Classical cryptanalysis of supersingular isogenies

Work in progress with...

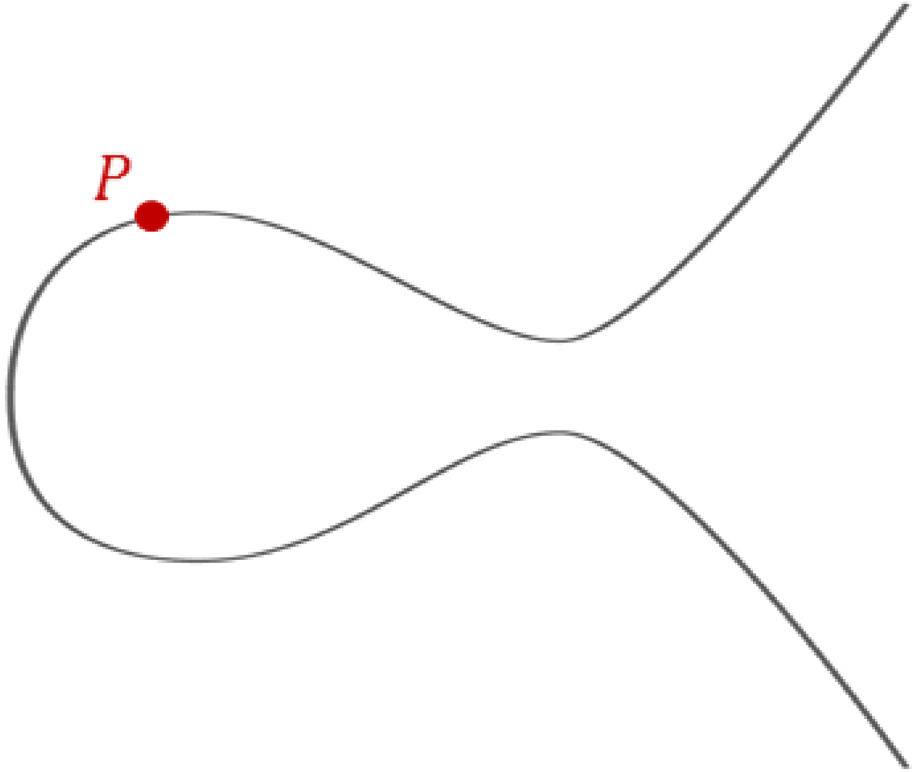
Craig Costello[1], Patrick Longa[1], Michael Naehrig[1], Joost Renes[2], Fernando Virdia[3]

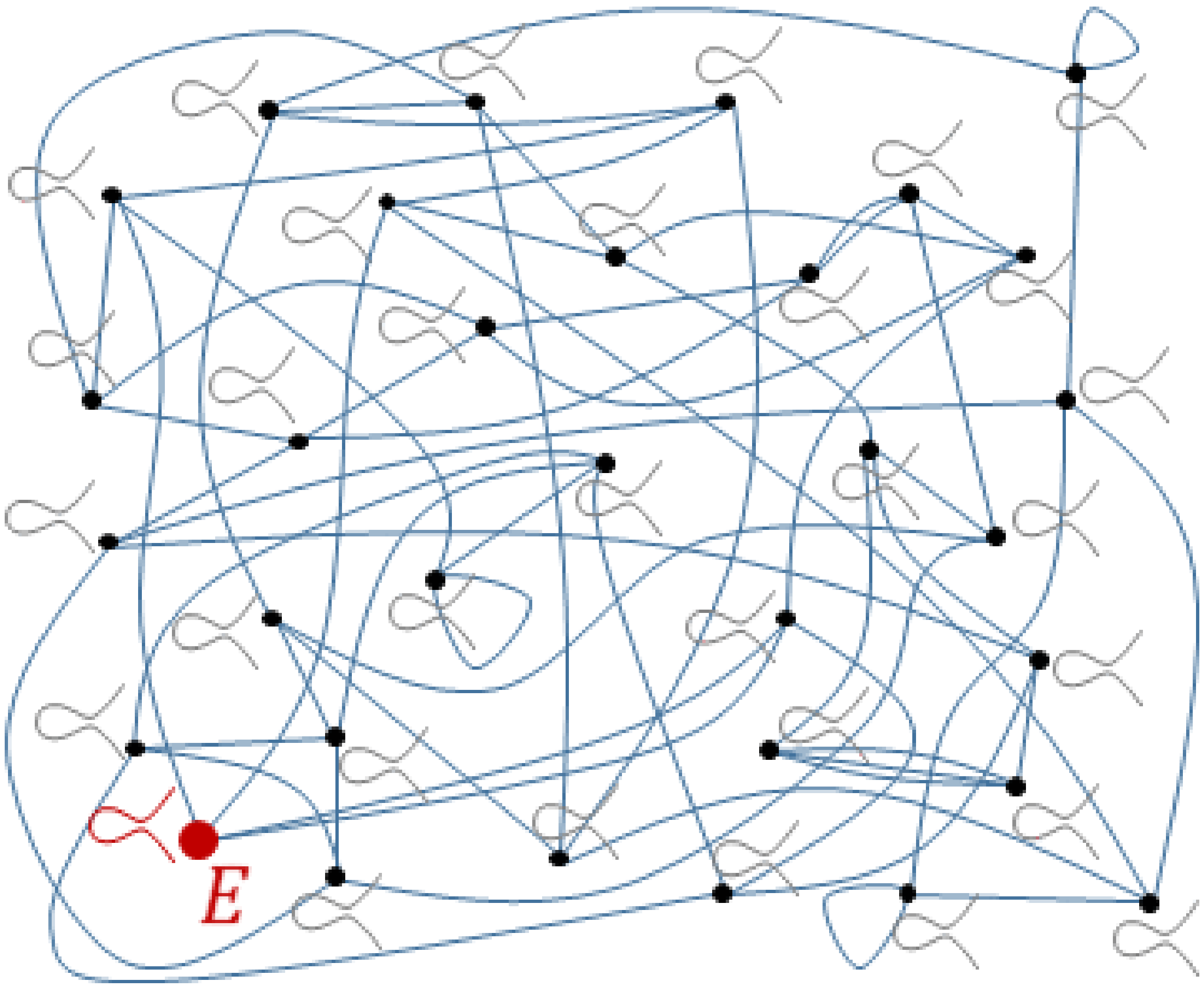
ASEC 2018
Adelaide, Australia
December 10

1 Microsoft® Research
2 Radboud University
3 Royal Holloway University of London

Embedded…?

ECC vs. post-quantum ECC





W. Castryck (GIF): https://www.esat.kuleuven.be/cosic/?p=7404

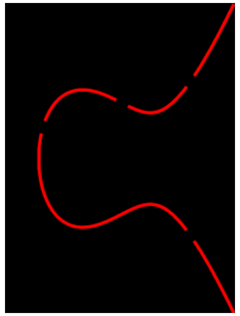
Diffie-Hellman instantiations

$\mathbb{Z}_q$

$g^a \bmod q$

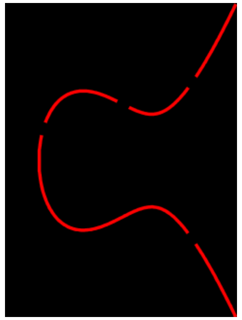
$g^b \bmod q$

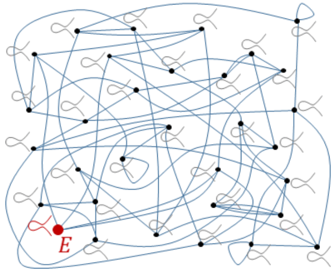
$\mathbb{Z}_q$



$[a]P$

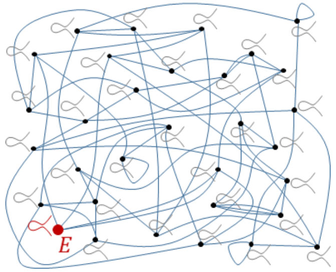
$[b]P$





$\phi_A(E)$

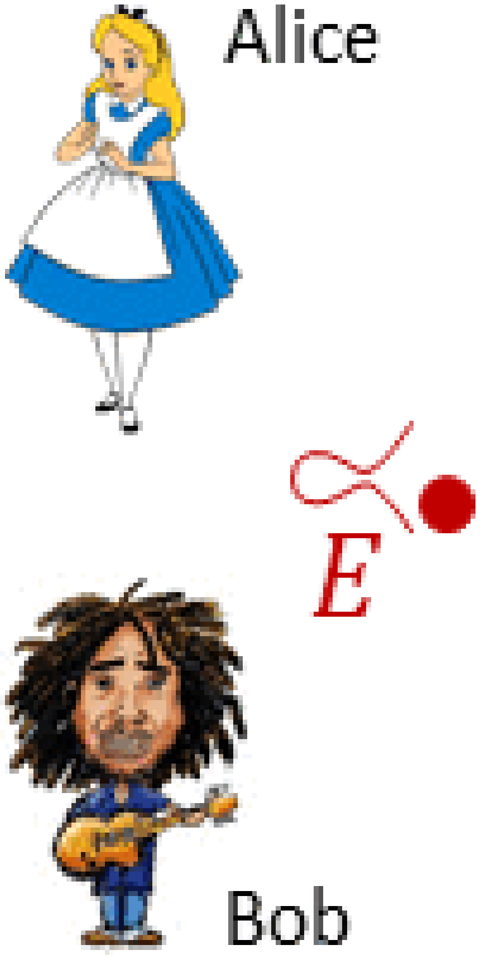
$\phi_B(E)$



Diffie-Hellman instantiations

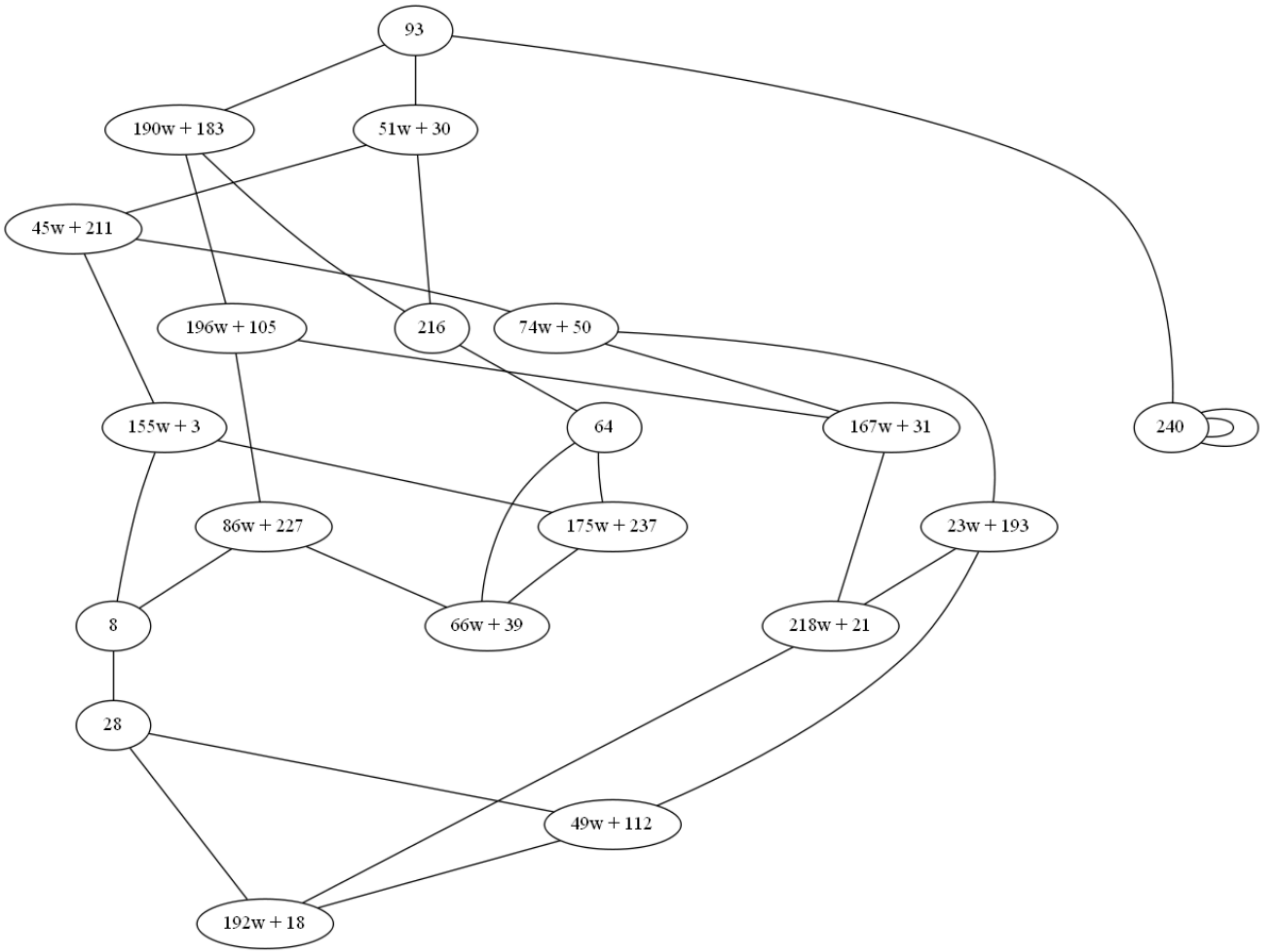
	DH	ECDH	SIDH
Elements	integers g modulo prime	points P in curve group	curves E in isogeny class
Secrets	exponents x	scalars k	isogenies ϕ
computations	$g, x \mapsto g^x$	$k, P \mapsto [k]P$	$\phi, E \mapsto \phi(E)$
hard problem	given g, g^x find x	given $P, [k]P$ find k	given $E, \phi(E)$ find ϕ

