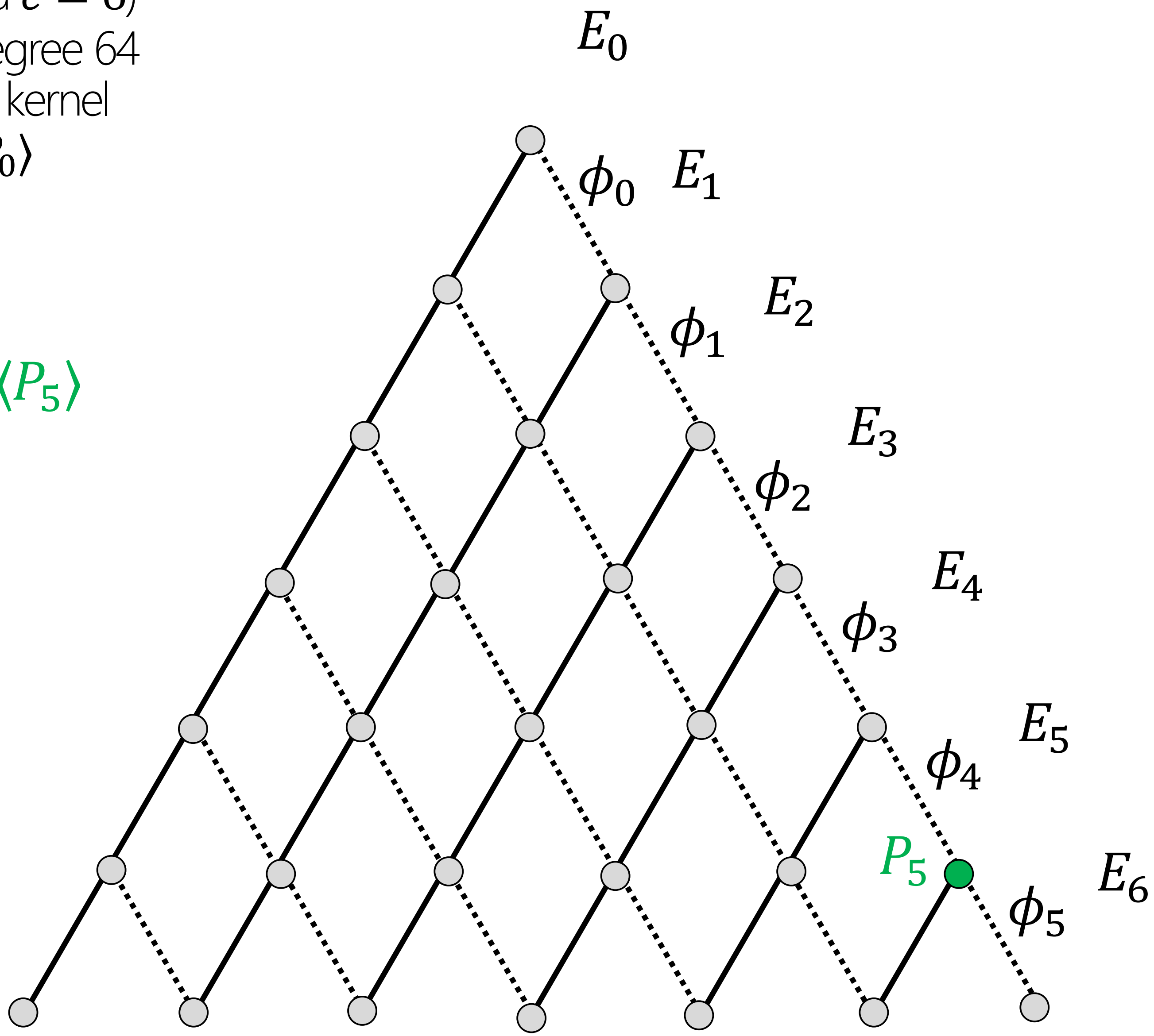
$P_5 = \phi_4(P_4)$



Computing ℓ^e degree isogenies

(suppose $\ell = 2$ and $e = 6$)
$\phi : \; E_0 \to E_6$ is degree 64
64 elements in its kernel
$\ker(\phi) = \langle P_0 \rangle$

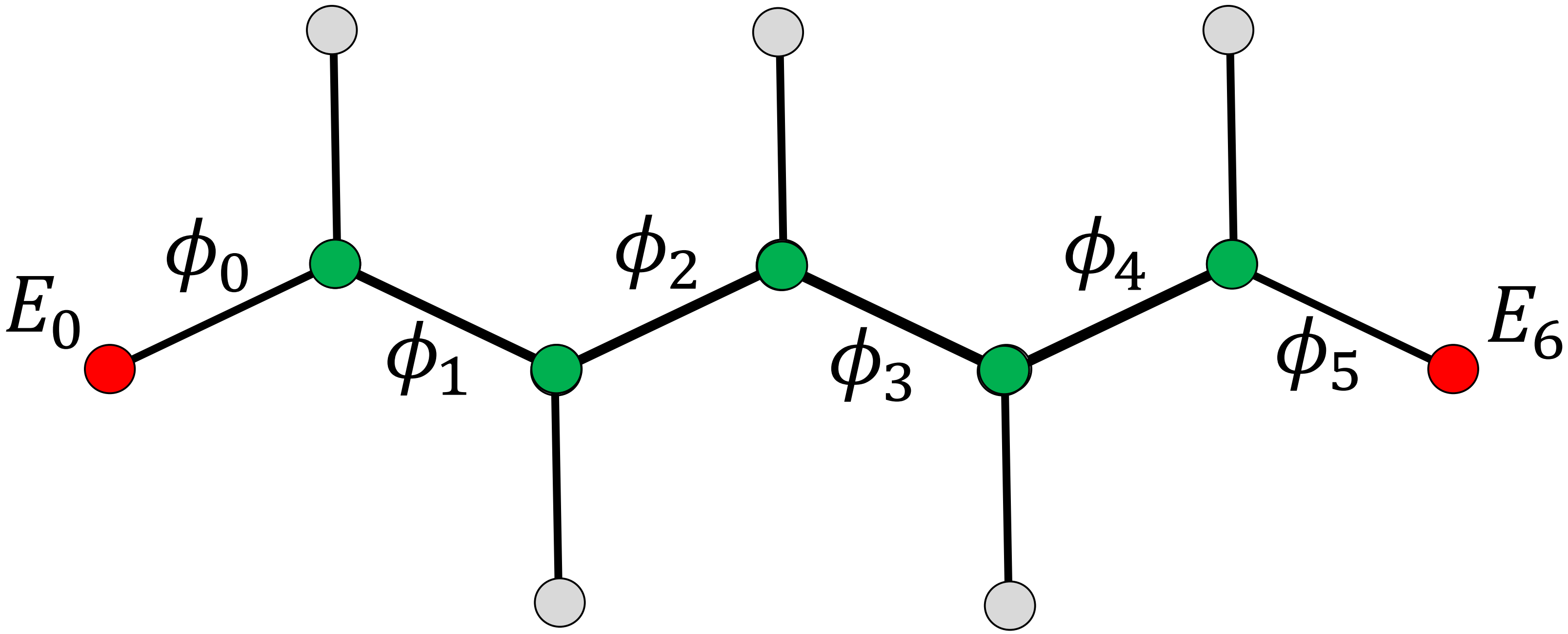
$E_6 = E_5 / \langle P_5 \rangle$



Computing ℓ^e degree isogenies

$$\phi \ : \ E_0 \to E_6$$

$$\phi = \phi_5 \circ \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1 \circ \phi_0$$