Alice does **2**-isogenies, Bob does **3**-isogenies

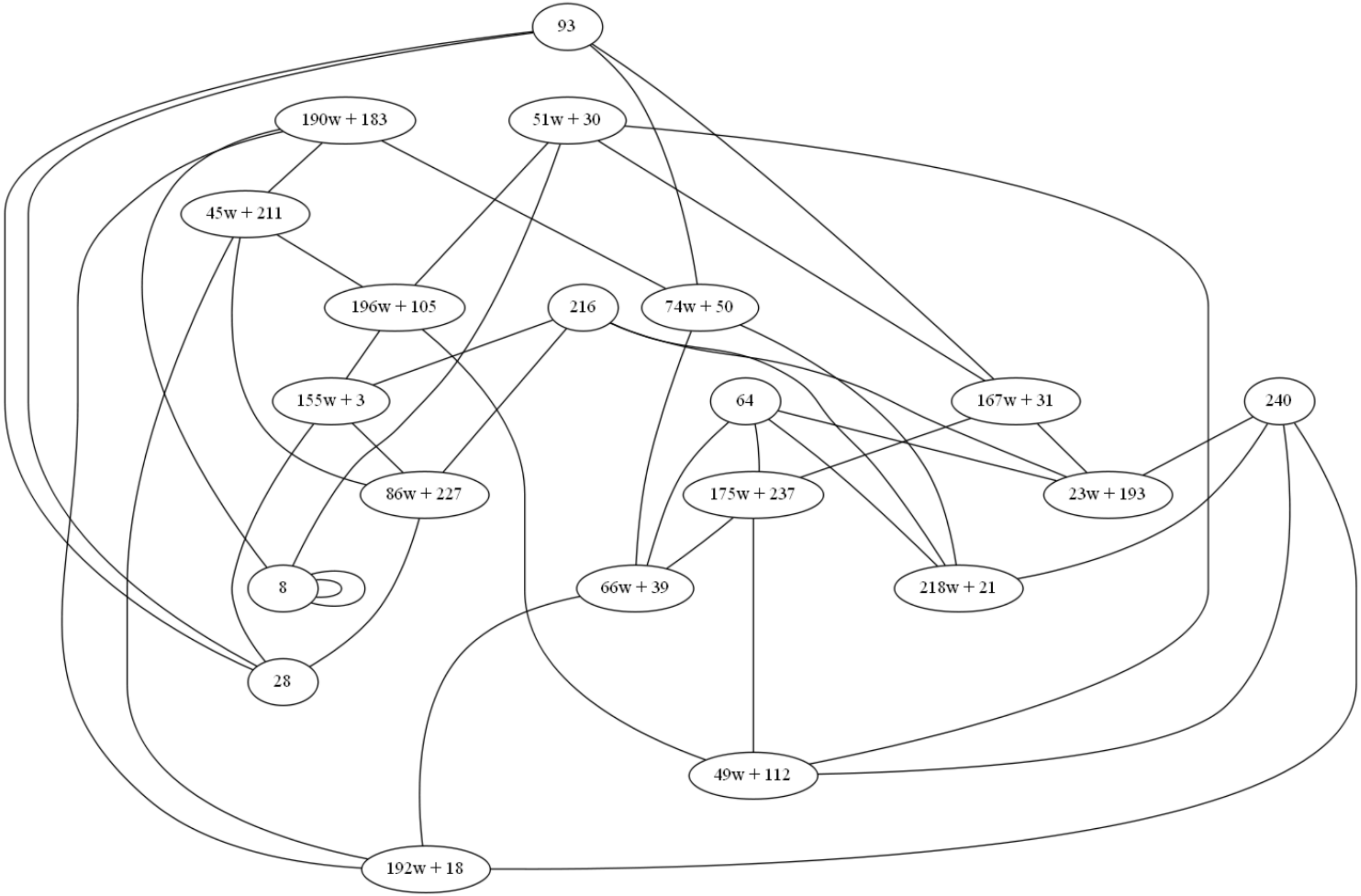


W. Castryck (GIF): "Elliptic curves are dead: long live elliptic curves" https://www.esat.kuleuven.be/cosic/?p=7404

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



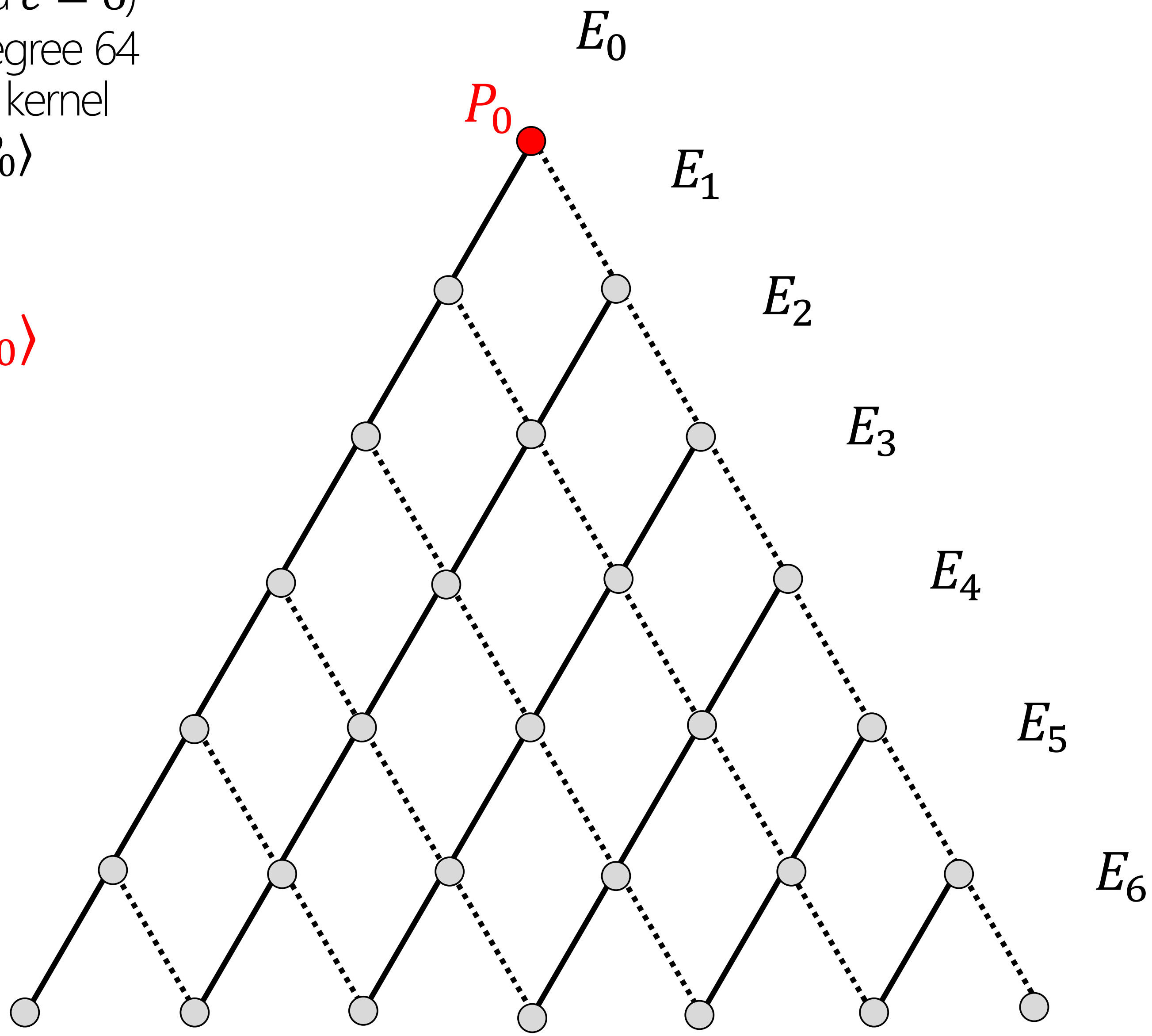
Supersingular isogeny graph for $\ell = 3$: $X(S_{241^2}, 3)$