E
?
E'

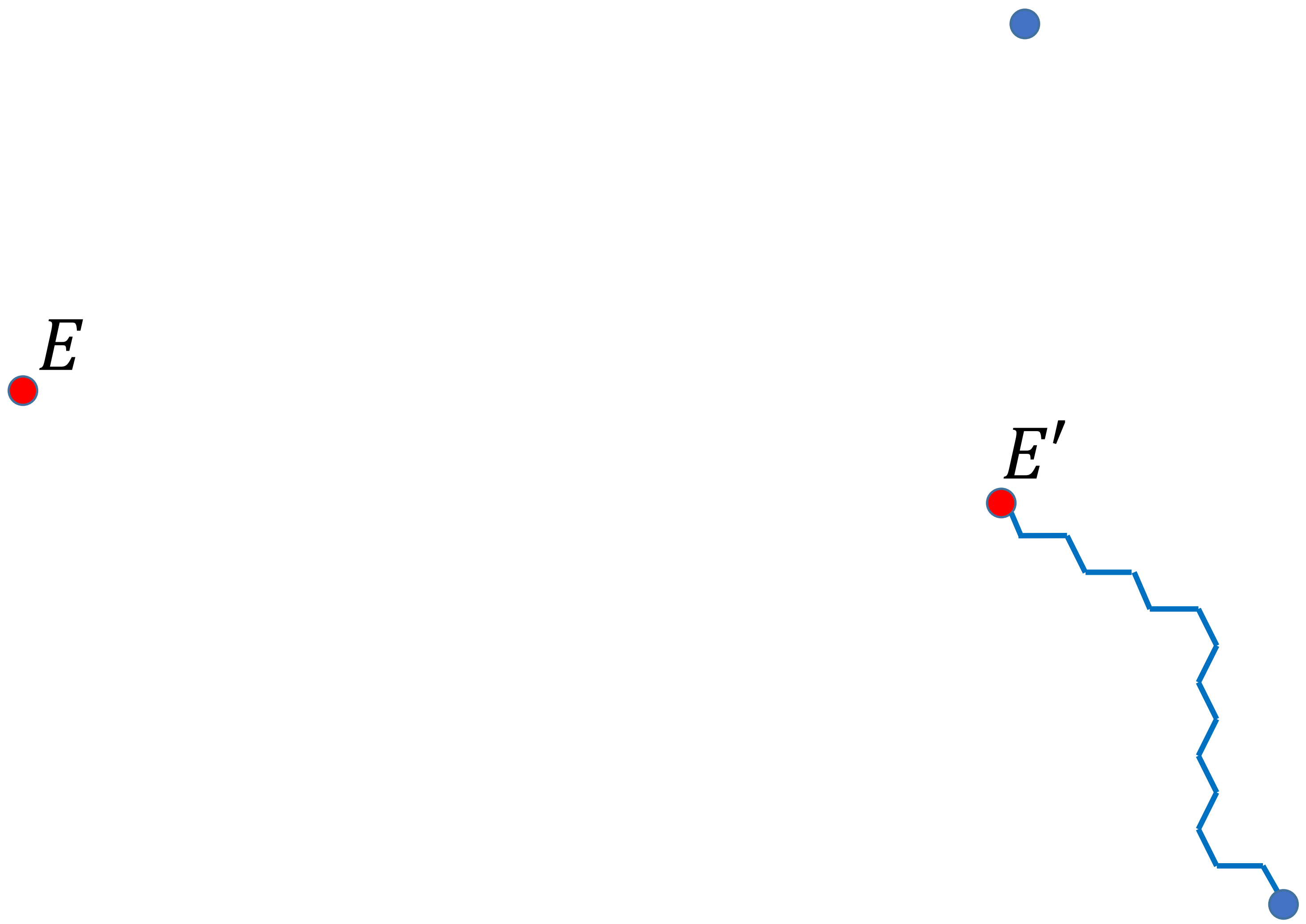
Claw algorithm

Given E and $E' = \phi(E)$, with ϕ degree ℓ^e, find ϕ

Claw algorithm

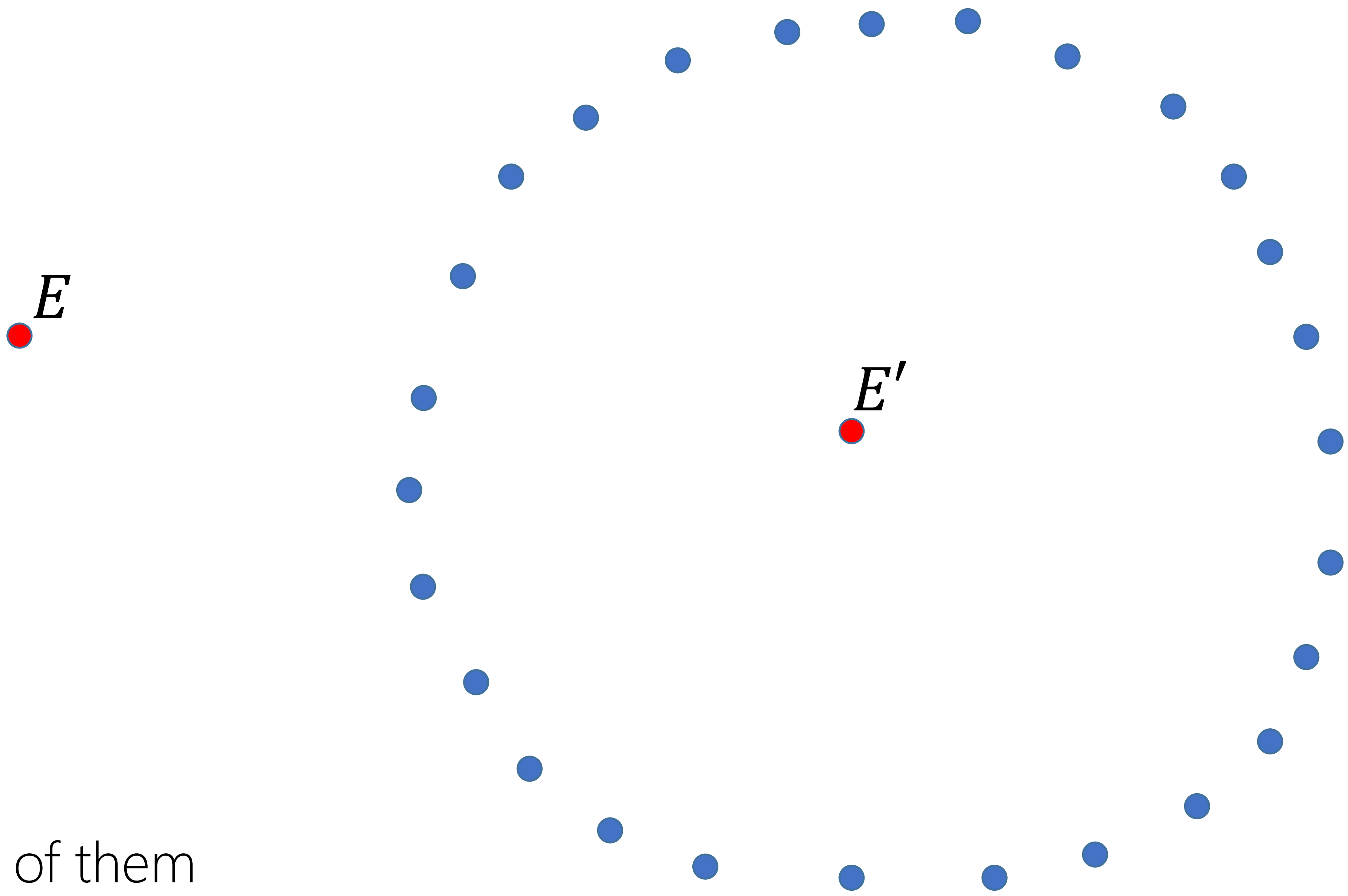
Compute and store $\ell^{e/2}$-isogenies on one side