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

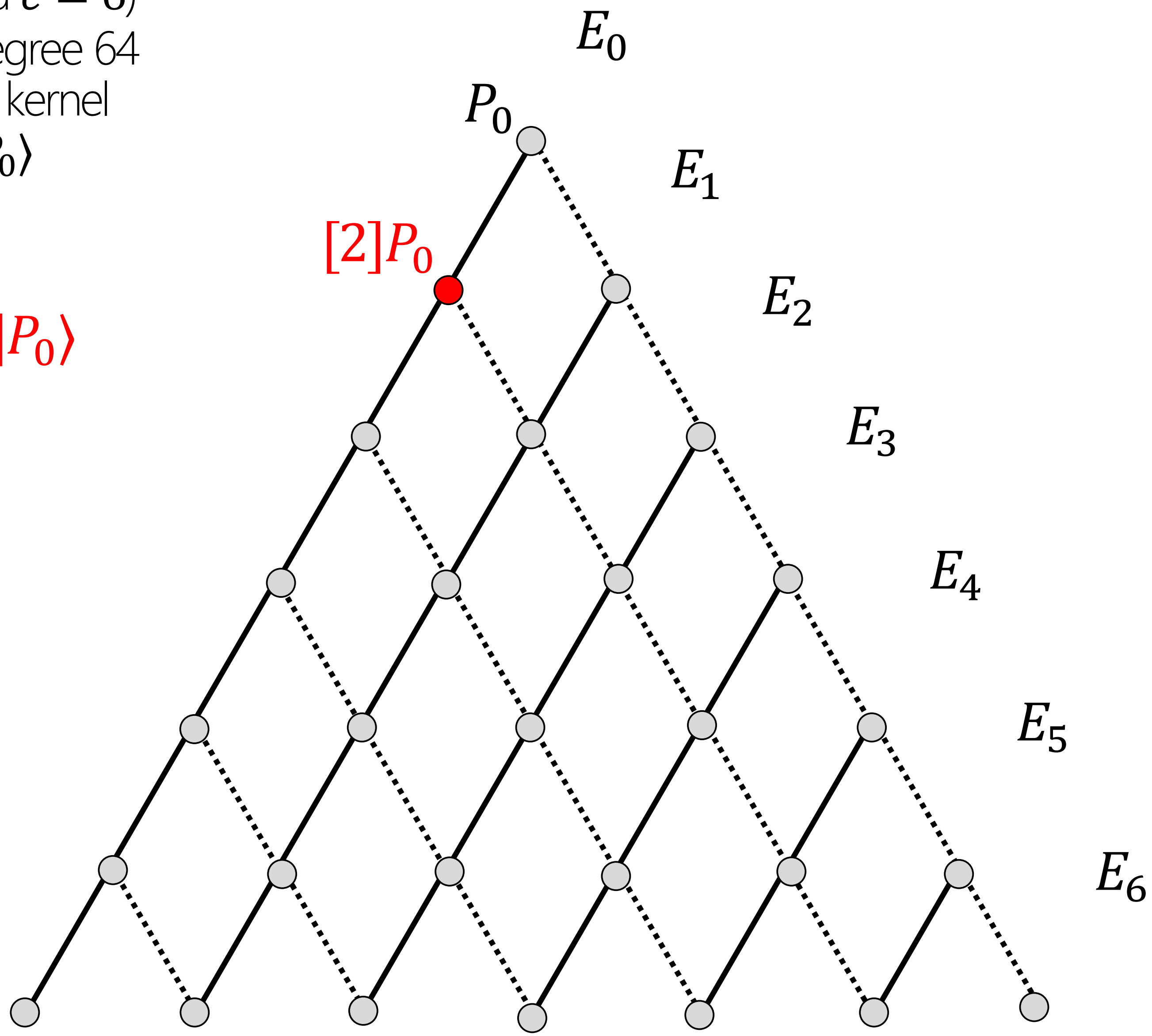
$E_6 = E_0 / \langle P_0 \rangle$



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

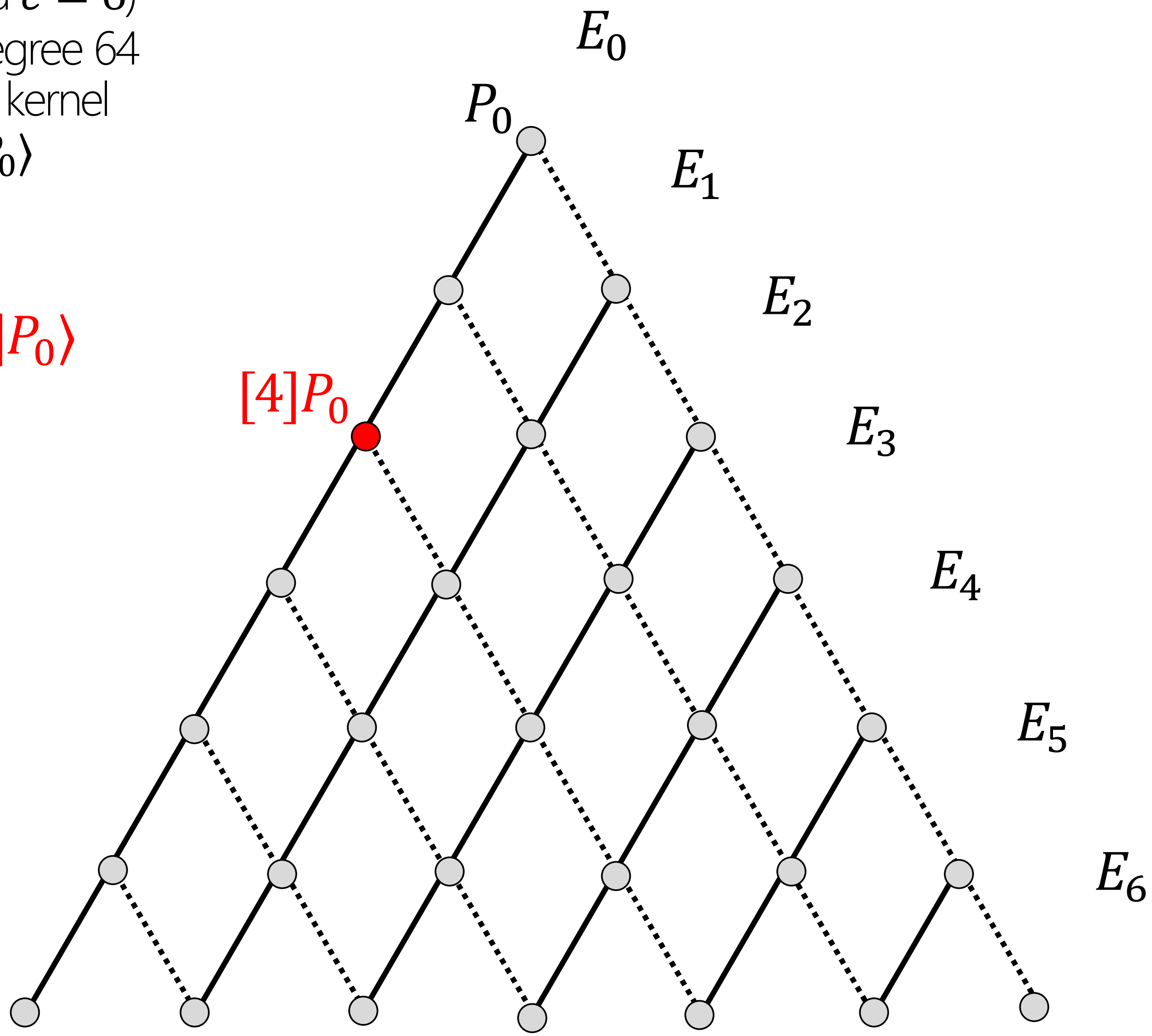
$E_5 = E_0/\langle [2]P_0 \rangle$



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

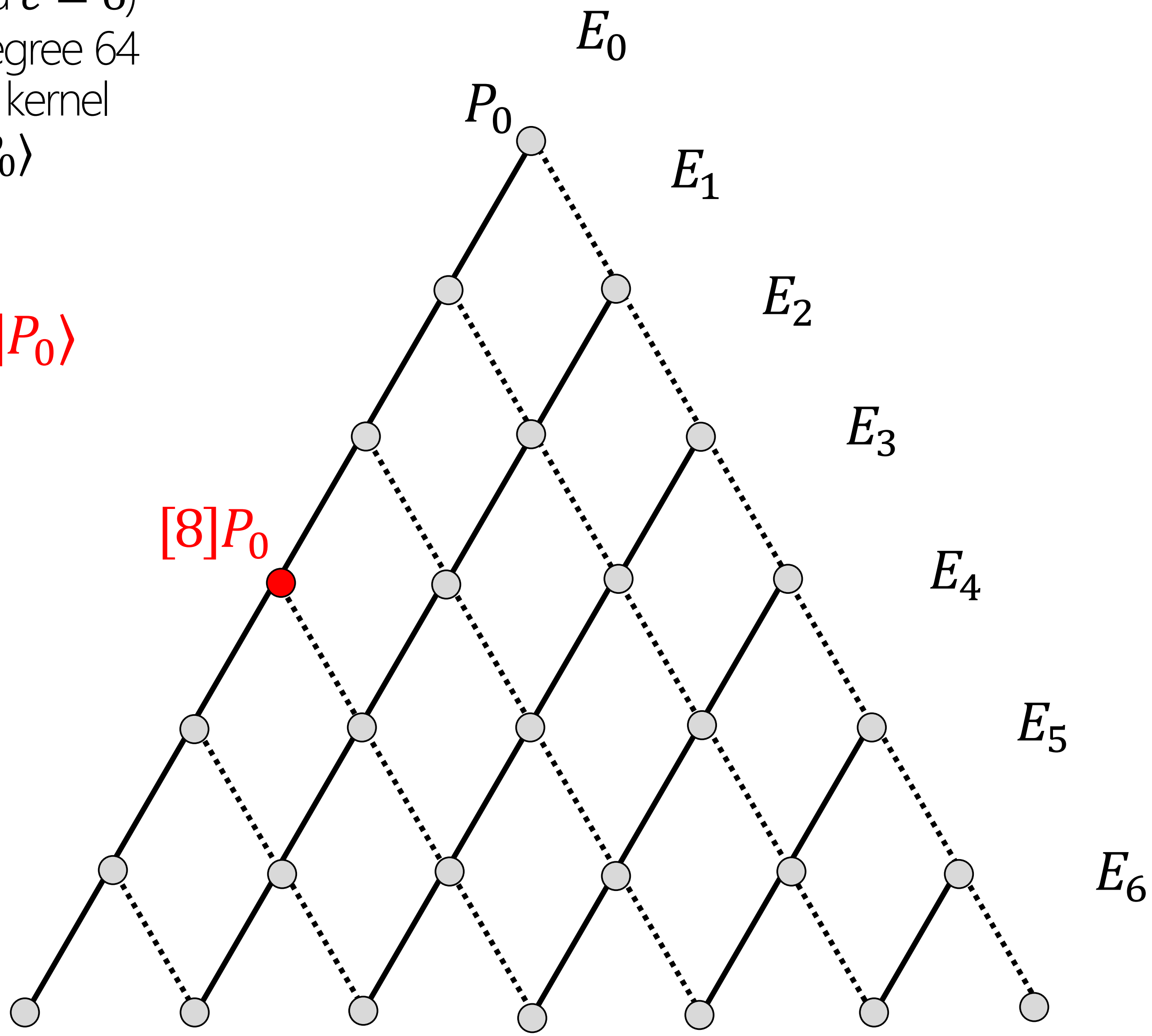
$E_4 = E_0 / \langle [4]P_0 \rangle$



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

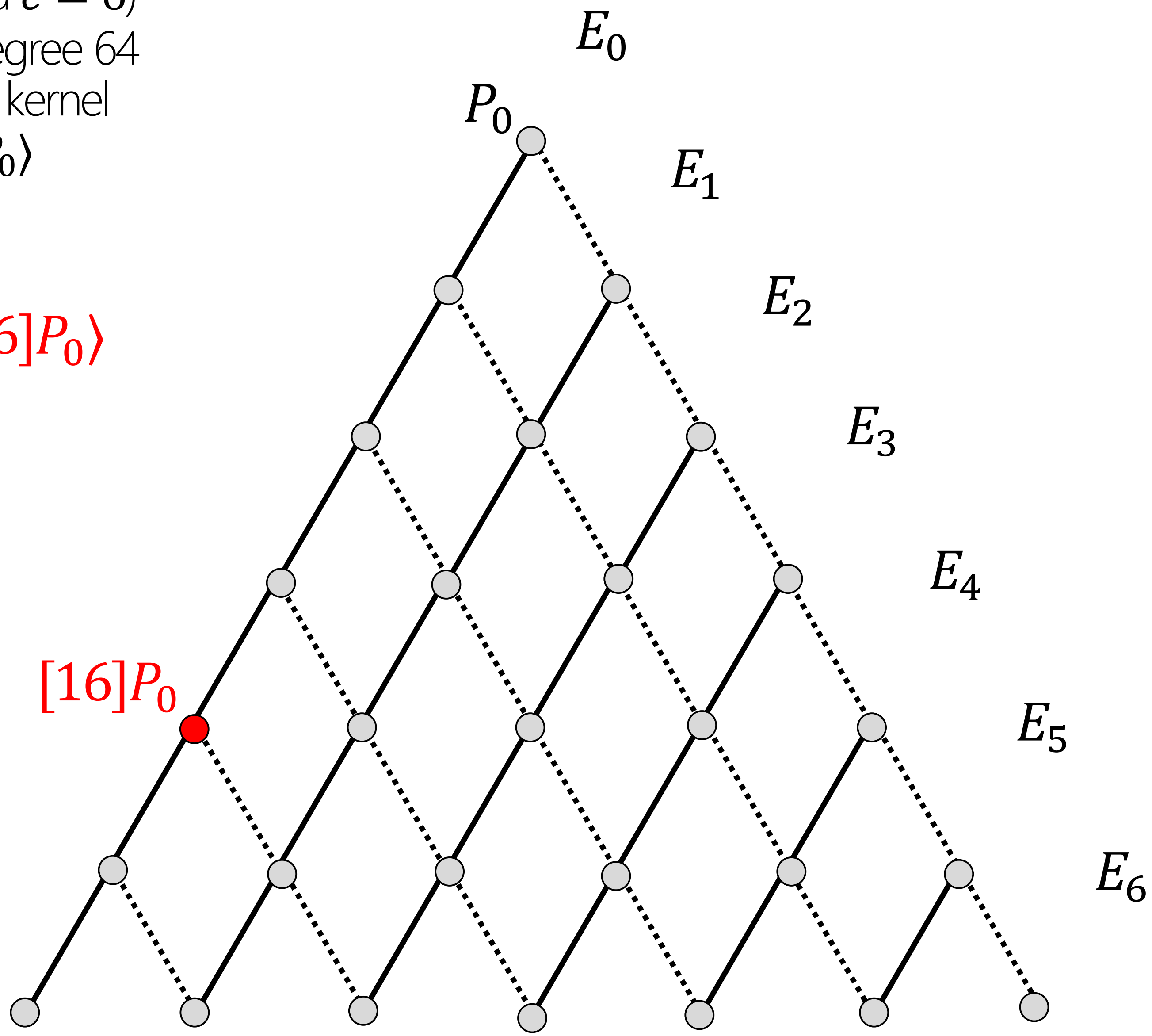
$E_3 = E_0/\langle [8]P_0 \rangle$



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

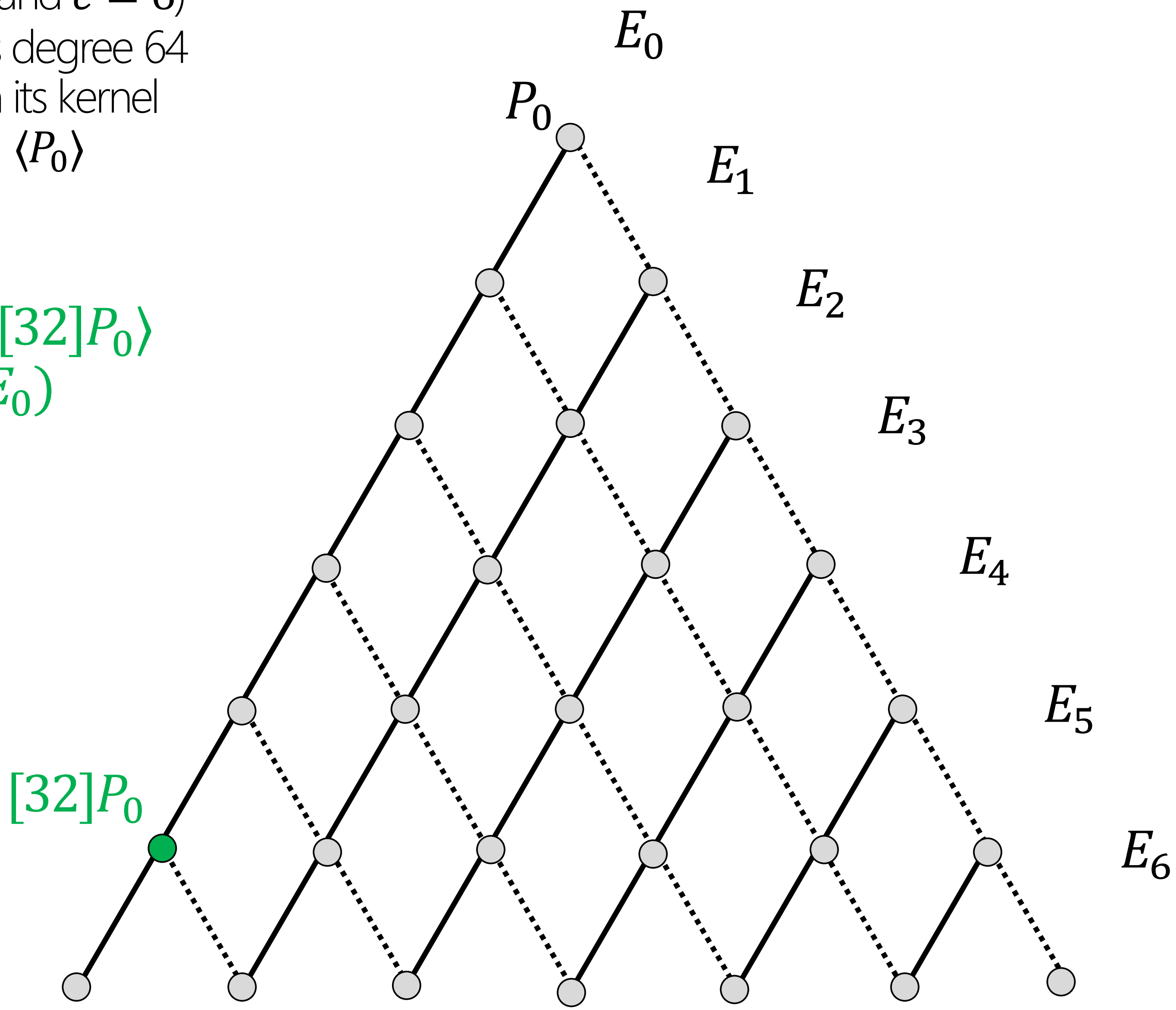
$E_2 = E_0 / \langle [16] P_0 \rangle$