Claw algorithm



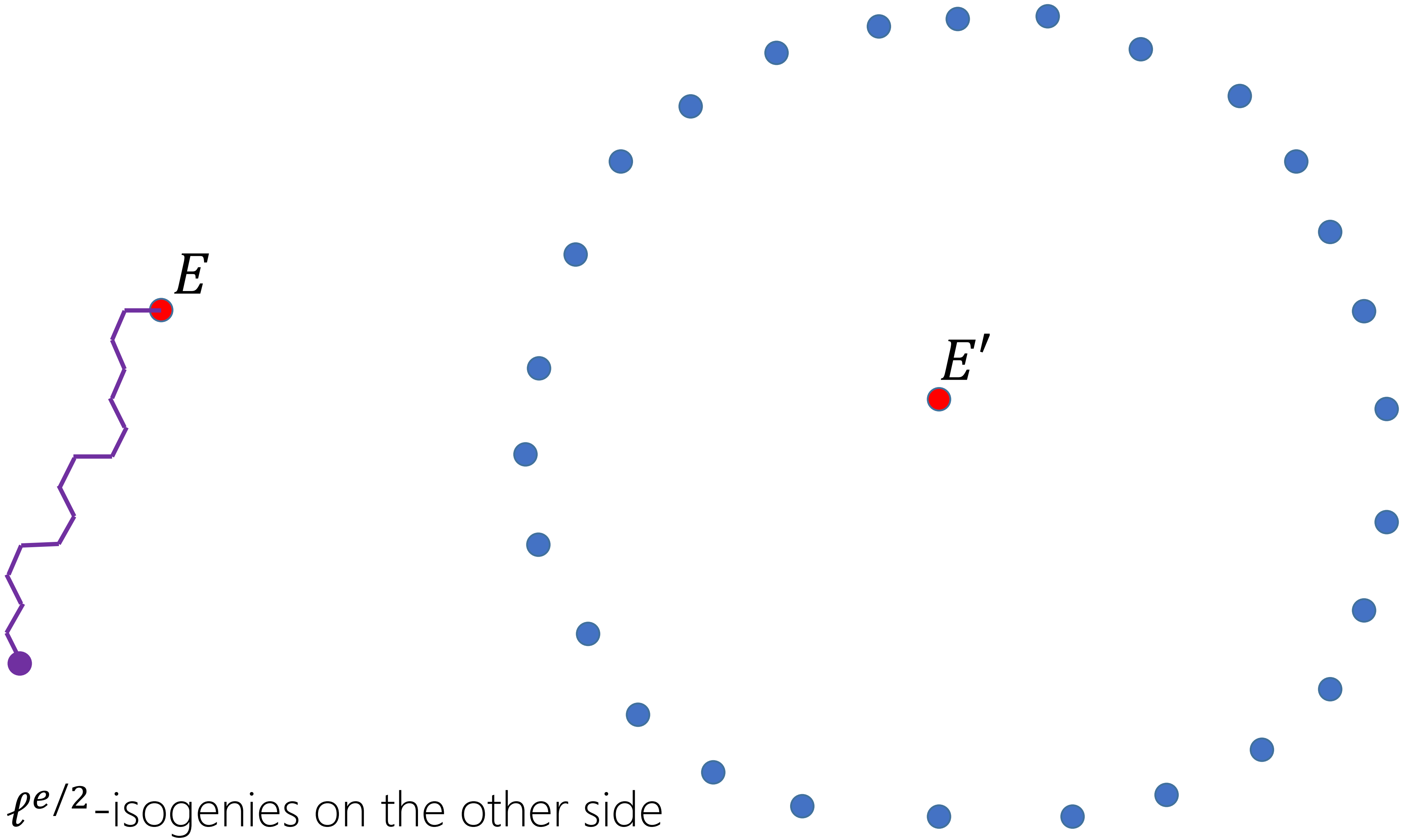
Compute and store $\ell^{e/2}$-isogenies on one side

Claw algorithm



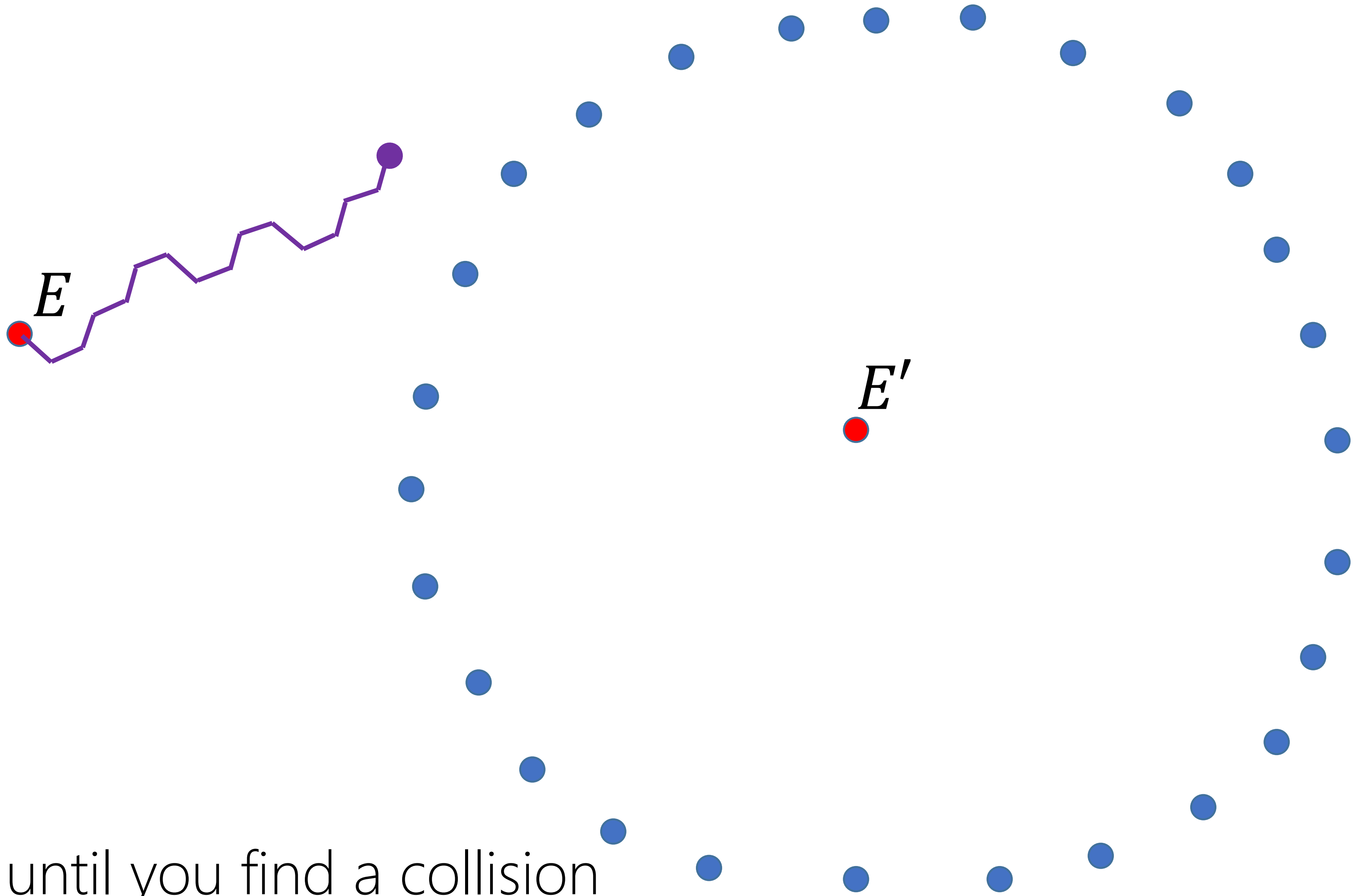
… until you have all of them

Claw algorithm



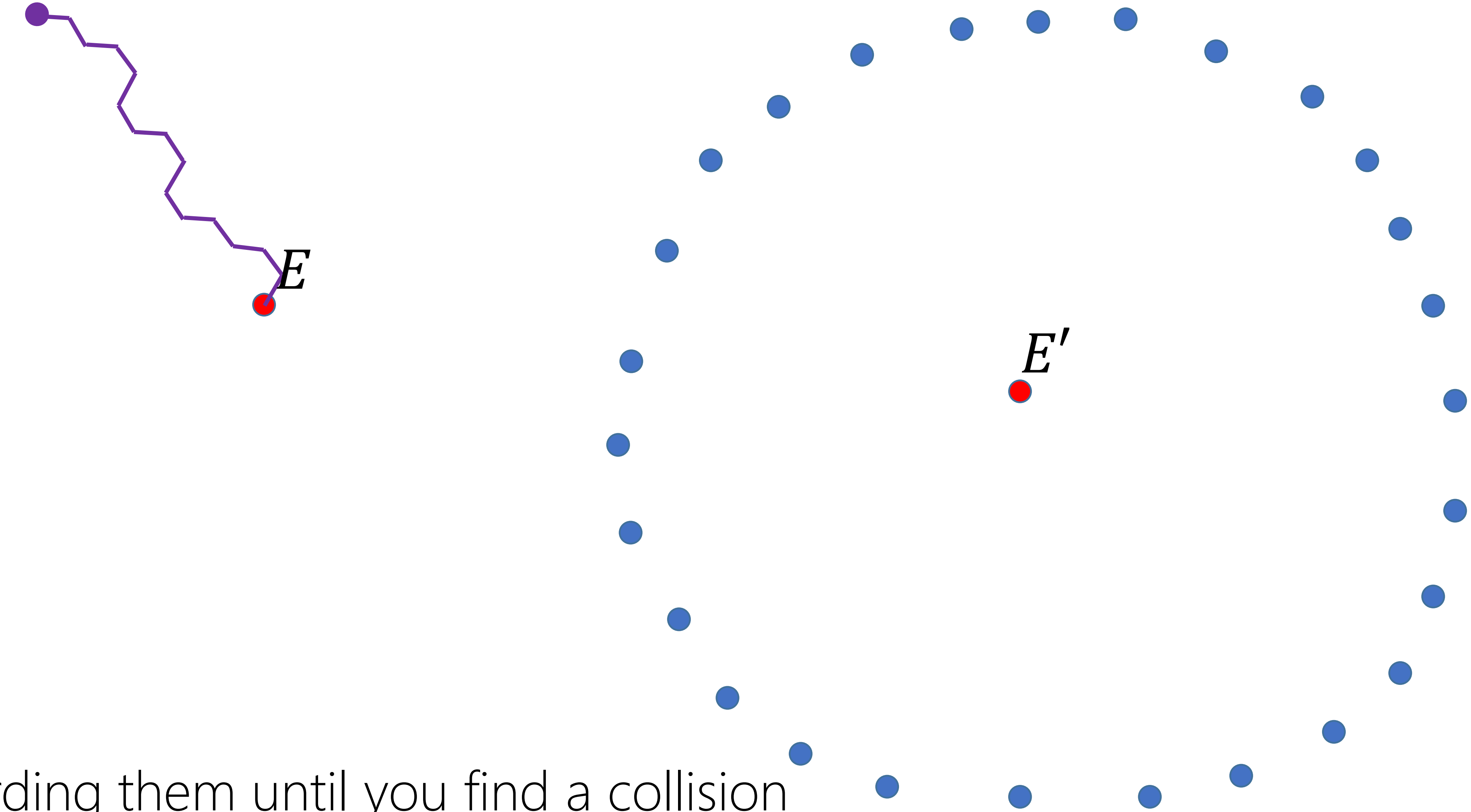
Now compute $\ell^{e/2}$-isogenies on the other side

Claw algorithm



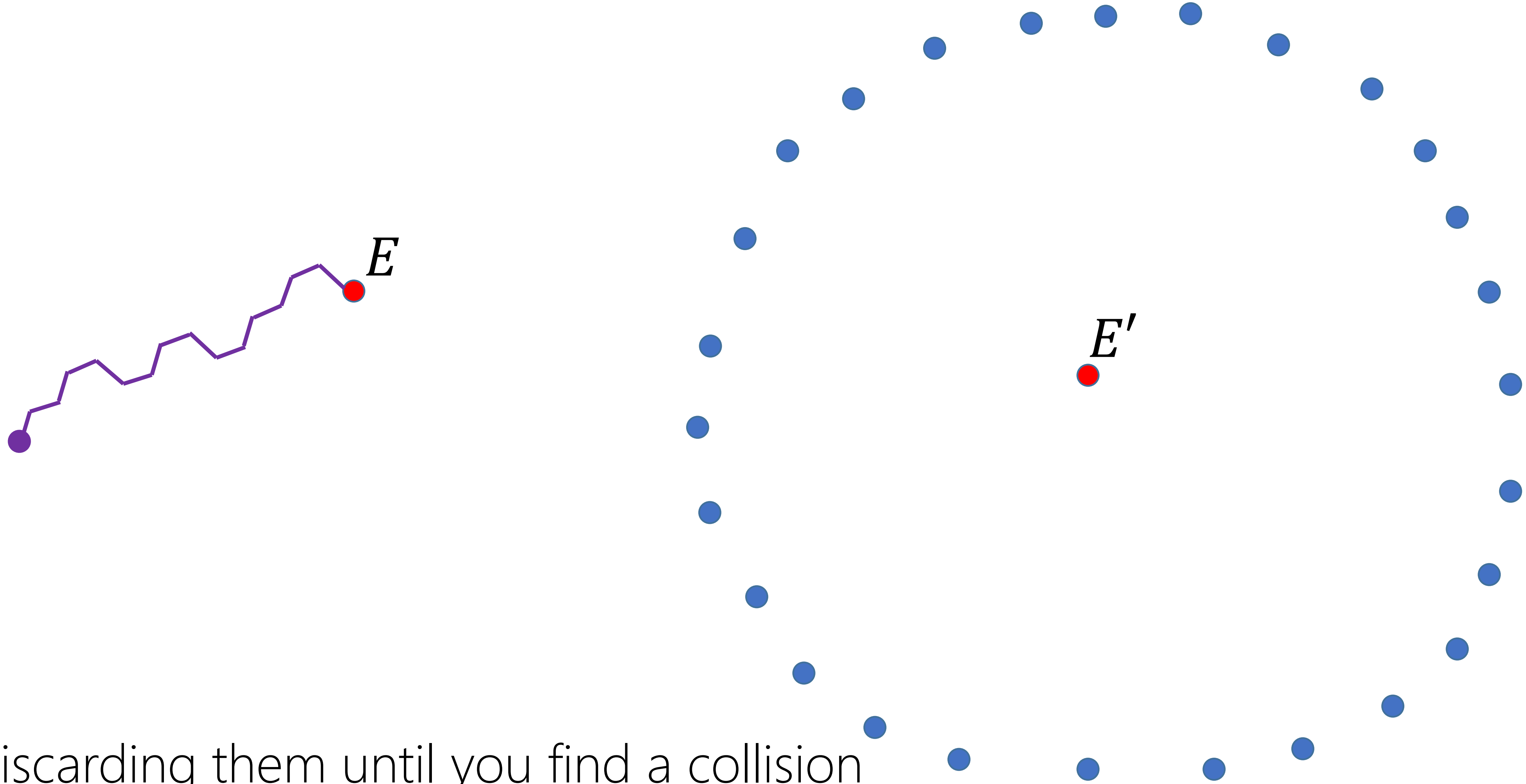
… discarding them until you find a collision