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

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

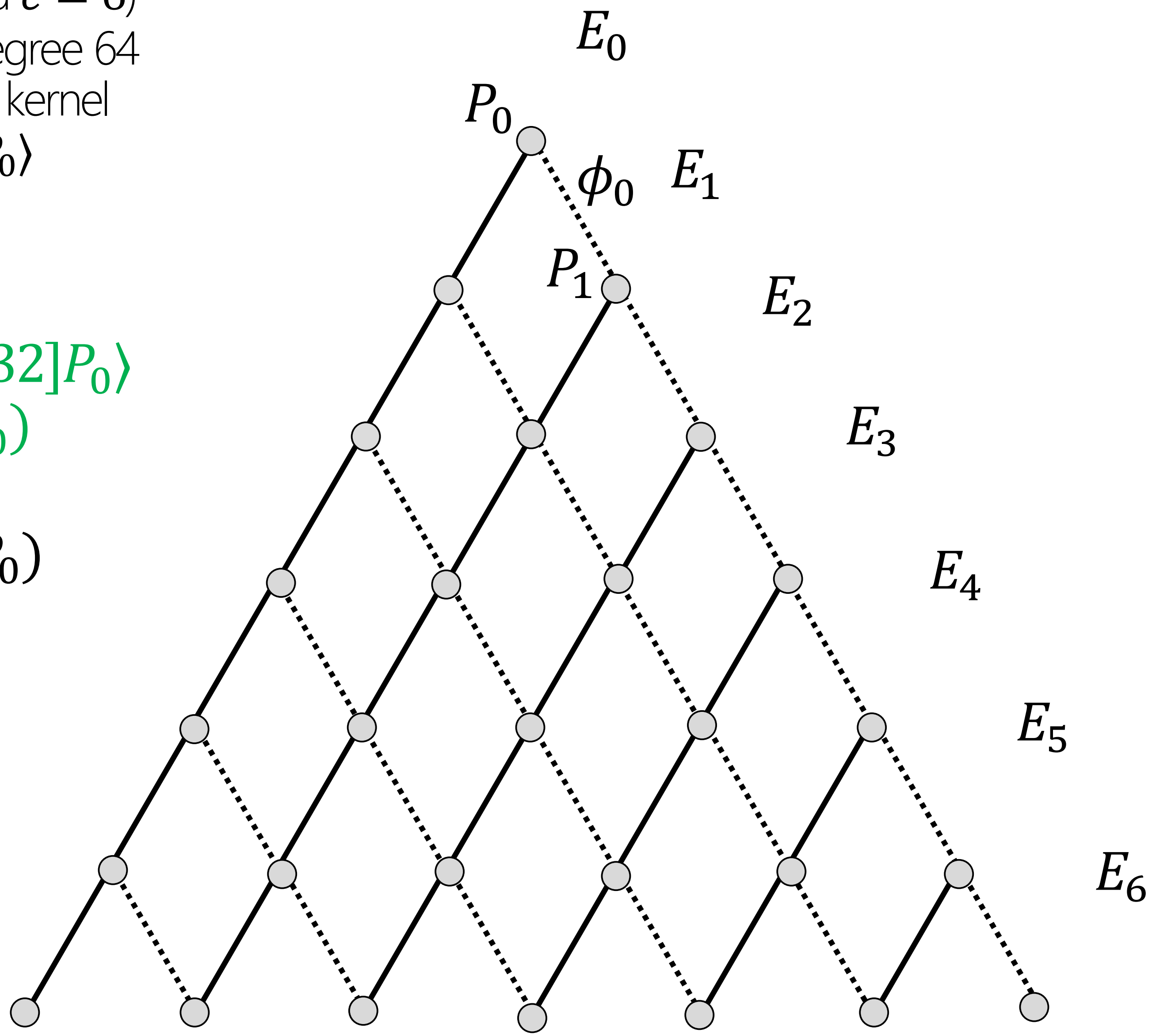


Computing ℓ^e degree isogenies

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

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

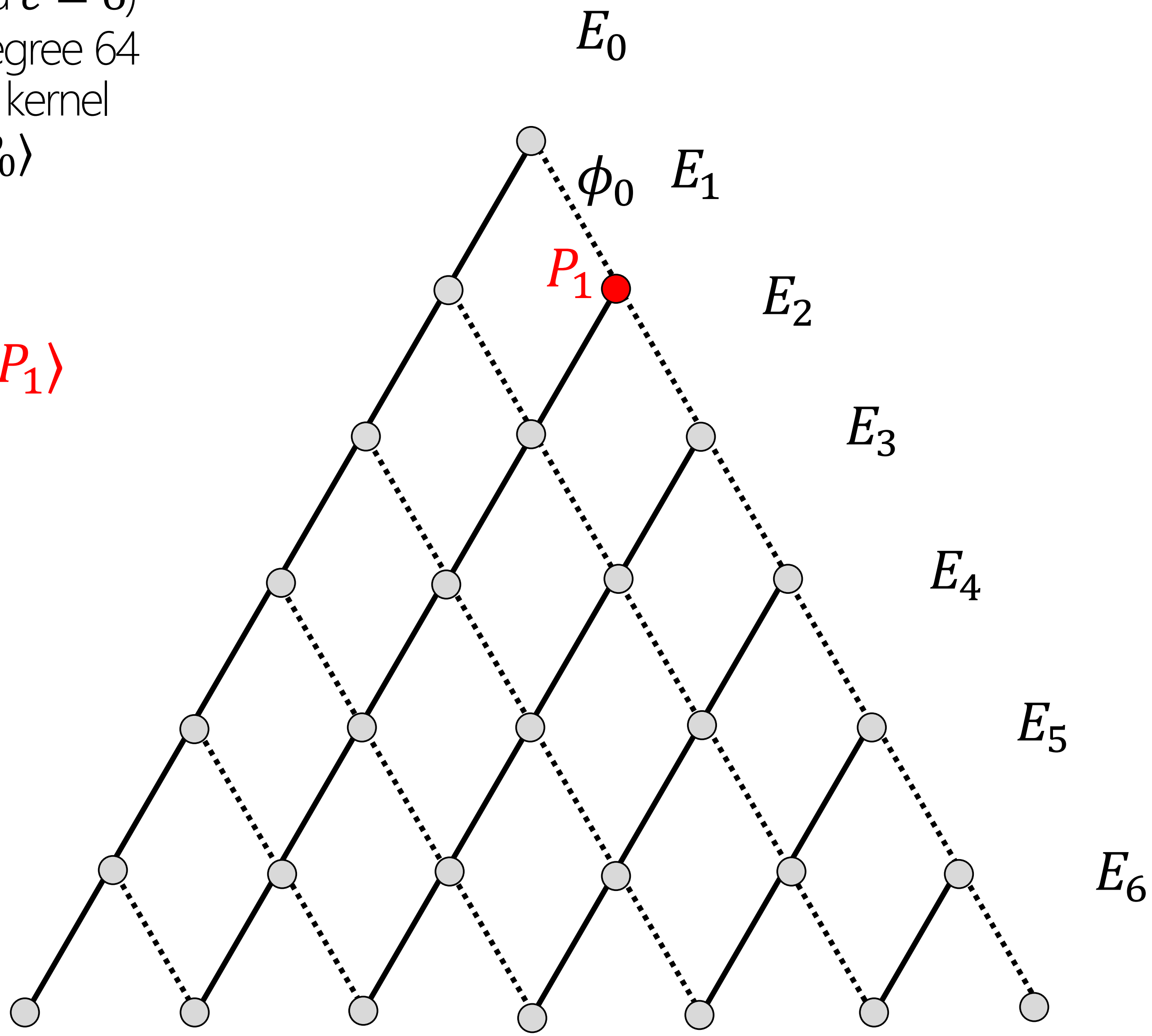
$$P_1 = \phi_0(P_0)$$



Computing ℓ^e degree isogenies

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

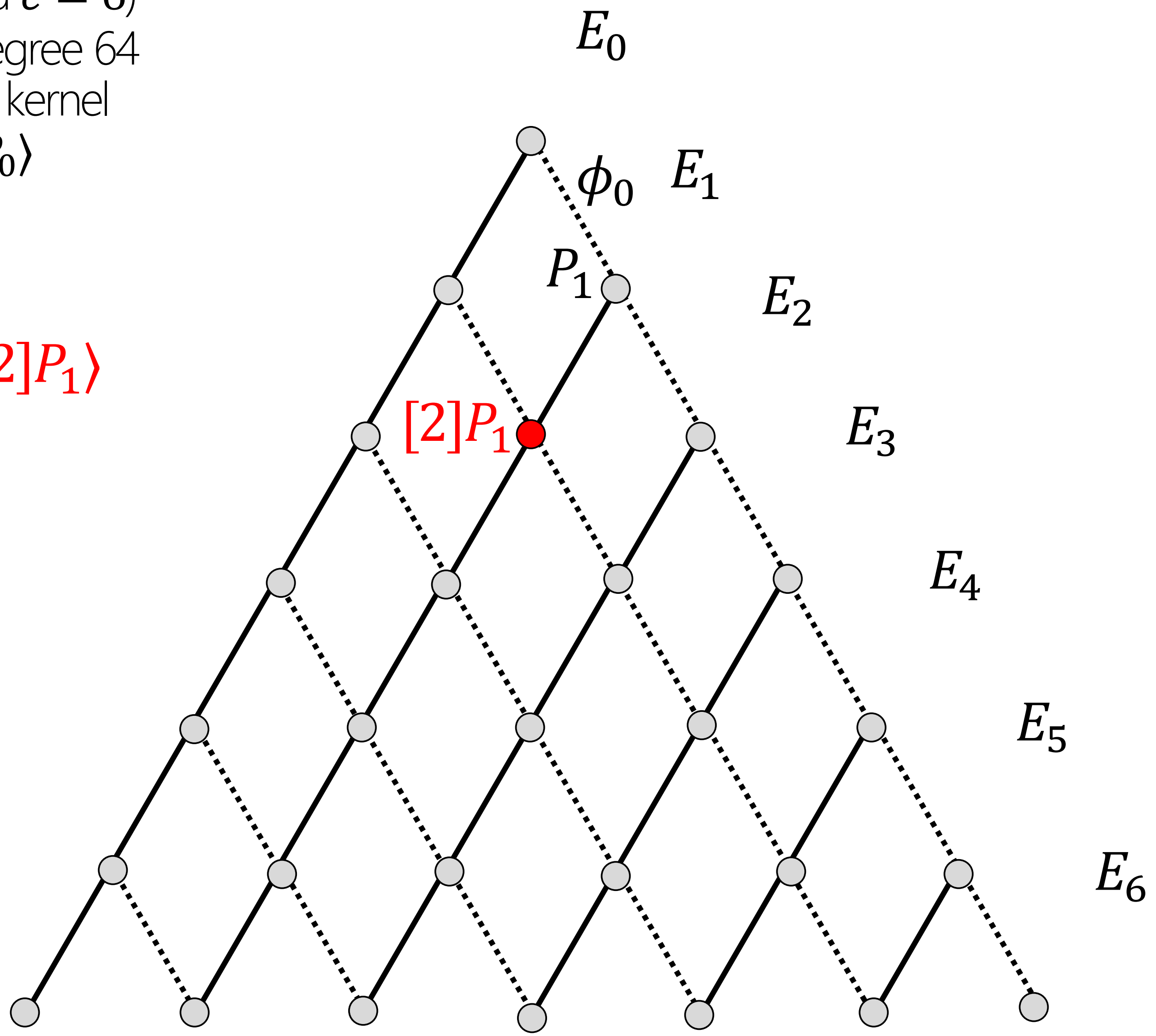
$E_6 = E_1/\langle P_1 \rangle$



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

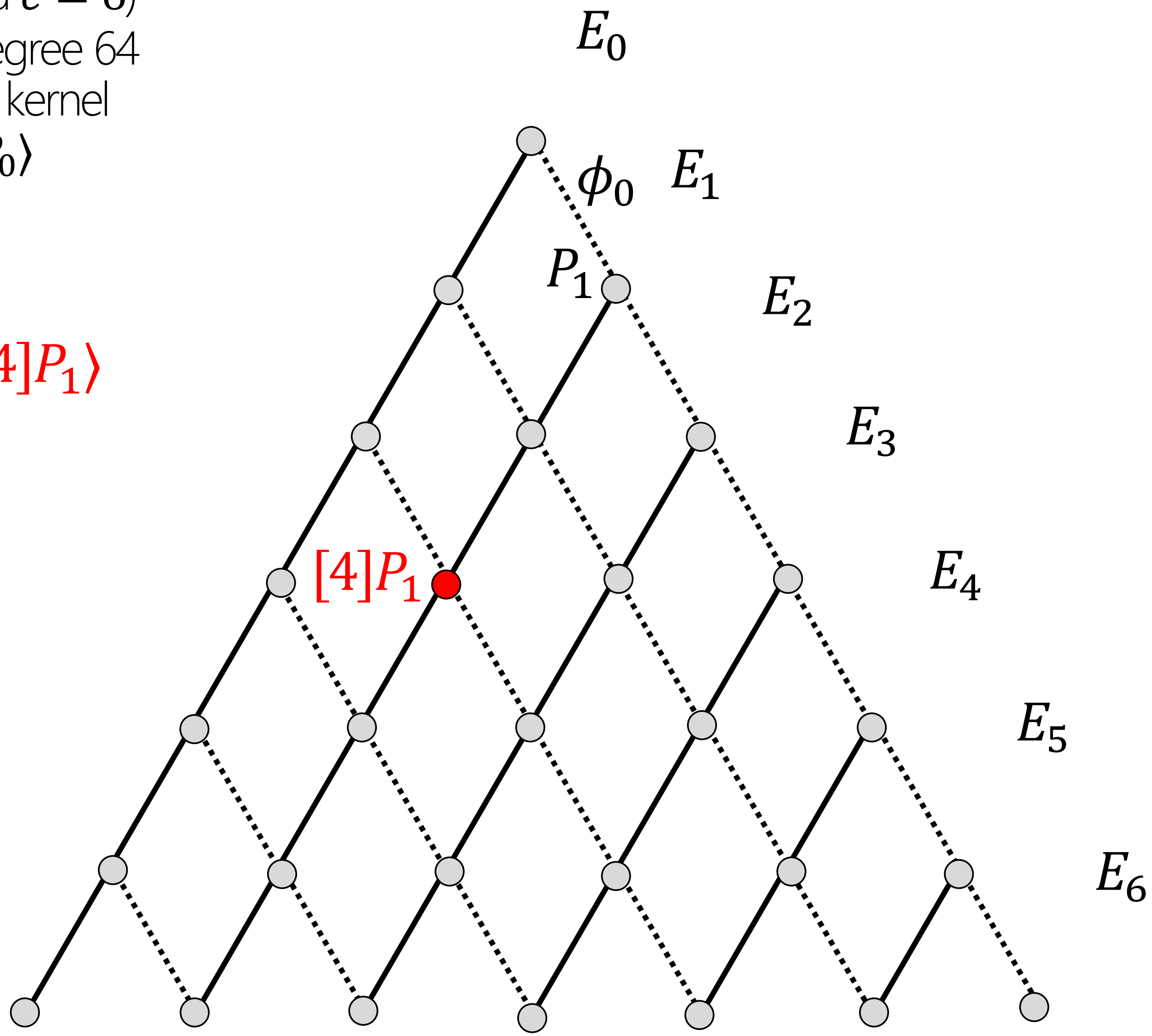
$E_5 = E_1/\langle [2]P_1 \rangle$



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