Claw algorithm



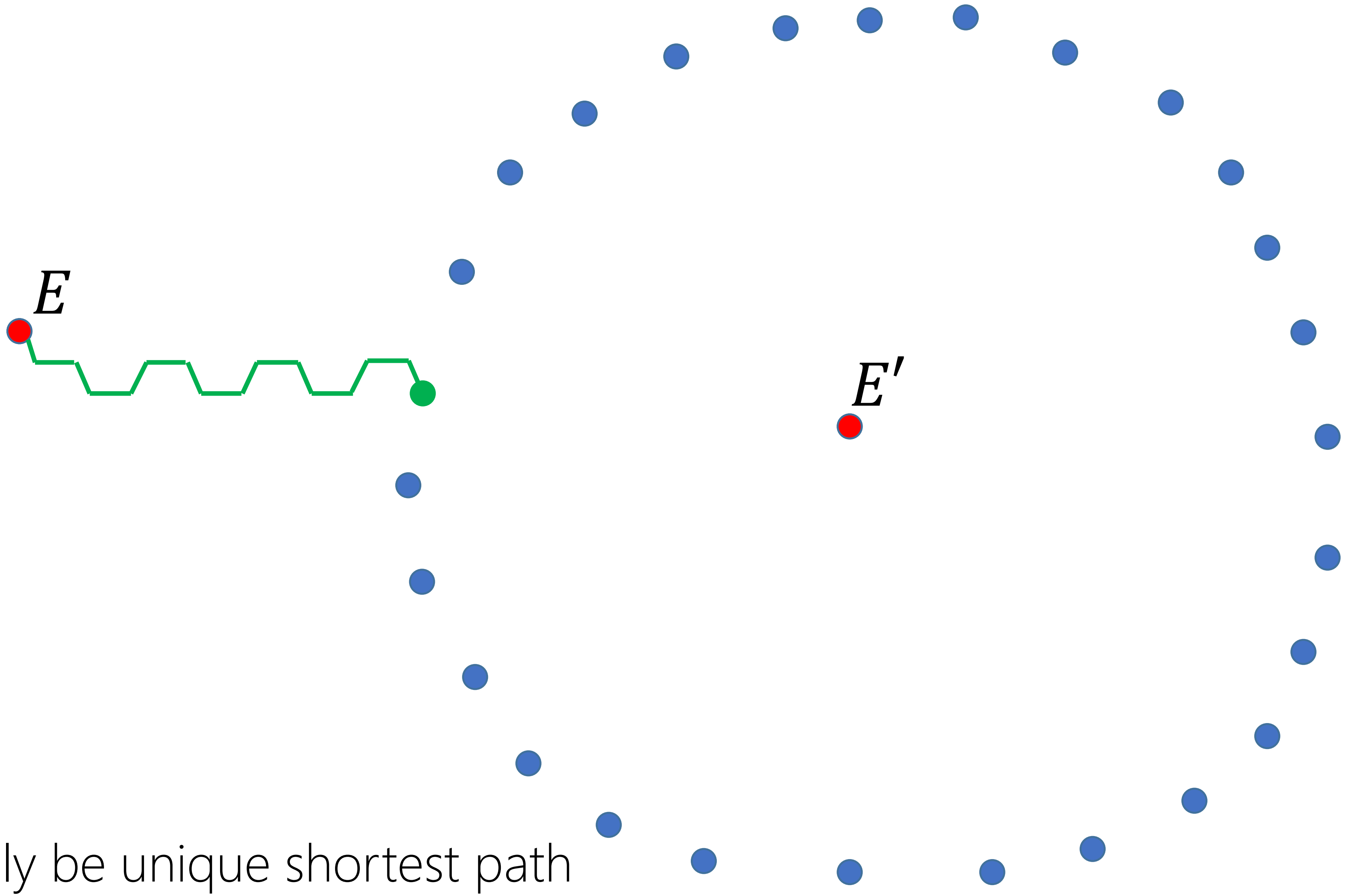
… discarding them until you find a collision

Claw algorithm



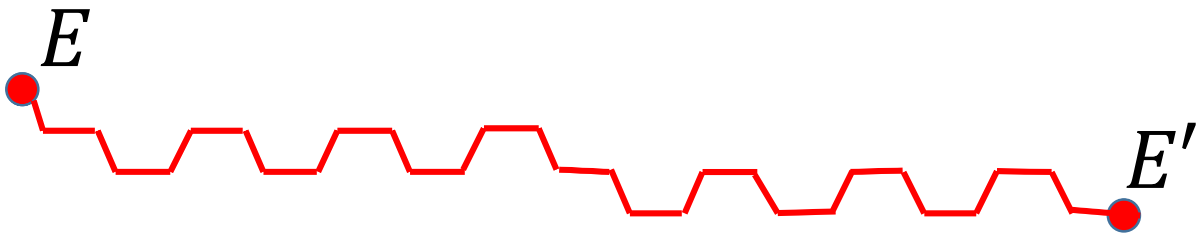
... discarding them until you find a collision

Claw algorithm



Collision will most likely be unique shortest path

Claw algorithm



This path describes secret isogeny $\phi : E \to E'$

Claw algorithm: classical analysis

- There are $O(\ell^{e/2})$ curves $\ell^{e/2}$-isogenous to E' (the blue nodes)

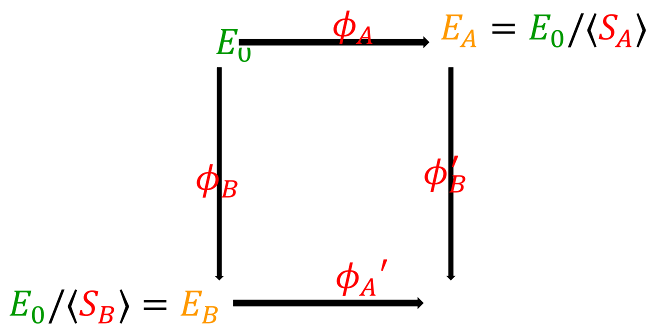
  thus $O(\ell^{e/2}) = O(p^{1/4})$ classical memory