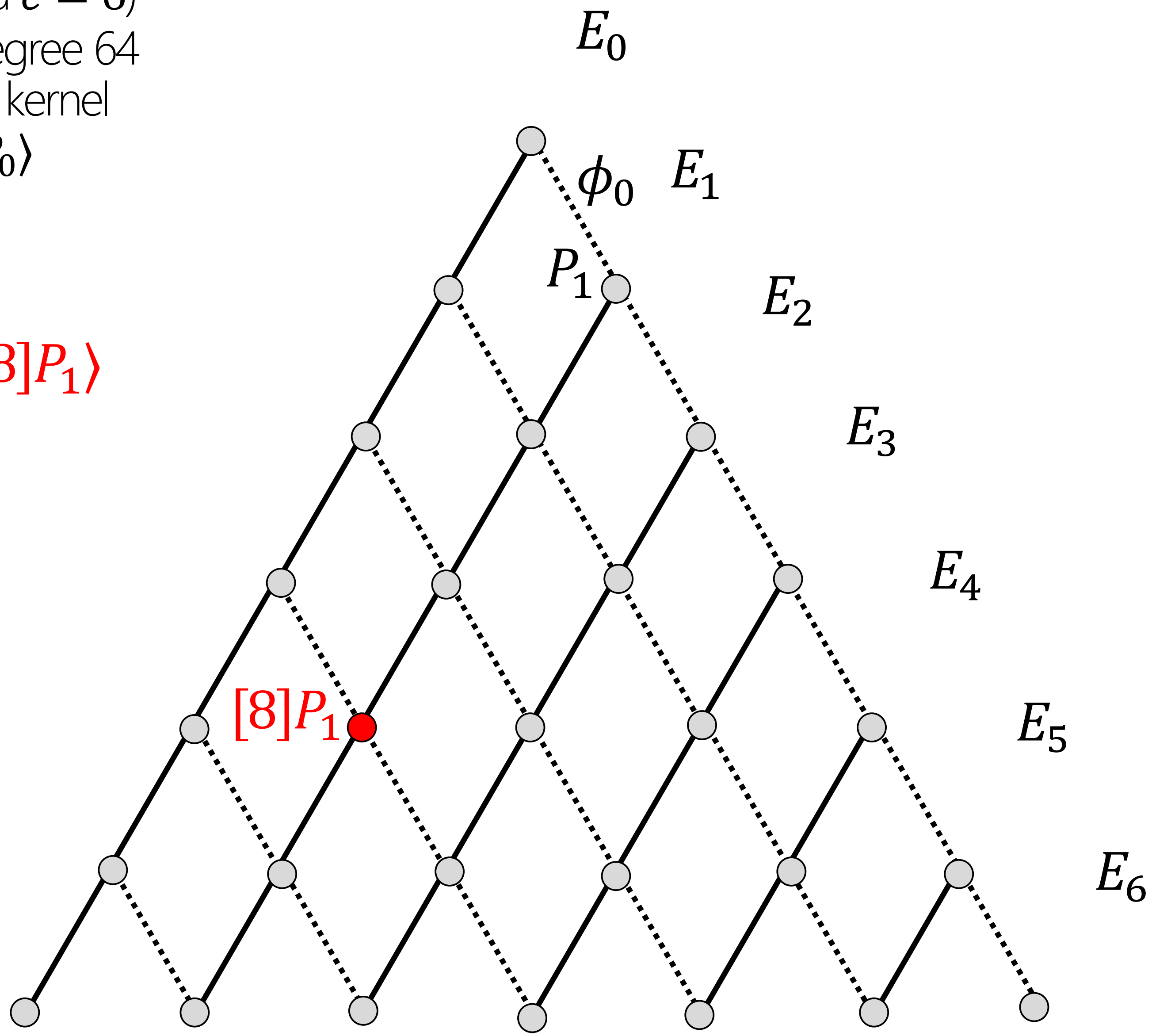
$E_4 = E_1 / \langle [4]P_1 \rangle$



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

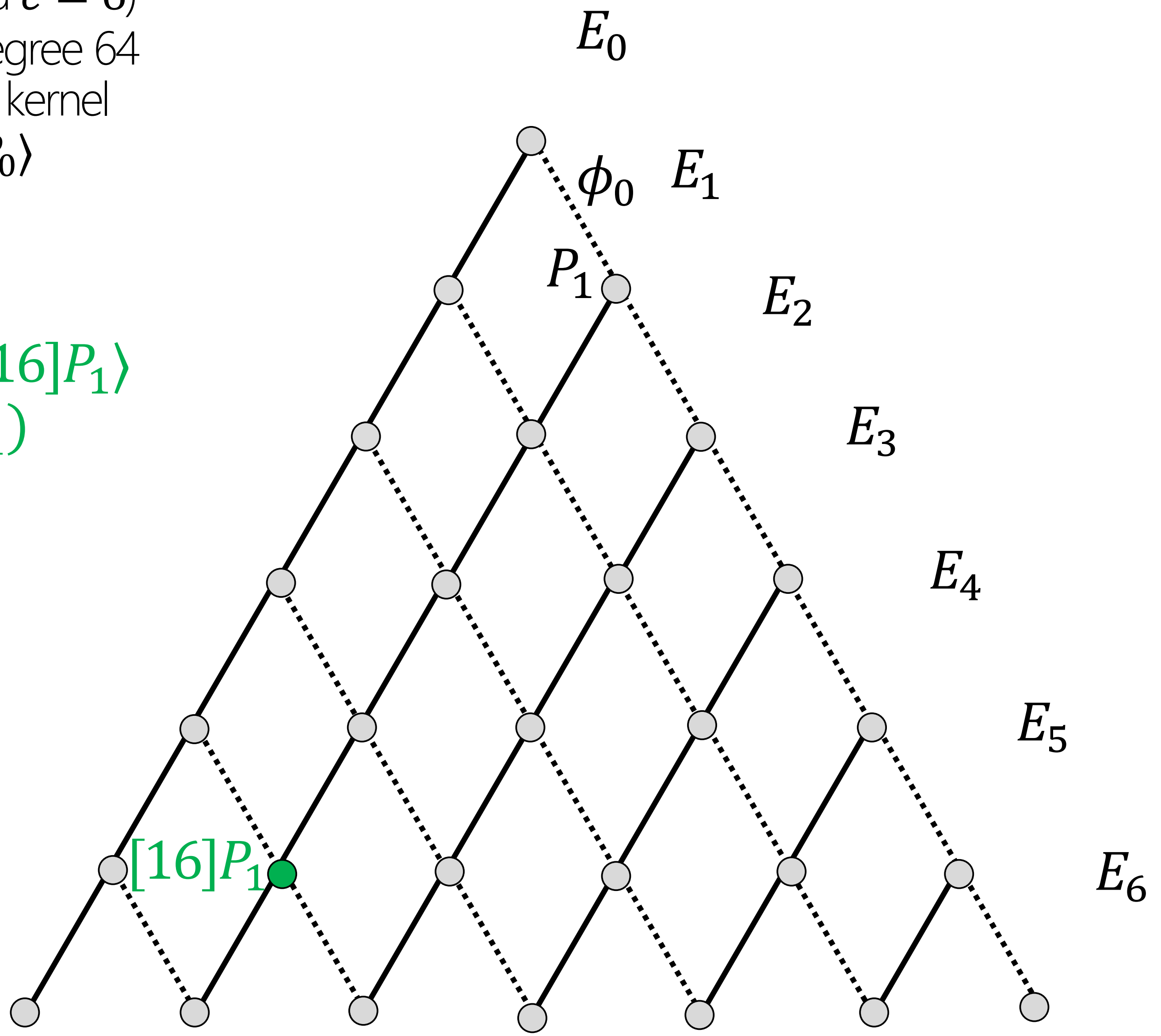
$E_3 = E_1 / \langle [8]P_1 \rangle$



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

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

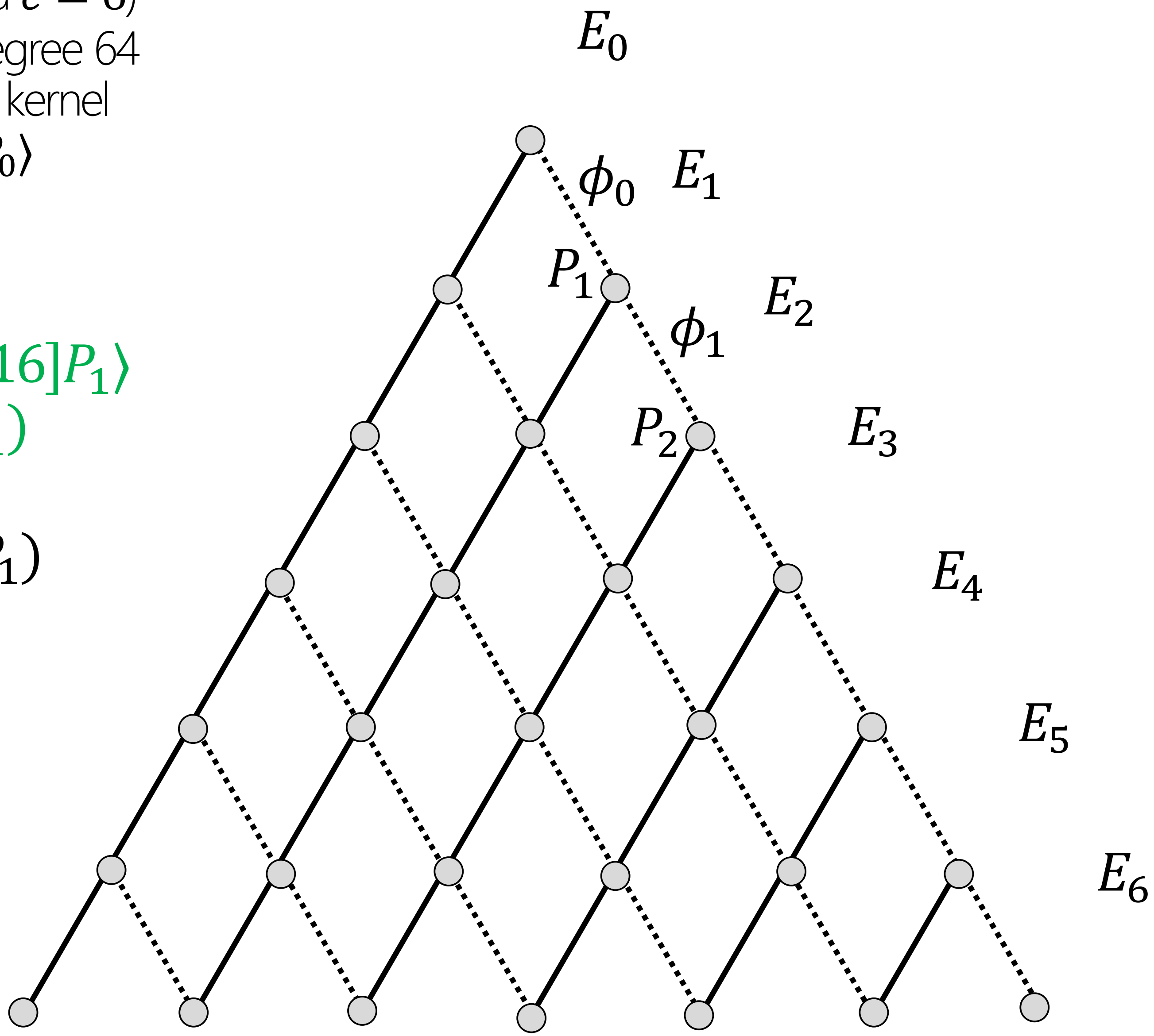


Computing ℓ^e degree isogenies

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

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

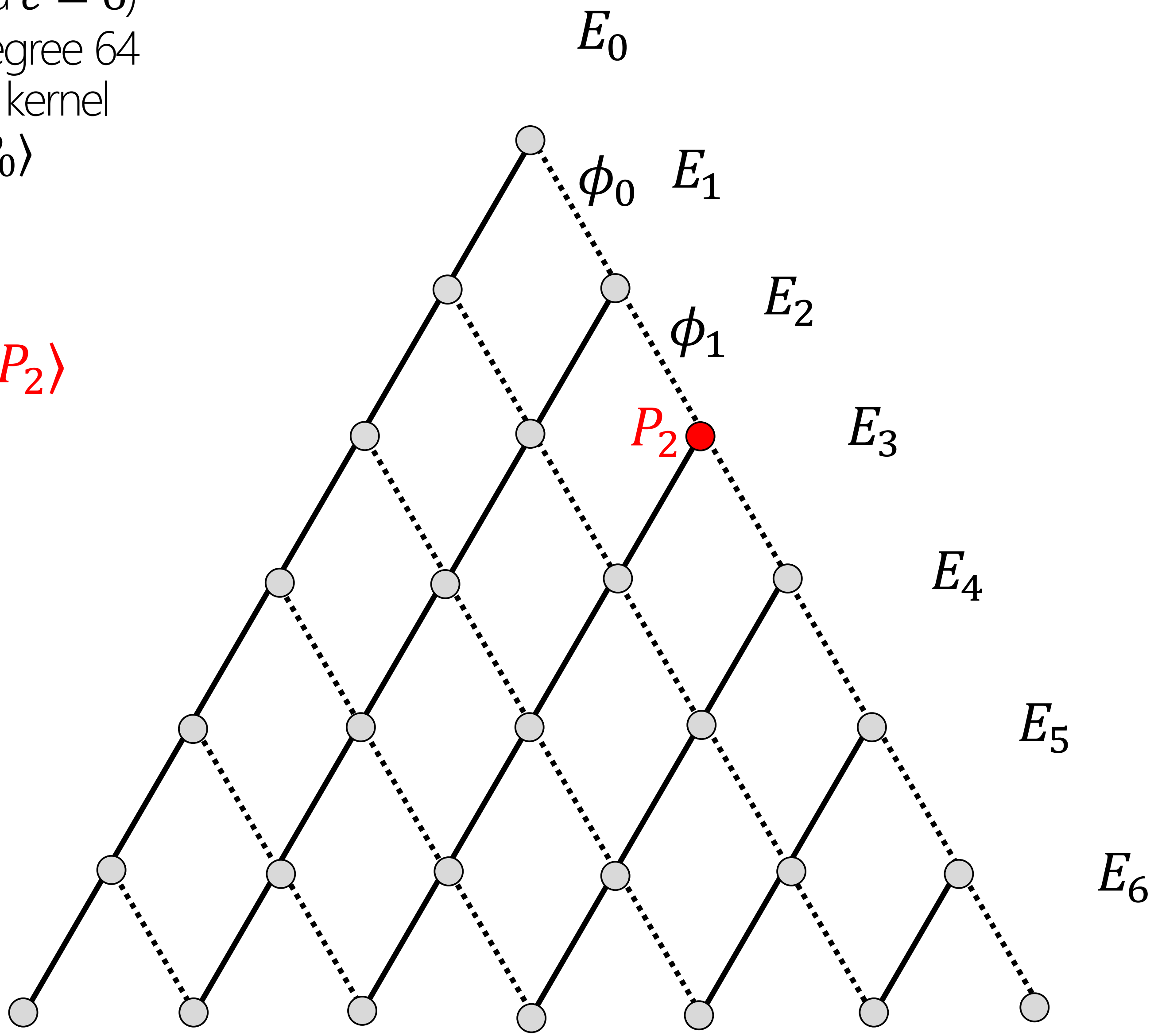
$P_2 = \phi_1(P_1)$



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

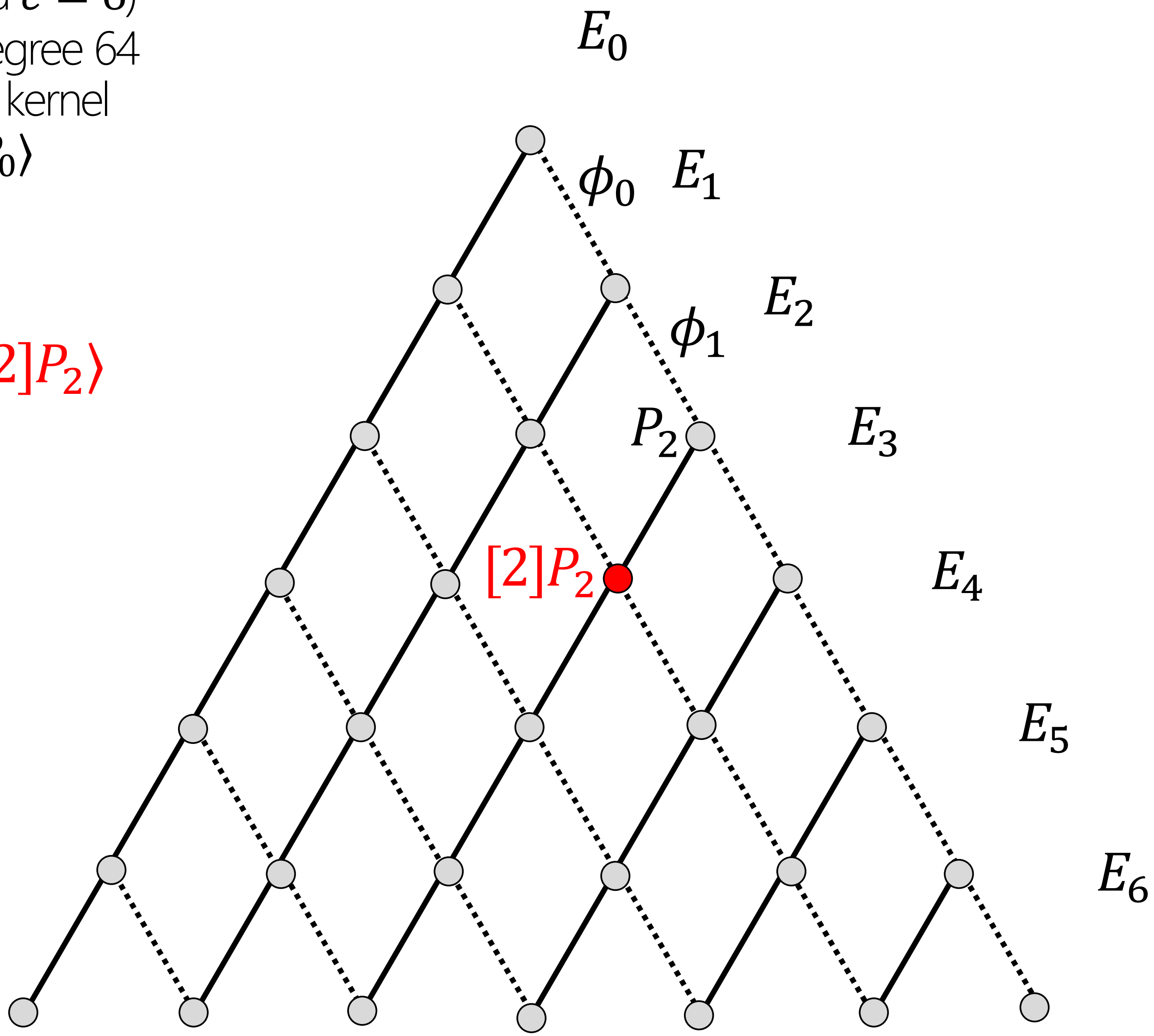
(suppose $\ell = 2$ and $e = 6$)

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

64 elements in its kernel

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

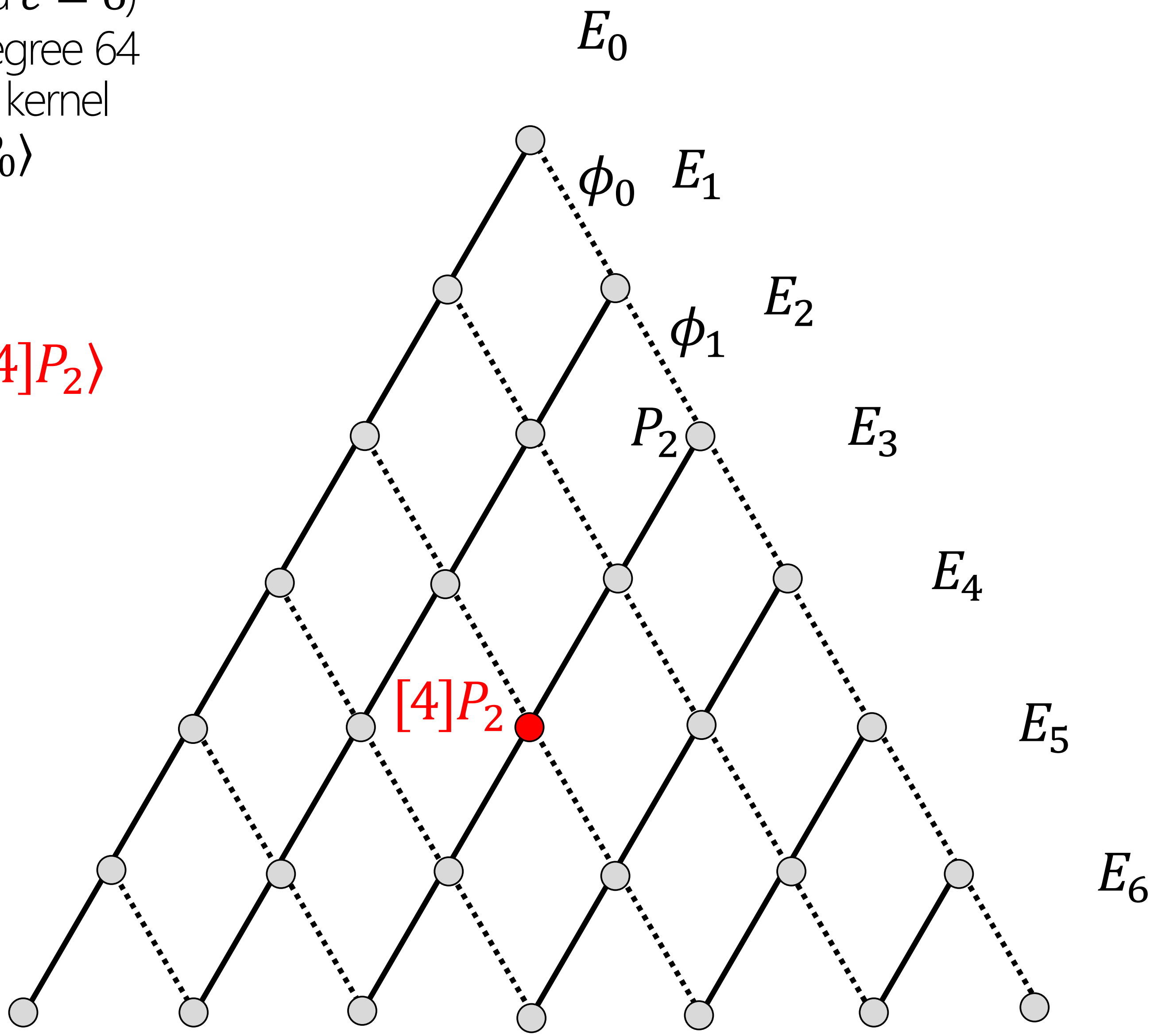
$E_5 = E_2 / \langle [2]P_2 \rangle$



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

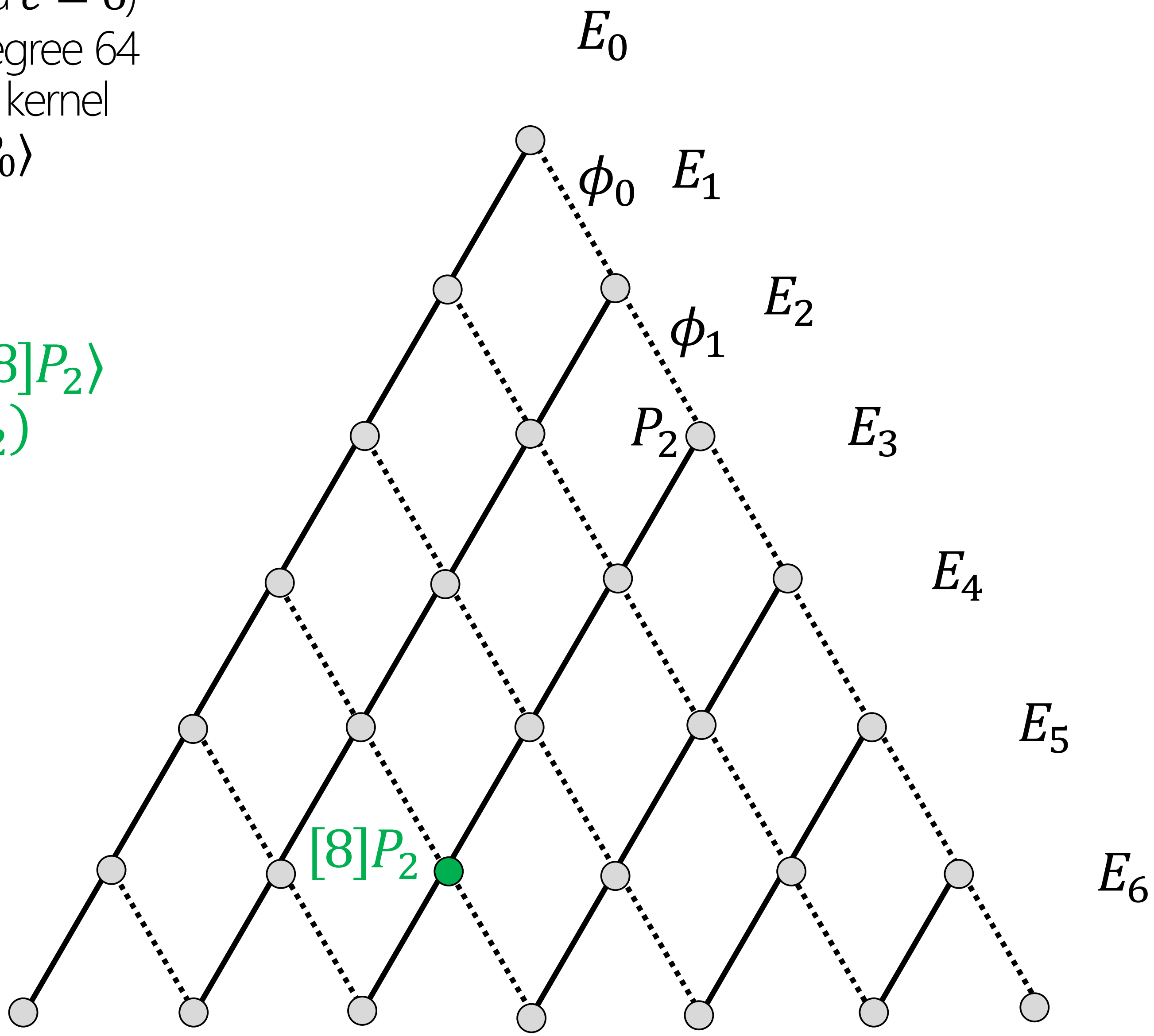
$E_4 = E_2 / \langle [4]P_2 \rangle$



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

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

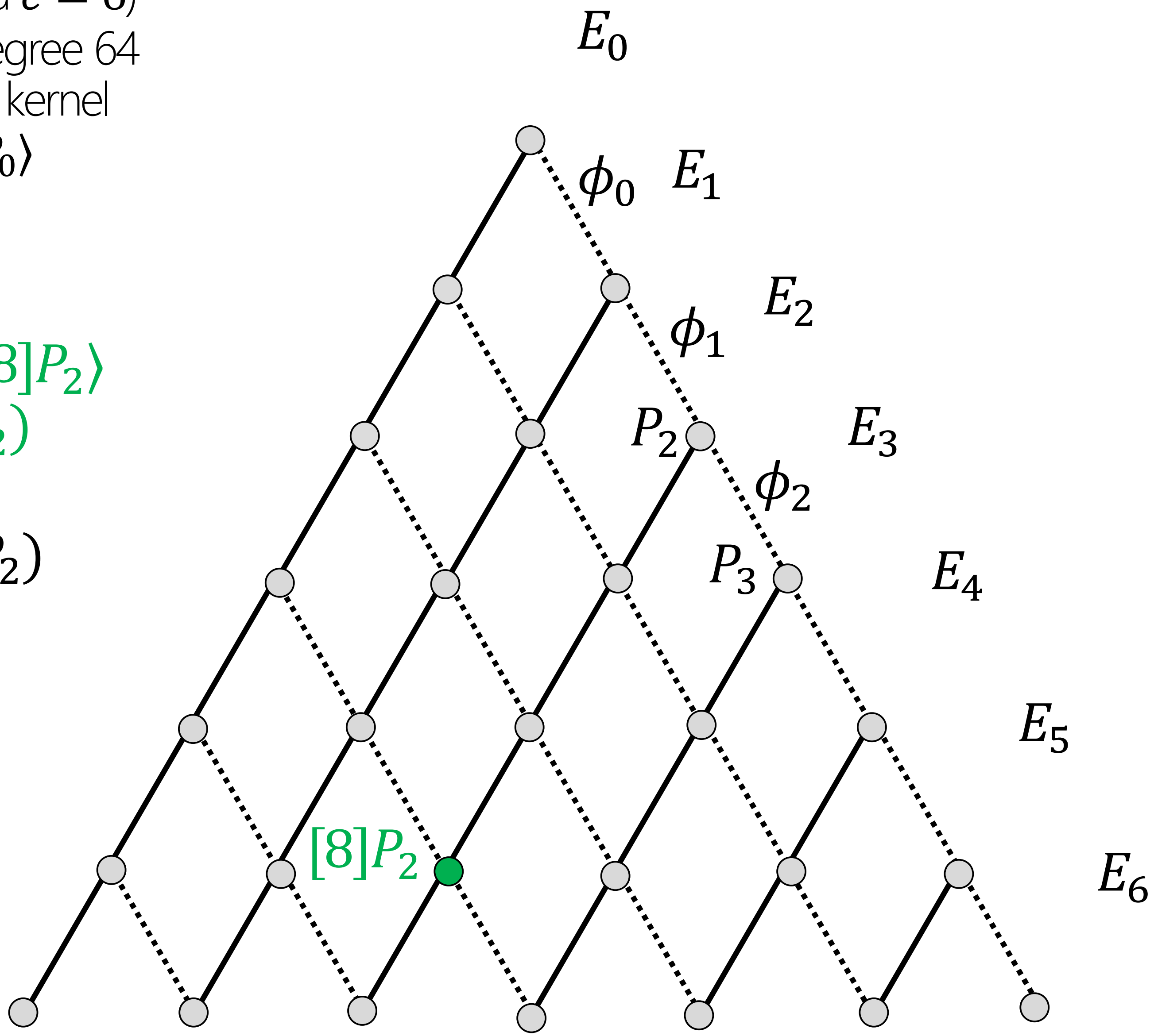


Computing ℓ^e degree isogenies

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

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

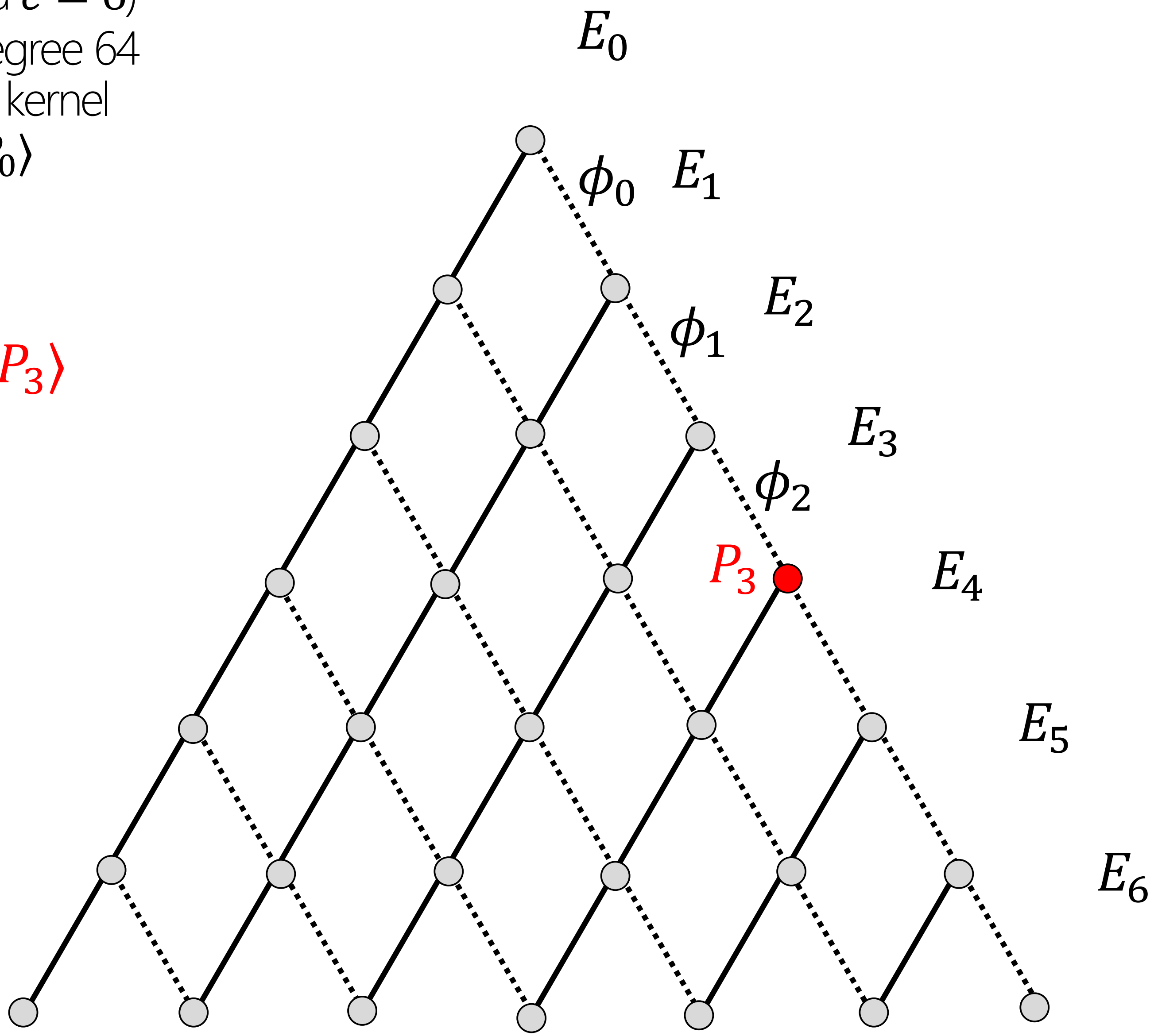
$P_3 = \phi_2(P_2)$



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

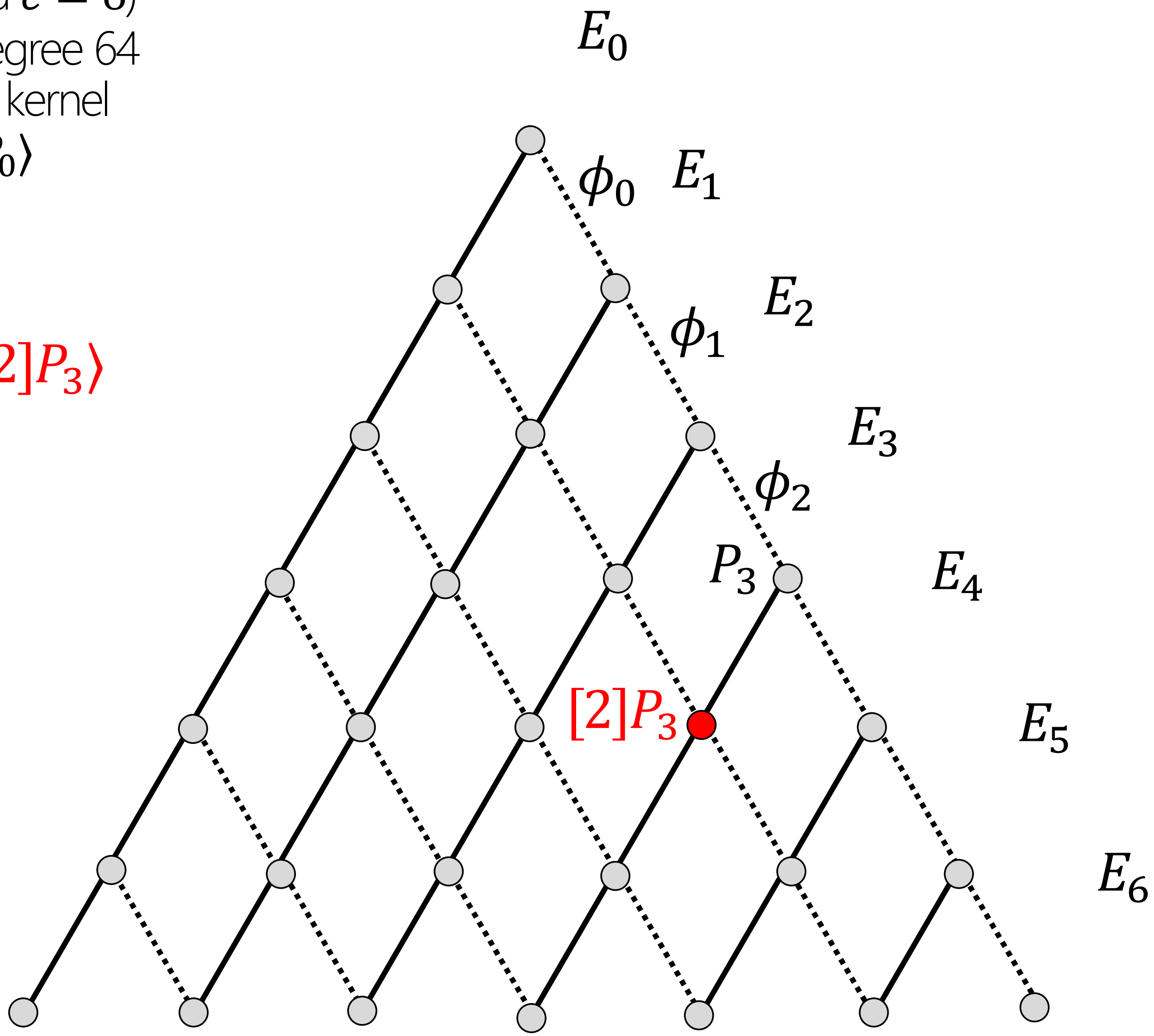
(suppose $\ell = 2$ and $e = 6$)