- There are $O(\ell^{e/2})$ curves $\ell^{e/2}$-isogenous to E' (the blue nodes), and there are $O(\ell^{e/2})$ curves $\ell^{e/2}$-isogenous to E (the purple nodes)

  thus $O(\ell^{e/2}) = O(p^{1/4})$ classical time

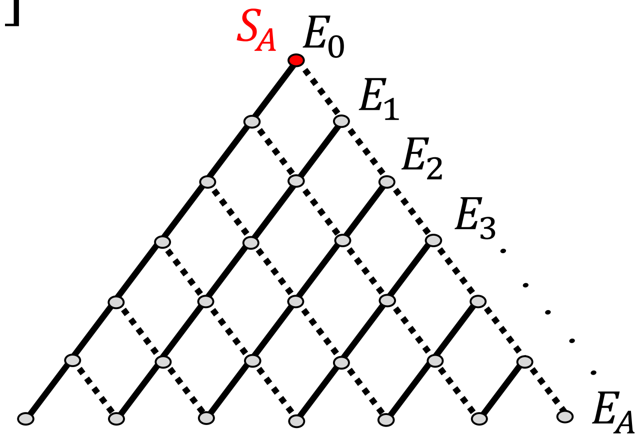
- **Best (known) attacks:** classical $O(p^{1/4})$ and quantum $O(p^{1/6})$
- **Confidence:** both complexities are optimal for a black-box claw attack

SIDH protocol summary



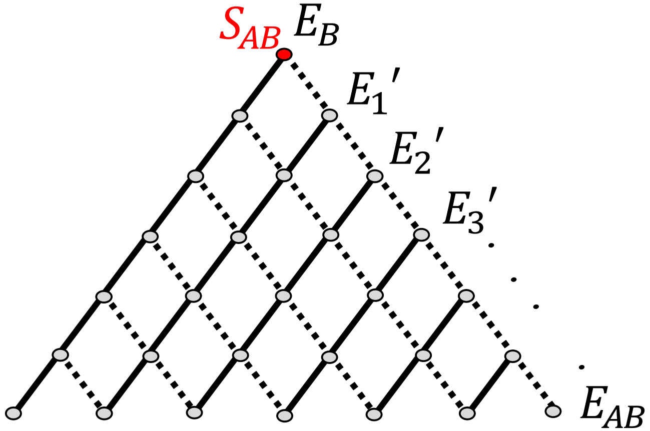
- **Setting:** supersingular elliptic curves $E/\mathbb{F}_{p^2}$ where $p = 2^i 3^j - 1$
- **Parameters:**

$$E_0/\mathbb{F}_{p^2} : y^3 = x^3 + x \quad \text{with} \quad \#E_0 = \left(2^i 3^j\right)^2$$
$$P_A, Q_A \in E_0\left[2^i\right] \text{ and } P_B, Q_B \in E_0[3^j]$$



- Public key generation (Alice):

$$s \in \left[0, 2^i\right)$$
$$S_A = P_A + [s]Q_A$$
$$\phi_A : \ E_0 \to E_A := E_0/\langle S_A \rangle$$
send $E_A, \phi_A(P_B), \phi_A(Q_B)$ to Bob



- Shared key generation (Alice):

$$S_{AB} = \phi_B(P_A) + [s]\phi_B(Q_A) \in E_B$$
$$\phi_{A'} : \ E_B \to E_{AB} := E_B/\langle S_{AB} \rangle$$
$$j_{AB} = j(E_{AB})$$

SIDH security summary

- **Setting:** supersingular elliptic curves $E/\mathbb{F}_{p^2}$ where p is a large prime

- **Hard problem**: Given $P, Q \in E$ and $\phi(P), \phi(Q) \in \phi(E)$, compute ϕ
  (where ϕ has fixed, smooth, public degree)

- **Best (known) attacks:** classical $O(p^{1/4})$ and quantum $O(p^{1/6})$

Part 1: Quick re-motivation

Part 2: Quick tutorial recap

Part 3: SIKE

"The poor user is given enough rope with which to hang himself – something a standard should not do."

- Ron Rivest, 1992 (on DSA standard)

public key compression

Point *and* isogeny arithmetic in $\mathbb{P}^1$

ECDH: move around different points on a fixed curve.

SIDH: move around different points and different curves

$$E_{a,b} \ : \quad by^2 = x^3 + ax^2 + x$$

$(x, y) \leftrightarrow (X : Y : Z)$ $\quad$ $(a, b) \leftrightarrow (A : B : C)$

$$E_{\frac{A}{C},\frac{B}{C}}: \quad BY^2Z = CX^3 + AX^2Z + CXZ^2$$

B coefficient only fixes the quadratic twist, but $j(E) = j(E')$

$\mathbb{P}^1$ point arithmetic: $\quad (X : Z) \mapsto (X' : Z')$

$\mathbb{P}^1$ isogeny arithmetic: $\quad (A : C) \mapsto (A' : C')$

Point *and* isogeny arithmetic in $\mathbb{P}^1$

$$\phi_3 : \quad E_{a,b} \to E_{a',b'}$$

$$(x,y) \mapsto \left(x \cdot \left(\frac{x \cdot x_3 - 1}{x - x_3} \right)^2, \frac{(x \cdot x_3 - 1)\left(x^2 \cdot x_3 - 3x \cdot x_3^2 + x + x_3\right)}{(x - x_3)^3} \right)$$

$$(a', b') = \left(\left(a \cdot x_3 - 6x_3^2 + 6 \right) \cdot x_3, b \cdot x_3^2 \right)$$

$$\phi_3 : \quad E_{A/C\,,B/C}/\{\pm 1\} \to E_{A'/C',B'/C'}/\{\pm 1\}$$

$$(X : Z) \quad \mapsto \quad \left(X(X_3 X - Z_3 Z)^2 \quad : \quad Z(Z_3 X - X_3 Z)^2 \right)$$

$$(A' : C') \; = \; \left(Z_3^4 + 18 X_3^2 Z_3^2 - 27 X_3^2 \quad : \quad 4 X_3 Z_3^3 \right)$$

Public keys are in $\mathbb{F}_{p^2}^3$

$$PK_A = \left(x_{\phi_A(P_B)}\ , x_{\phi_A(Q_B)}\ , x_{\phi_A(Q_B - P_B)} \right)$$

Conversely, if $R = \pm(Q - P)$ on $E_a :\ \ y^2 = x^3 + ax^2 + x$, then

$$a = \frac{\left(1 - x_P x_Q - x_P x_R - x_Q x_R\right)^2}{4 x_P x_Q x_R} - x_P - x_Q - x_R$$

The starting curve

$$E_0 \; : \quad y^2 = x^3 + x$$

Computing $\phi : E_0 \to E'$ is broadly equivalent to computing $\mathbf{End}(E')$

(see Kohel's thesis, Galbraith-Vercauteren survey, Galbraith-Petit-Shani-Ti)

Computing $\phi : E_0 \to E'$ is subexponential if E' is defined over $\mathbb{F}_p$

(see Biasse-Jao-Sankar, Galbraith-Delfs)

Known security not damaged, but perhaps we'd prefer to start on $E_0/\mathbb{F}_{p^2}$ when $\mathbf{End}(E)$ is not known. Don't know how?