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

64 elements in its kernel

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

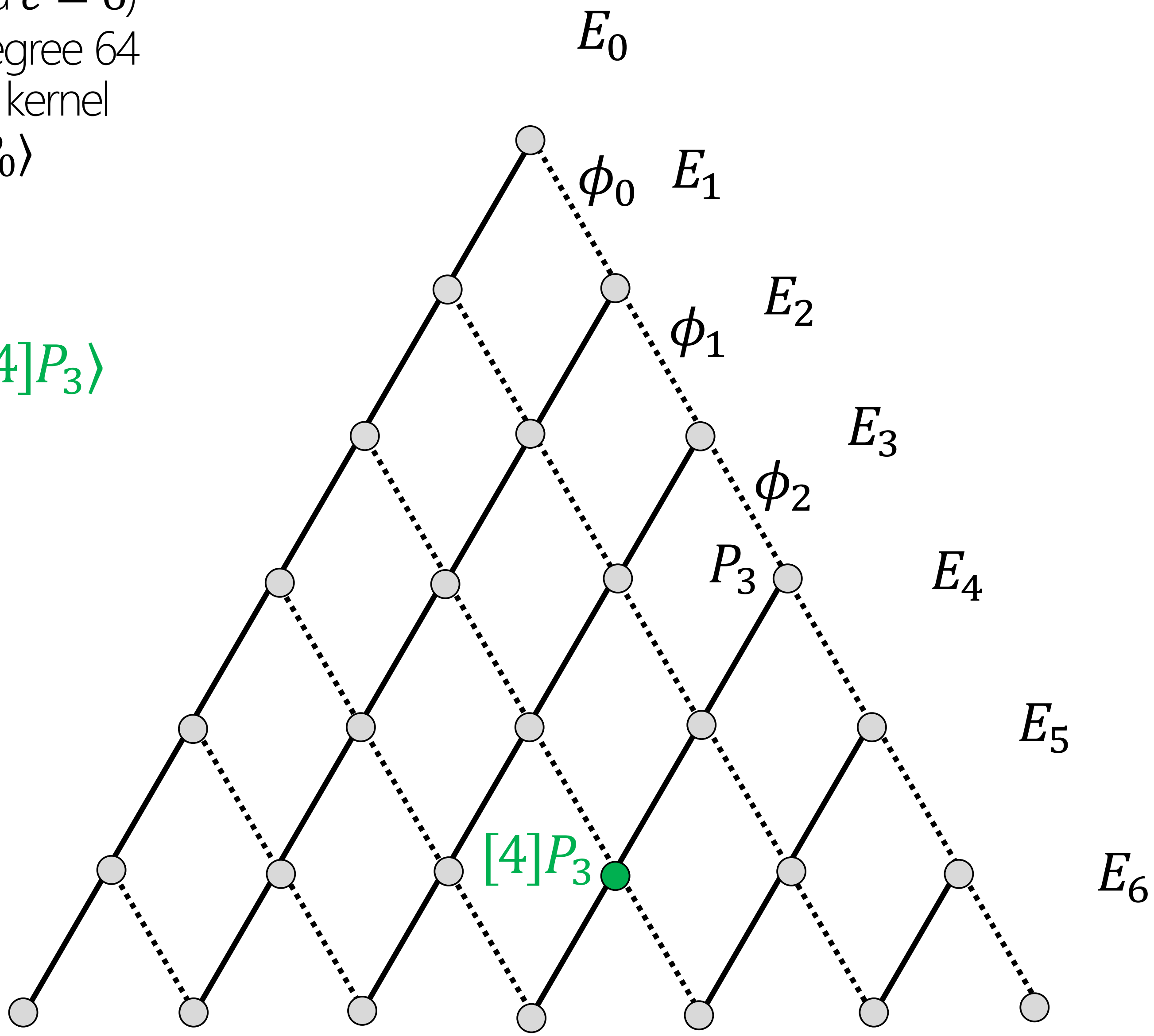
$E_5 = E_3/\langle [2]P_3 \rangle$



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

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

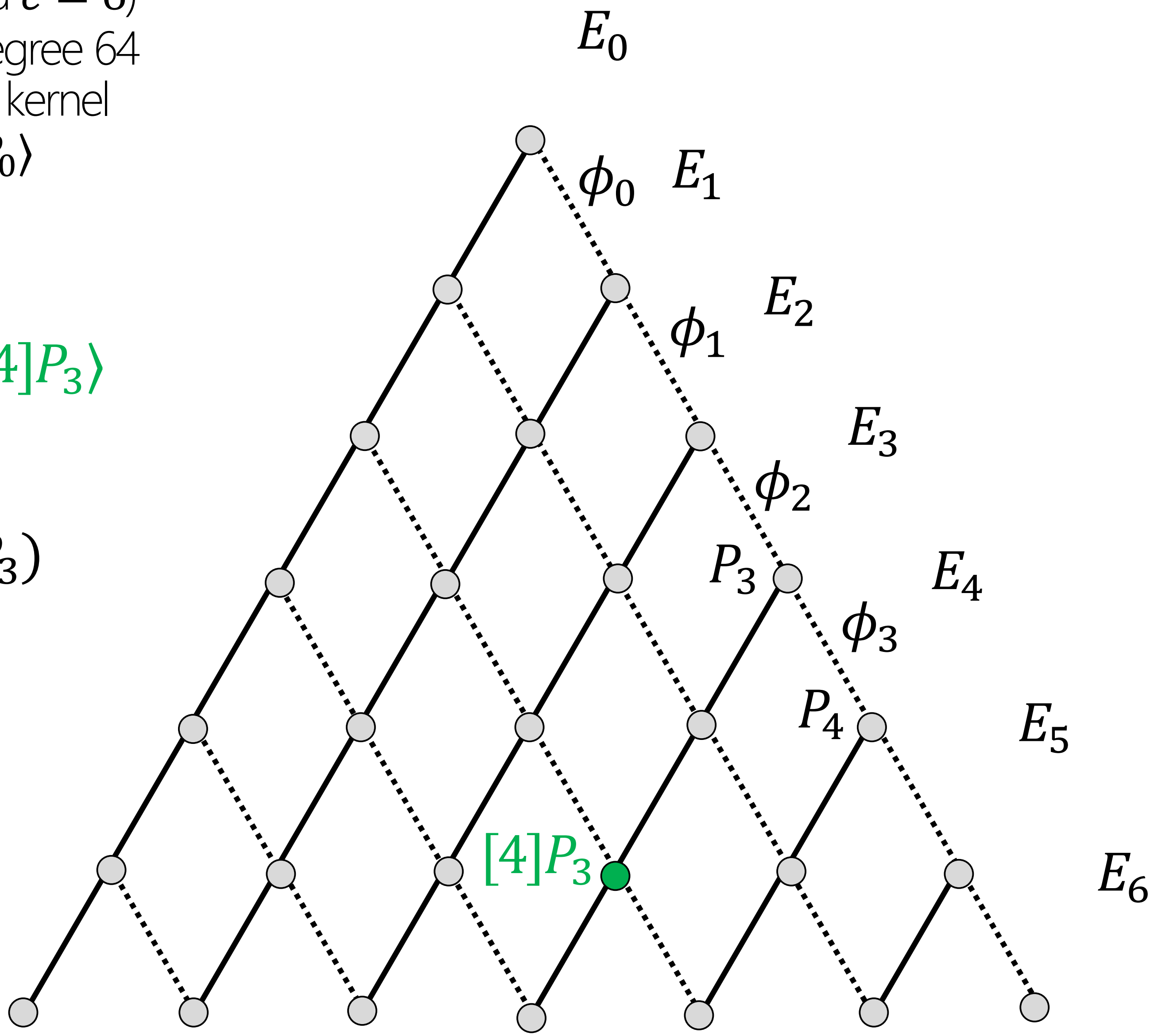


Computing ℓ^e degree isogenies

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

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

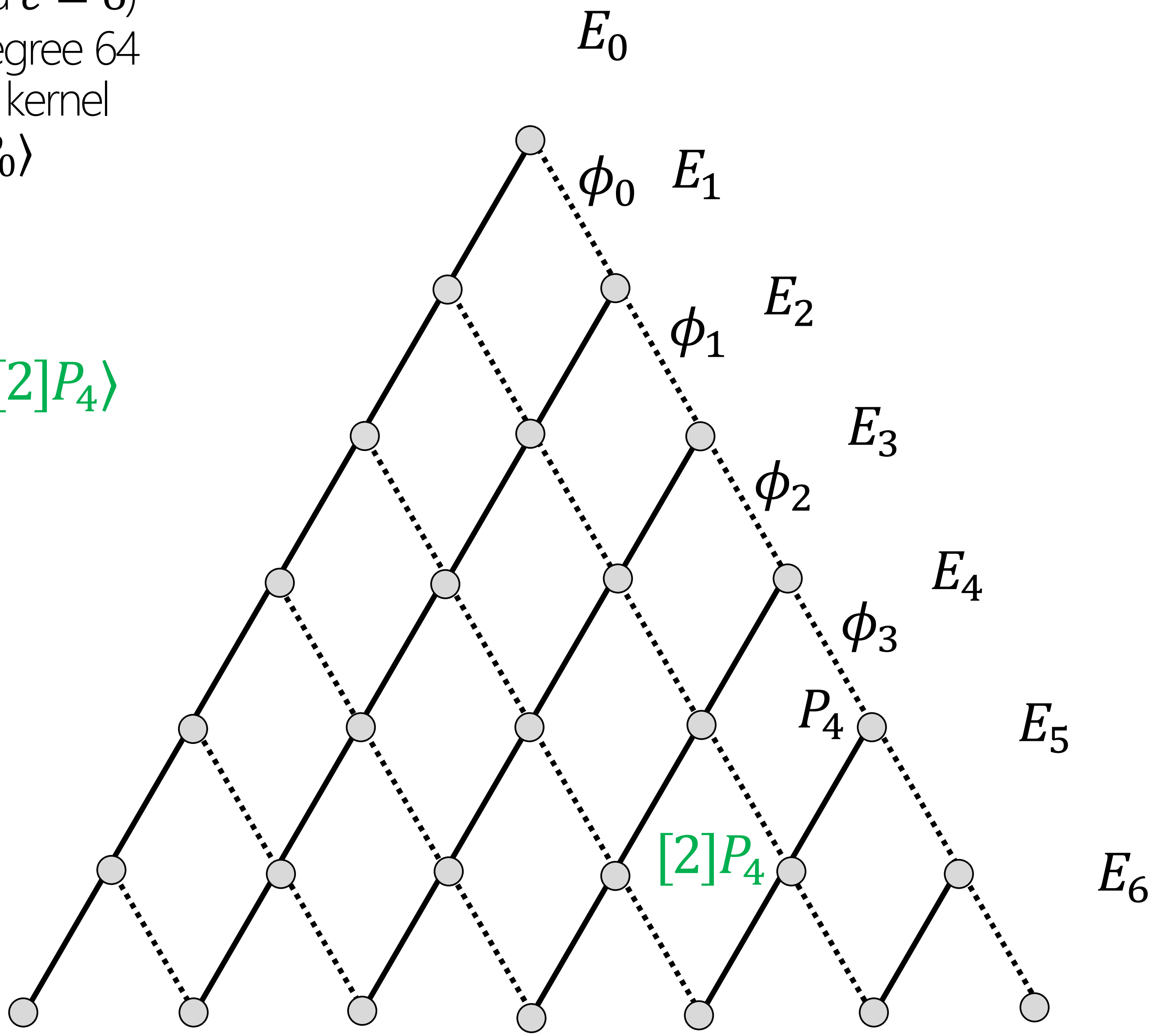
$P_4 = \phi_3(P_3)$



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

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

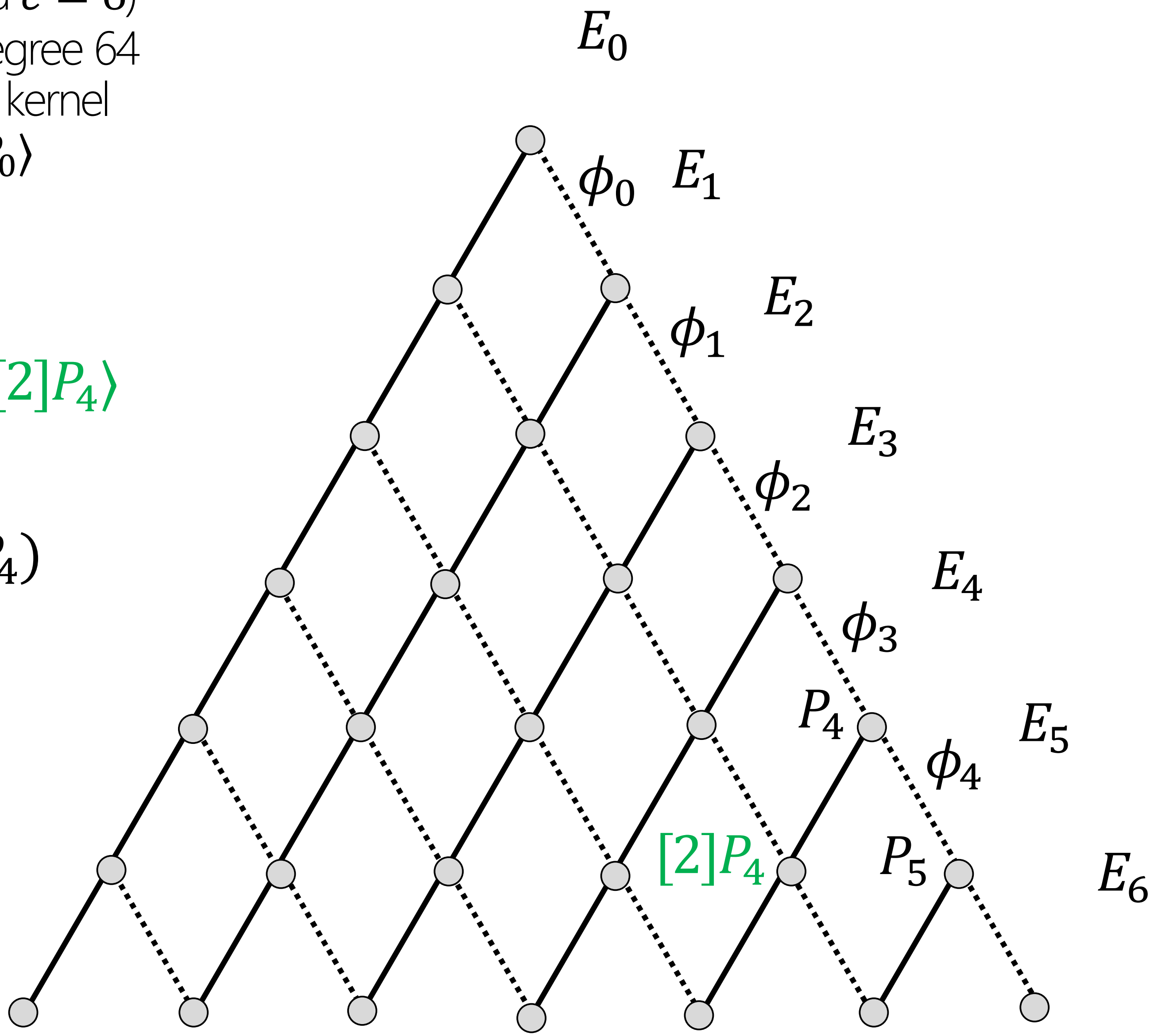


Computing ℓ^e degree isogenies

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

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

$P_5 = \phi_4(P_4)$



Computing ℓ^e degree isogenies

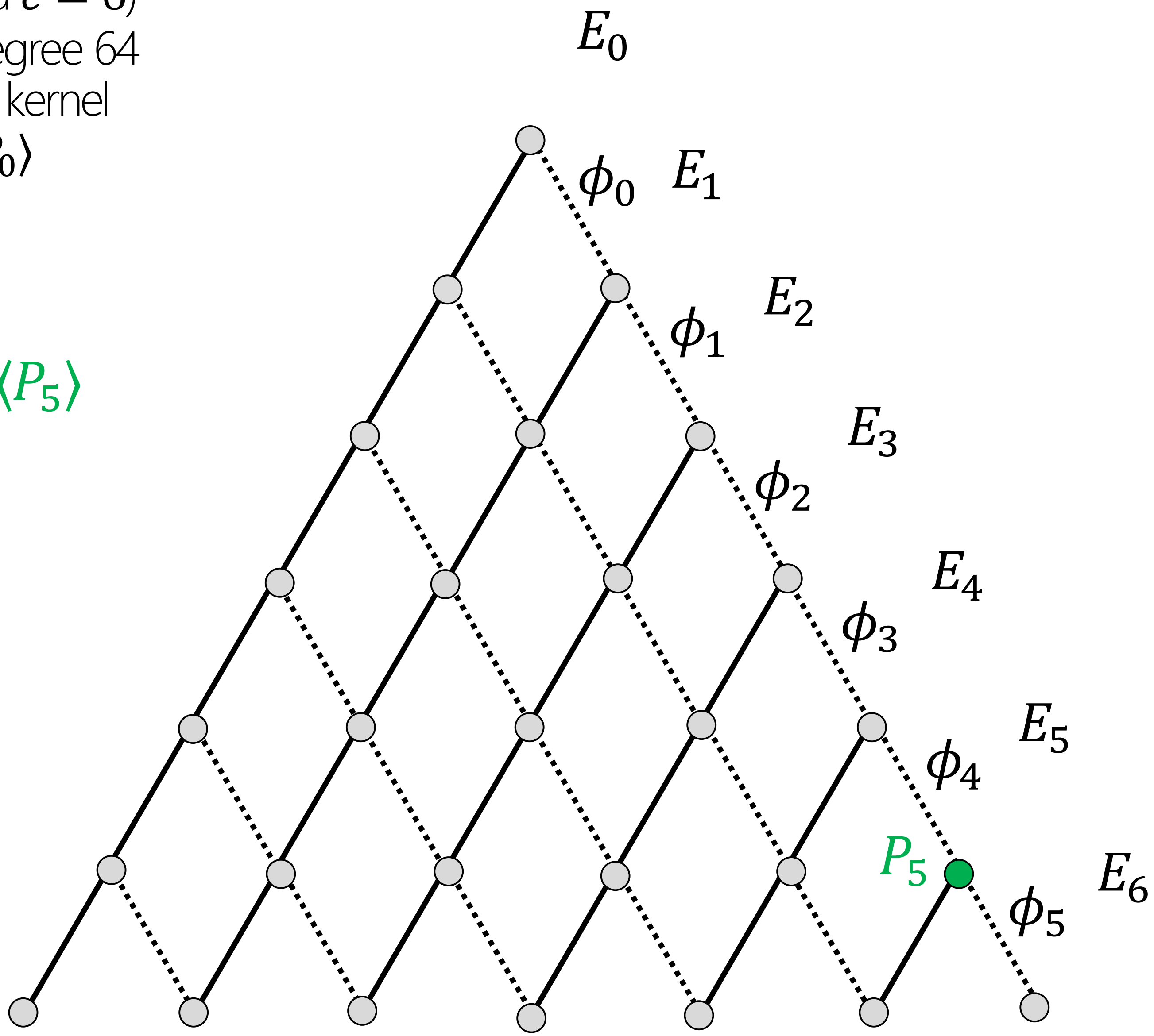
(suppose $\ell = 2$ and $e = 6$)