Generating secret kernels

Recall

- $P_A, Q_A \in E_0[2^{e_A}]$ and $P_B, Q_B \in E_0[3^{e_B}]$ with *full order* Weil pairings
- Alice's secret is $\langle [m_A]P_A + [n_A]Q_A \rangle$, Bob's is $\langle [m_B]P_B + [n_B]Q_B \rangle$

We take

- $m_A = m_B = 1$, $n_A \in [0, 2^{\ell})$ and $n_B \in [0, 2^{\ell'})$
- $Q_A = [3^{e_B}](z_1, -)$ and $P_A = [3^{e_B}](z_2 + i, -)$
- $Q_B = [2^{e_A}](z_3, -)$ and $P_B = [2^{e_A}](z_4 + i, -)$

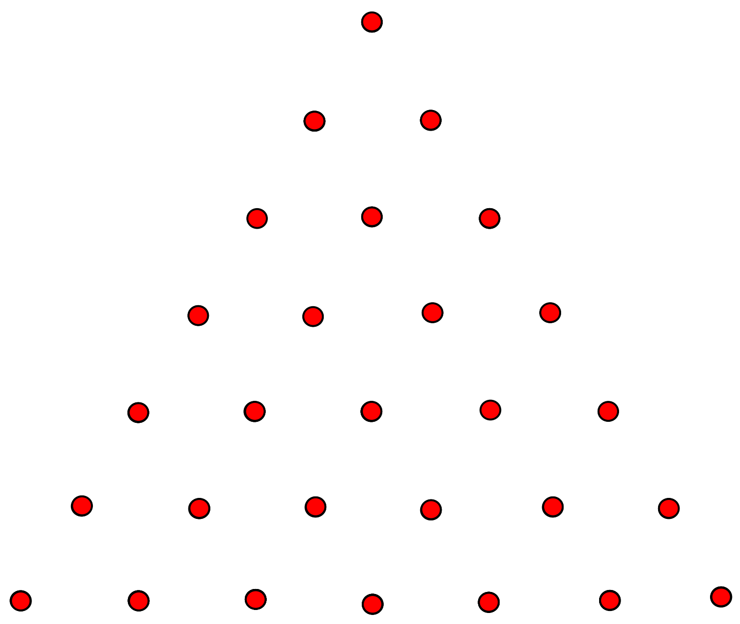
$z_i \in \mathbb{N}$ smallest such that points span torsions

$(z_1, -)$, $(z_3, -)$: $\mathbb{F}_p$ $\quad$ $(z_2 + i, -)$, $(z_4 + i, -)$: $\mathbb{F}_{p^2}$

Consequences

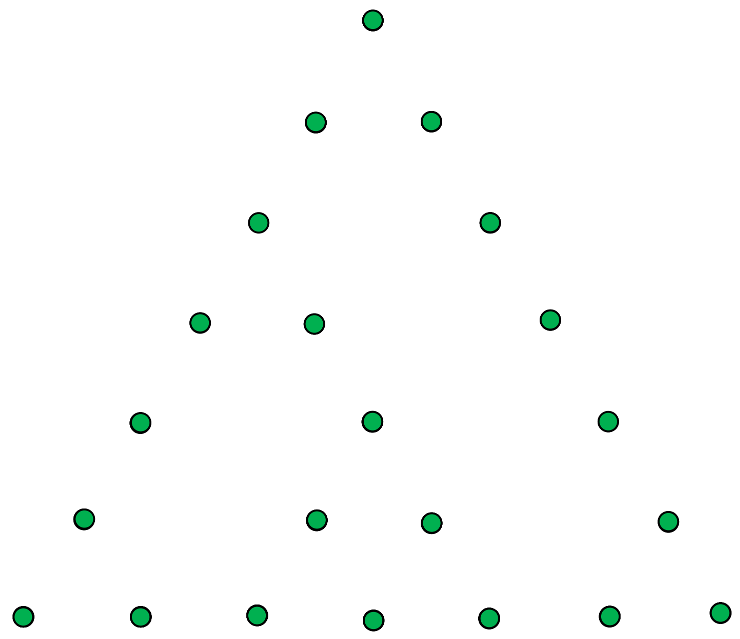
- Simple, uniform "3 point ladder" for computing $P + [n]Q$ [see FLOR'17] 👍
- $R = P + [n]Q$ can never be such that $[2^z]R = (0,0)$, so one 4-isogeny function 👍
- Don't reach all possible subgroups. Problem? 🤷‍♂️

The main loop



Simple, but slow

e.g. $\mathbf{28441} \times [3] + \mathbf{239} \times \phi_3(x)$



Optimal strategy [DJP'11] is harder, but much faster

e.g. $\mathbf{811} \times [3] + \mathbf{1124} \times \phi_3(x)$

Spec/code gives concrete algorithm for deriving, checking and executing the optimal strategy

The problem with reusing static keys

- Galbraith-Petit-Shani-Ti: $\boldsymbol{P}, \boldsymbol{Q}$ both order $\mathbf{2}^{e_A}$, and Alice's static secret $\boldsymbol{\alpha} \in \mathbb{Z}$

$$\langle P + [\alpha]Q \rangle = \langle P + [\alpha](Q + [2^{e_A - 1}]P) \rangle \qquad \text{iff } \alpha \text{ is even}$$

- Send Alice $\tilde{P} = P$ and $\tilde{Q} = (Q + [2^{e_A-1}]P)$, if DH works fine, then $\boldsymbol{\alpha}$ is even, else odd

- **Even case** ($\alpha = 2\hat{\alpha}$):

$$\langle P + [2\hat{\alpha}]Q \rangle = \langle P + [2\hat{\alpha}]\left(Q + [2^{e_A-2}]P\right) \rangle \qquad \text{iff } \hat{\alpha} \text{ is even}$$

so send $\tilde{P} = P$and $\tilde{Q} = (Q + [2^{e_A-2}]P)$

- **Odd case** ($\alpha = 2\hat{\alpha} + 1$):

$$\langle P + [2\hat{\alpha} + 1]Q \rangle = \langle P - [2^{e_A-2}]Q + [2\hat{\alpha} + 1]\left(Q + [2^{e_A-2}]Q\right) \rangle \qquad \text{iff } \hat{\alpha} \text{ is even}$$

so send $\tilde{P} = [1 - 2^{e_A-2}]P$ and $\tilde{Q} = [1 + 2^{e_A-2}]Q$

- … continuing yields $\boldsymbol{\alpha}$ in $\mathbf{log_2}\boldsymbol{\alpha}$ adaptive interactions!!!

No known *Weil* to detect foul play, provided $\tilde{P}, \tilde{Q}$ are scaled correctly!

Passively secure encryption (IND-CPA PKE), à la ElGamal

Alice

Bob

$PK_A = [\ \phi_A(E_0), \phi_A(P_B), \phi_A(Q_B)\ ]$

$PK_B = [\ \phi_B(E_0), \phi_B(P_A), \phi_B(Q_A)\ ]$

$j = j(E_{BA}) = j\left(\phi_B\big(\phi_A(E_0)\big)\right)$

$[PK_B\ , H_1(j) \oplus m]$

$j = j(E_{AB}) = j\left(\phi_A\big(\phi_B(E_0)\big)\right)$

Actively secure key encapsulation (IND-CCA KEM)

Alice

$$PK_A = [\, \phi_A(E_0), \phi_A(P_B), \phi_A(Q_B) \,]$$

Bob

$$PK_B = [\, \phi_B(E_0), \phi_B(P_A), \phi_B(Q_A) \,]$$

$$j = j(E_{BA}) = j\left(\phi_B(\phi_A(E_0))\right)$$

$$[PK_B \,, H_1(j) \oplus m]$$

$$j = j(E_{AB}) = j\left(\phi_A(\phi_B(E_0))\right)$$

Actively secure key encapsulation (IND-CCA KEM)

Alice

$$PK_A = [\ \phi_A(E_0), \phi_A(P_B), \phi_A(Q_B)\ ]$$

$$s \in_R \{0,1\}^\ell$$

Bob

$$m \in_R \{0,1\}^\ell$$

$$r = H_2(PK_A, m)$$

$$PK_B(r) = [\ \phi_B(E_0), \phi_B(P_A), \phi_B(Q_A)\ ]$$

$$j = j(E_{BA}) = j\left(\phi_B\left(\phi_A(E_0)\right)\right)$$

Bob → Alice: $[PK_B(r)\ , H_1(j) \oplus m]$

$$j = j(E_{AB}) = j\left(\phi_A\left(\phi_B(E_0)\right)\right)$$

Actively secure key encapsulation (IND-CCA KEM)

Alice

$$PK_A = [\ \phi_A(E_0), \phi_A(P_B), \phi_A(Q_B)\ ]$$

$$s \in_R \{0,1\}^{\ell}$$

$$c = [PK_B(r)\ , H_1(j) \oplus m]$$

$$j = j(E_{AB}) = j\left(\phi_A\left(\phi_B(E_0)\right)\right)$$

Bob

$$m \in_R \{0,1\}^{\ell}$$

$$r = H_2(PK_A, m)$$

$$PK_B(r) = [\ \phi_B(E_0), \phi_B(P_A), \phi_B(Q_A)\ ]$$

$$j = j(E_{BA}) = j\left(\phi_B\left(\phi_A(E_0)\right)\right)$$

$$K = H_3(c, m)$$

Actively secure key encapsulation (IND-CCA KEM)

Alice

$$PK_A = [\,\phi_A(E_0), \phi_A(P_B), \phi_A(Q_B)\,]$$

$$s \in_R \{0,1\}^\ell$$

$$c = [PK_B(r)\,, H_1(j) \oplus m]$$

$$j = j(E_{AB}) = j\left(\phi_A\left(\phi_B(E_0)\right)\right)$$

$$m' = c[2] \oplus H_1(j)$$

Bob

$$m \in_R \{0,1\}^\ell$$

$$r = H_2(PK_A, m)$$

$$PK_B(r) = [\,\phi_B(E_0), \phi_B(P_A), \phi_B(Q_A)\,]$$

$$j = j(E_{BA}) = j\left(\phi_B\left(\phi_A(E_0)\right)\right)$$

$$K = H_3(c, m)$$

Actively secure key encapsulation (IND-CCA KEM)

Alice

$$PK_A = [\ \phi_A(E_0), \phi_A(P_B), \phi_A(Q_B)\ ]$$

$$s \in_R \{0,1\}^\ell$$

Bob

$$m \in_R \{0,1\}^\ell$$

$$r = H_2(PK_A, m)$$

$$PK_B(r) = [\ \phi_B(E_0), \phi_B(P_A), \phi_B(Q_A)\ ]$$

$$j = j(E_{BA}) = j\left(\phi_B\left(\phi_A(E_0)\right)\right)$$

$$K = H_3(c, m)$$

$$c = [PK_B(r)\ , H_1(j) \oplus m]$$

$$j = j(E_{AB}) = j\left(\phi_A\left(\phi_B(E_0)\right)\right)$$

$$m' = c[2] \oplus H_1(j)$$

$$r' = H_2(PK_A, m')$$

Actively secure key encapsulation (IND-CCA KEM)

Alice

$$PK_A = [\ \phi_A(E_0), \phi_A(P_B), \phi_A(Q_B)\ ]$$

$$s \in_R \{0,1\}^{\ell}$$

Bob

$$m \in_R \{0,1\}^{\ell}$$

$$r = H_2(PK_A, m)$$

$$PK_B(r) = [\ \phi_B(E_0), \phi_B(P_A), \phi_B(Q_A)\ ]$$

$$j = j(E_{BA}) = j\left(\phi_B\left(\phi_A(E_0)\right)\right)$$

$$K = H_3(c, m)$$

Bob → Alice:

$$c = [PK_B(r)\ , H_1(j) \oplus m]$$

Alice:

$$j = j(E_{AB}) = j\left(\phi_A\left(\phi_B(E_0)\right)\right)$$

$$m' = c[2] \oplus H_1(j)$$

$$r' = H_2(PK_A, m')$$

if $PK_B(r') = c[1]$ then $K = H_3(c, m')$ else $K = H_3(c, s)$

Actively secure key encapsulation (IND-CCA KEM)

Alice

$$PK_A = [\ \phi_A(E_0), \phi_A(P_B), \phi_A(Q_B)\ ]$$

$$s \in_R \{0,1\}^{\ell}$$

Bob

$$m \in_R \{0,1\}^{\ell}$$

$$r = H_2(PK_A, m)$$

$$PK_B(r) = [\ \phi_B(E_0), \phi_B(P_A), \phi_B(Q_A)\ ]$$

$$j = j(E_{BA}) = j\left(\phi_B(\phi_A(E_0))\right)$$

$$K = H_3(c, m)$$

$$c = [PK_B(r)\ , H_1(j) \oplus m]$$

$$j = j(E_{AB}) = j\left(\phi_A(\phi_B(E_0))\right)$$

$$m' = c[2] \oplus H_1(j)$$

$$r' = H_2(PK_A, m')$$

if $PK_B(r') = c[1]$ then $K = H_3(c, m')$ else $K = H_3(c, s)$

$$H_1(j) = \text{cSHAKE256}(j, k, \text{""}, 2)$$

$$H_2(PK_A, m) = \text{cSHAKE256}(m || PK_A, e_2, \text{""}, 0)$$

$$H_3(c, m) = \text{cSHAKE256}(m || c, k, \text{""}, 1)$$

The curves and their security estimates

$$p = 2^{e_A} 3^{e_B} - 1$$

Name (SIKEp+ $\lceil \log_2 p \rceil$)	(e_A, e_B)	k	2^{k-1}	**min** $(\sqrt{2^{e_A}}, \sqrt{3^{e_3}})$	$\sqrt{2^k}$	**min** $(\sqrt[3]{2^{e_2}}, \sqrt[3]{3^{e_3}})$
SIKEp503	(250,159)	128	2^{127}	2^{125}	2^{64}	2^{83}
SIKEp761	(372,239)	192	2^{191}	2^{186}	2^{96}	2^{124}
SIKEp964	(486,301)	256	2^{255}	2^{238}	2^{128}	2^{159}

SIKE vs. IND-CCA lattice KEMs

Name	Primitive	Quantum sec (bits)	Encaps+ Decaps (ms)	Size of Encaps. (KB)
NTRU-KEM	NTRU	123	0.03	1.3
Kyber	M-LWE	161	0.07	1.2
FrodoKEM	LWE	103-150	1.2 – 2.3	9.5 – 15.4
SIKE	Supersingular Isogeny	84-125	10 – 30	0.4 – 0.6

Results obtained on 3.4GHz Intel Haswell (Kyber and NTRU-KEM) or Skylake (FrodoKEM and SIKE)

Easy ECDH hybrid

There are exponentially many a such that $E_a / \mathbb{F}_{p^2}: y^2 = x^3 + ax^2 + x$ is in the supersingular isogeny class. These are all unsuitable for ECDH.

There are also exponentially many A such that $E_a / \mathbb{F}_p: y^2 = x^3 + ax^2 + x$ is suitable for ECDH. E.g., smallest $a \in \mathbb{F}_p$ such that E_a is twist-secure.

Public keys only 1.17x larger, slowdown less than this, but….

e.g., on smallest curve we replace 128-bit classical security (SSDDH) with 256-bit classical security (ECDLP)

Questions?