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

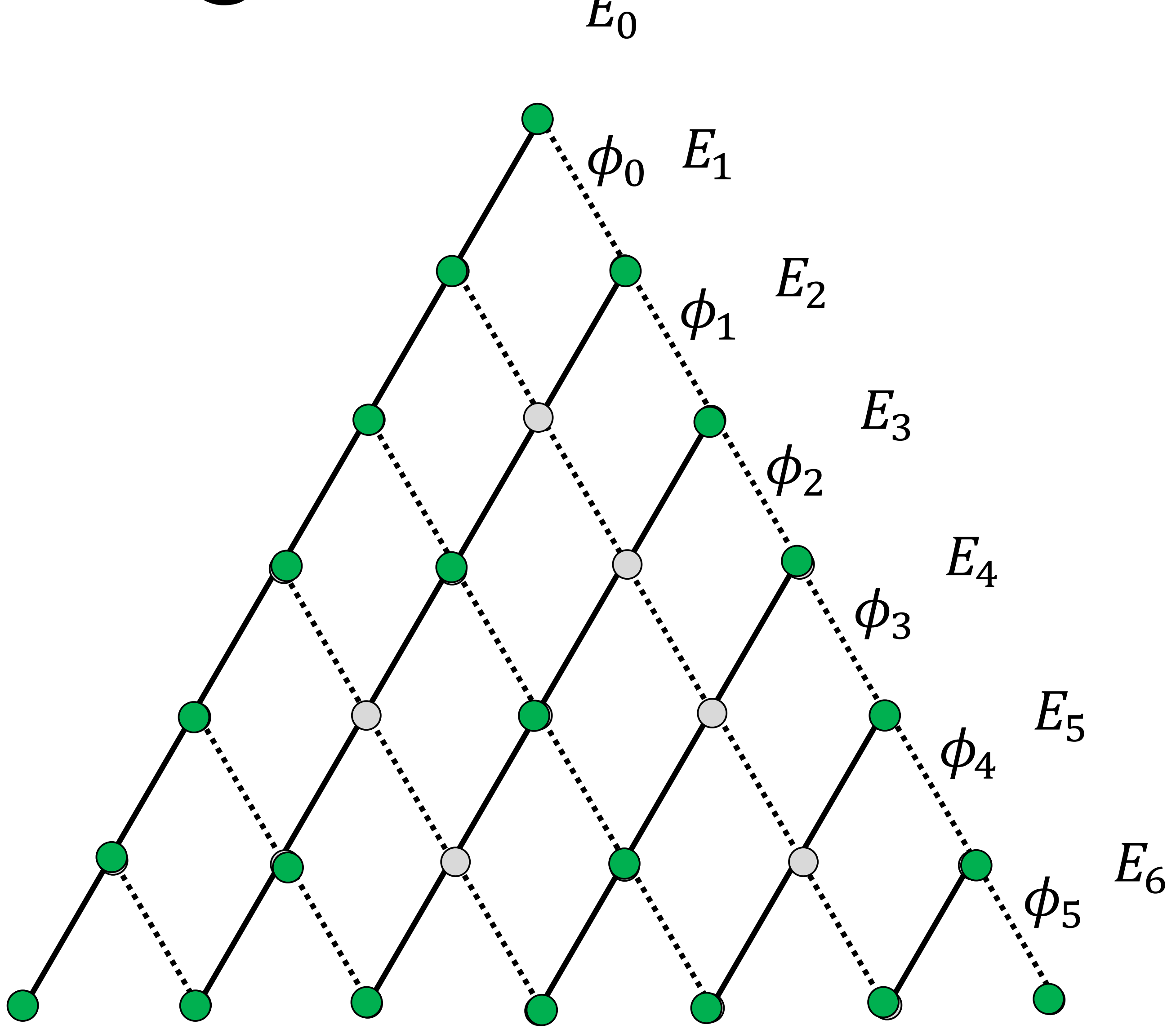
64 elements in its kernel

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

$E_6 = E_5/\langle P_5 \rangle$

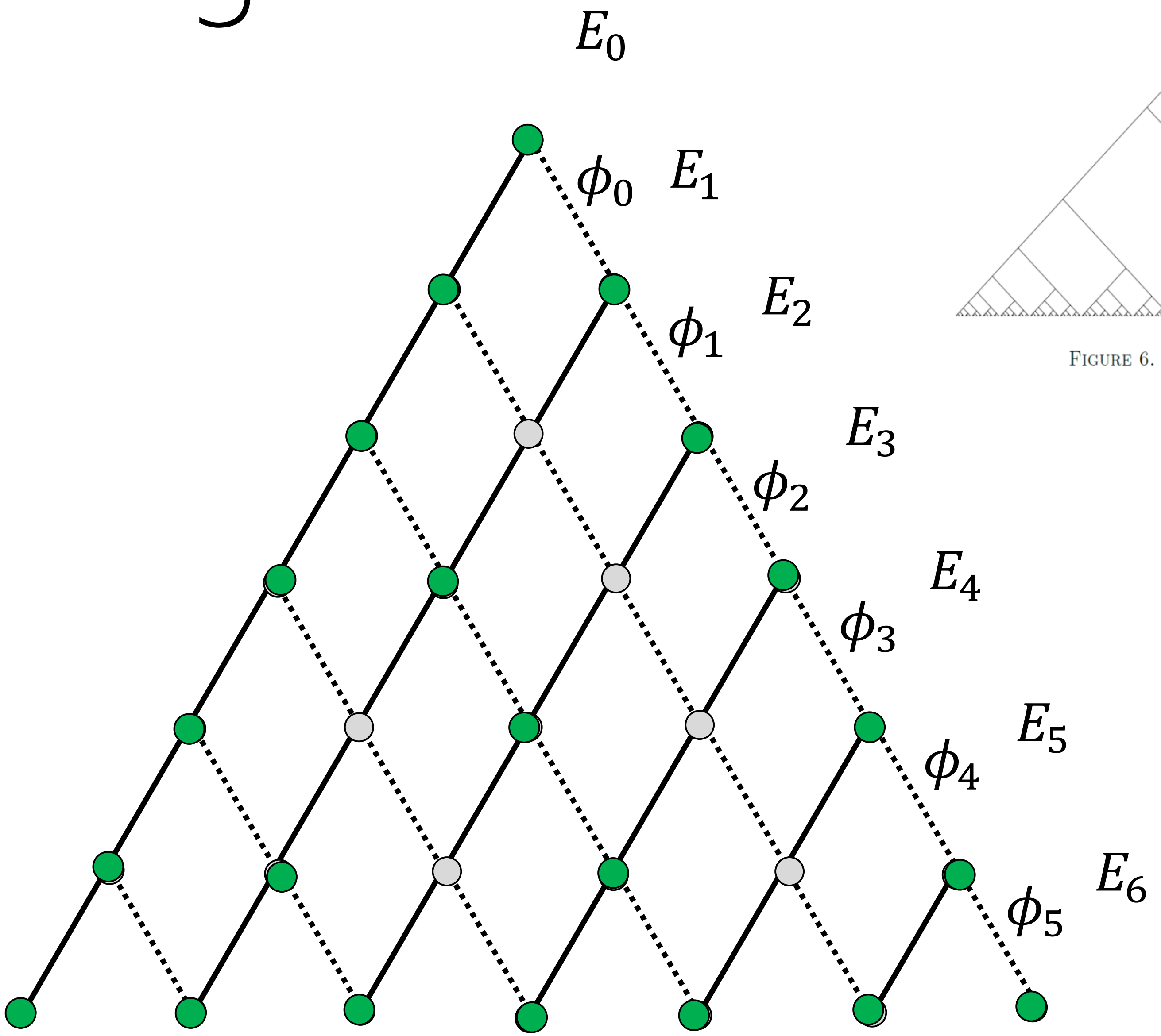


Optimal strategies



Optimal strategies

$$n^2$$
$$\rightarrow$$
$$n \log n$$



TOWARDS QUANTUM-RESISTANT CRYPTOSYSTEMS FROM ISOGENIES

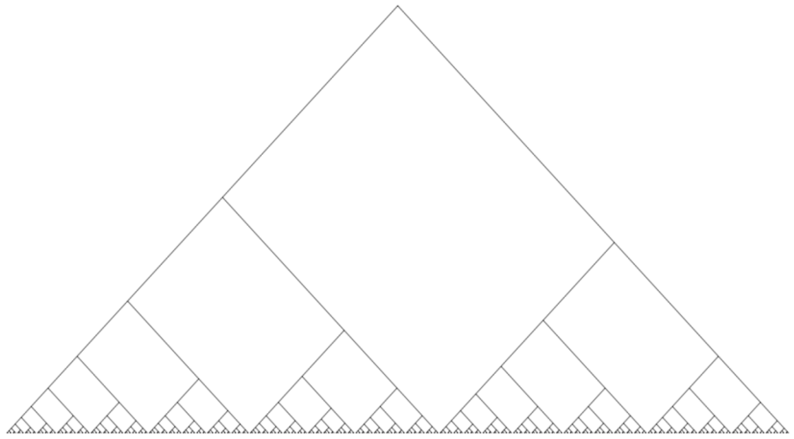
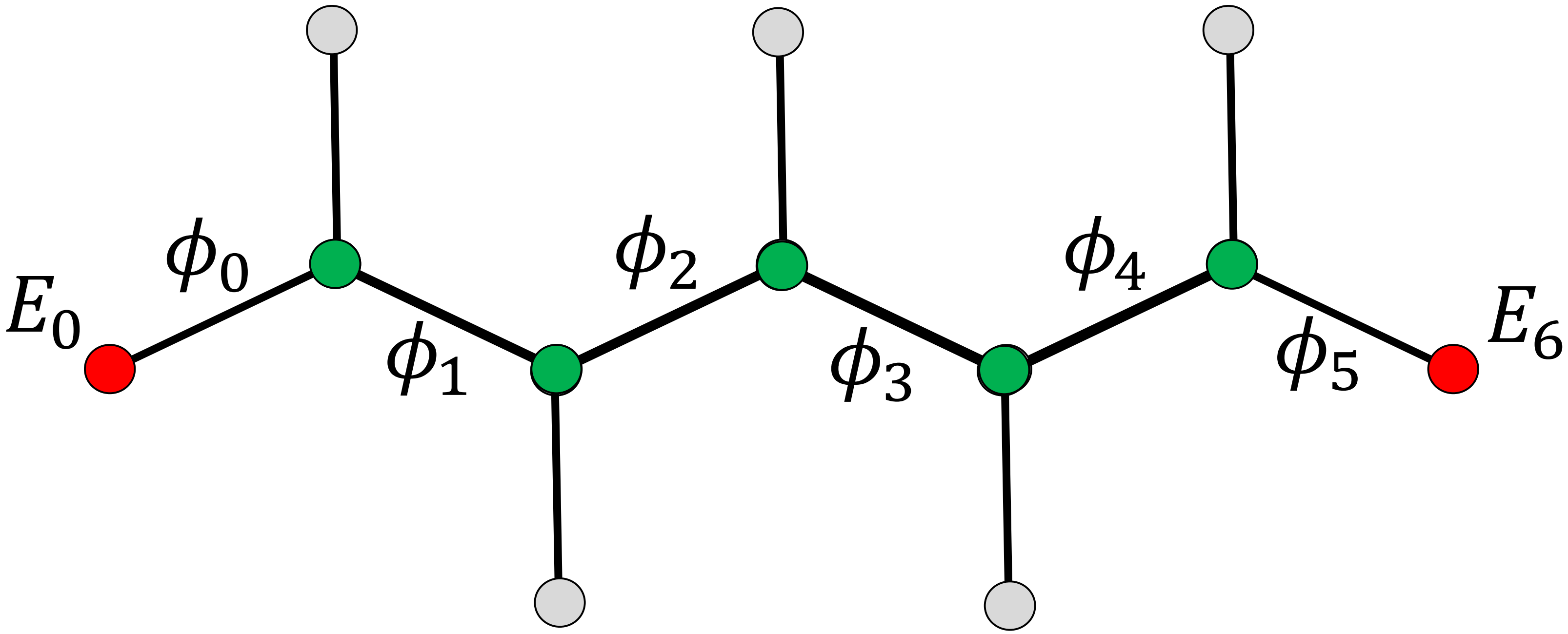


FIGURE 6. Optimal strategy for $n = 512, p = 4.6, q = 2.8$.

Computing ℓ^e degree isogenies

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

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



Rest of talk: given $\boldsymbol{E}, \boldsymbol{E'}$, find path (of known length)…

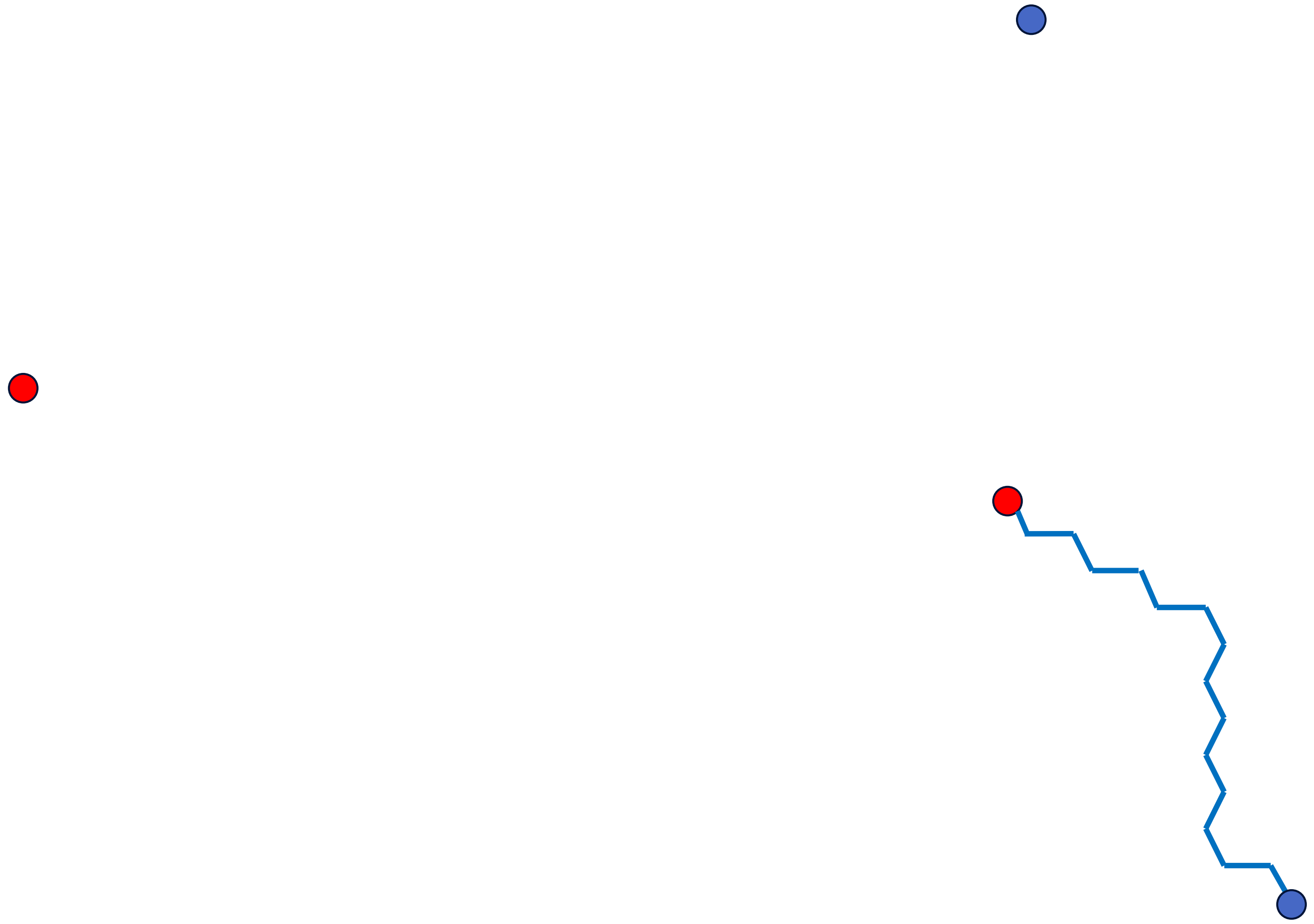
Claw algorithm: meet-in-the-middle

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

Claw algorithm: meet-in-the-middle

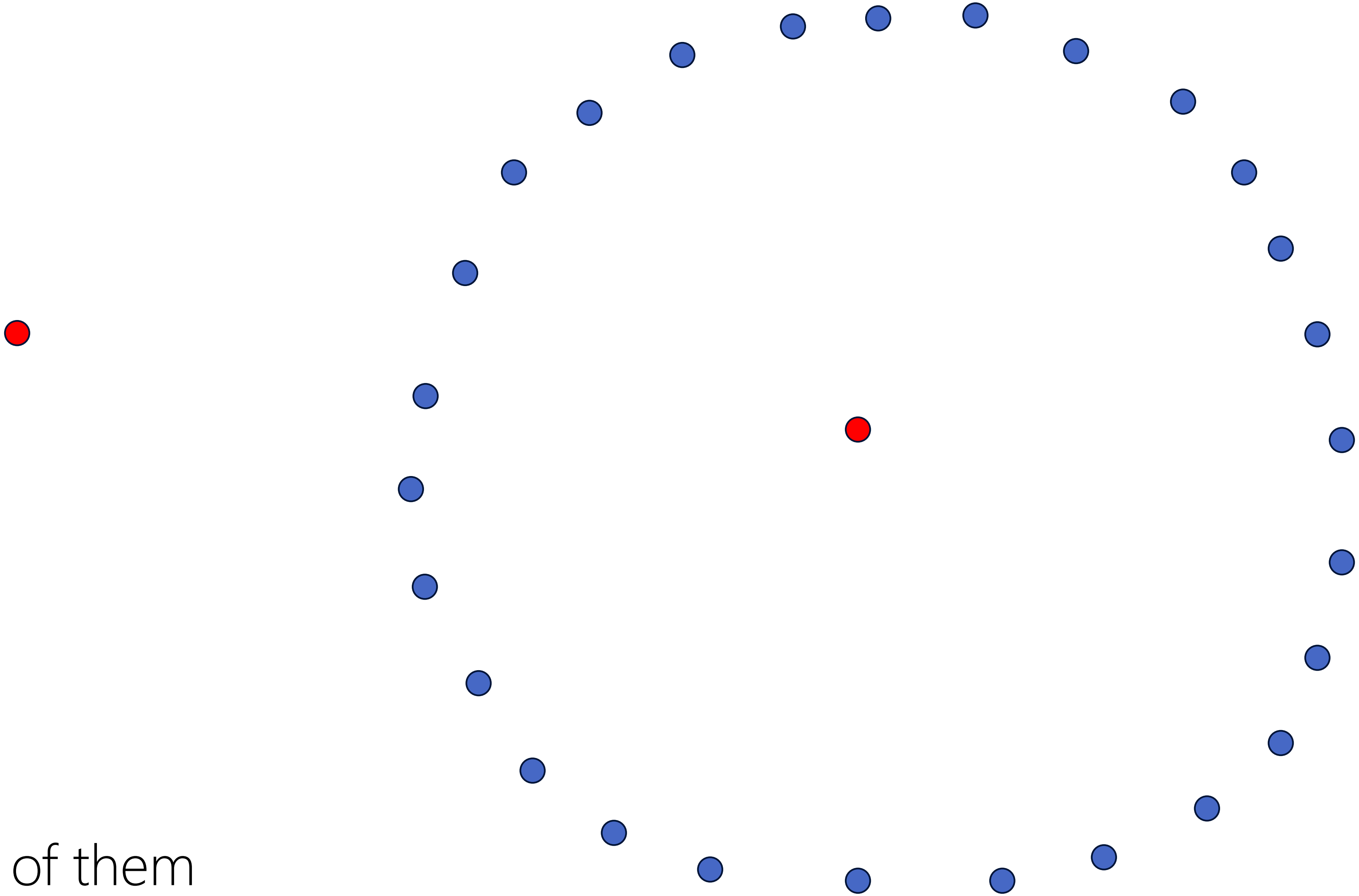
Compute and store $\ell^{e/2}$-isogenies on one side

Claw algorithm: meet-in-the-middle



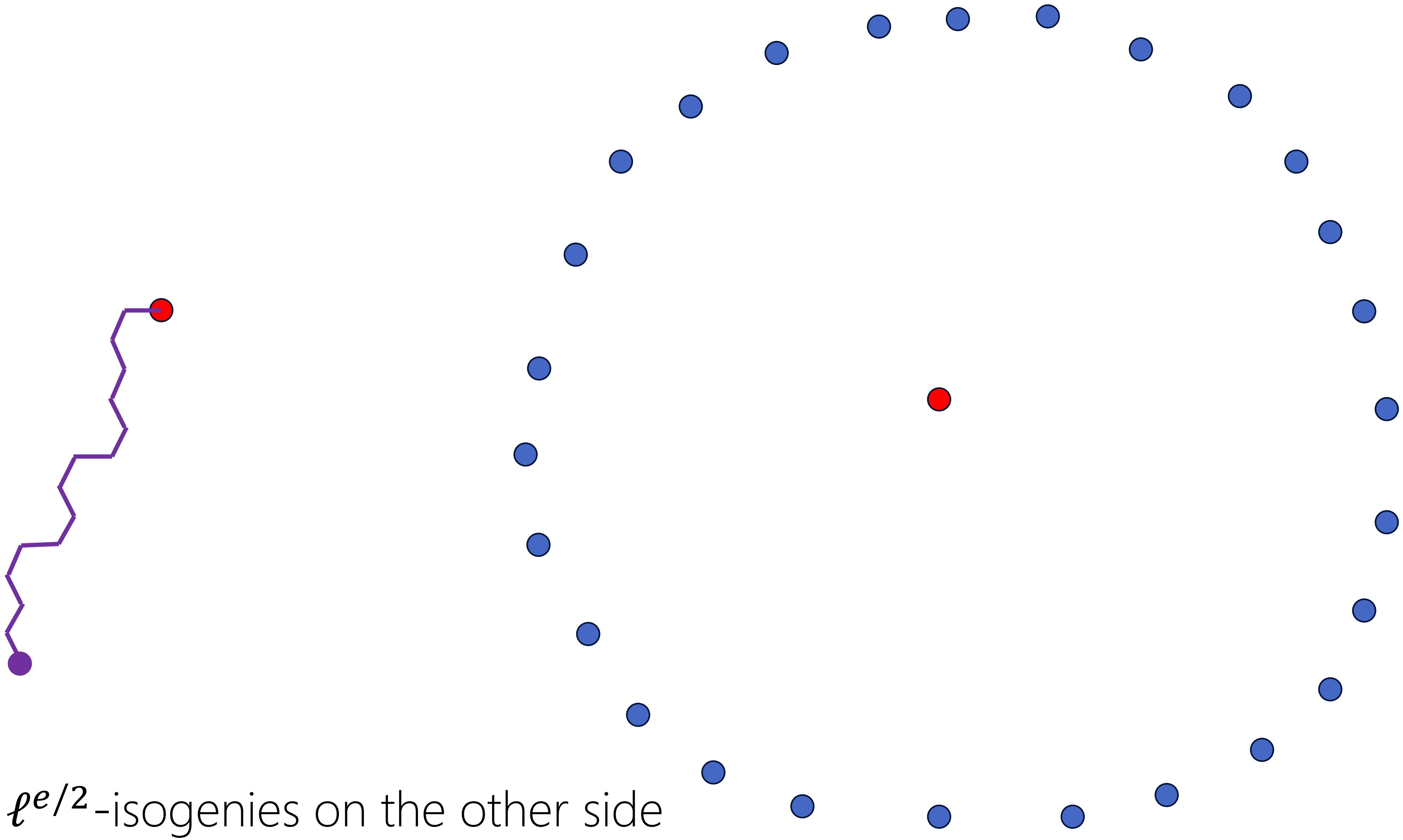
Compute and store $\ell^{e/2}$-isogenies on one side

Claw algorithm: meet-in-the-middle



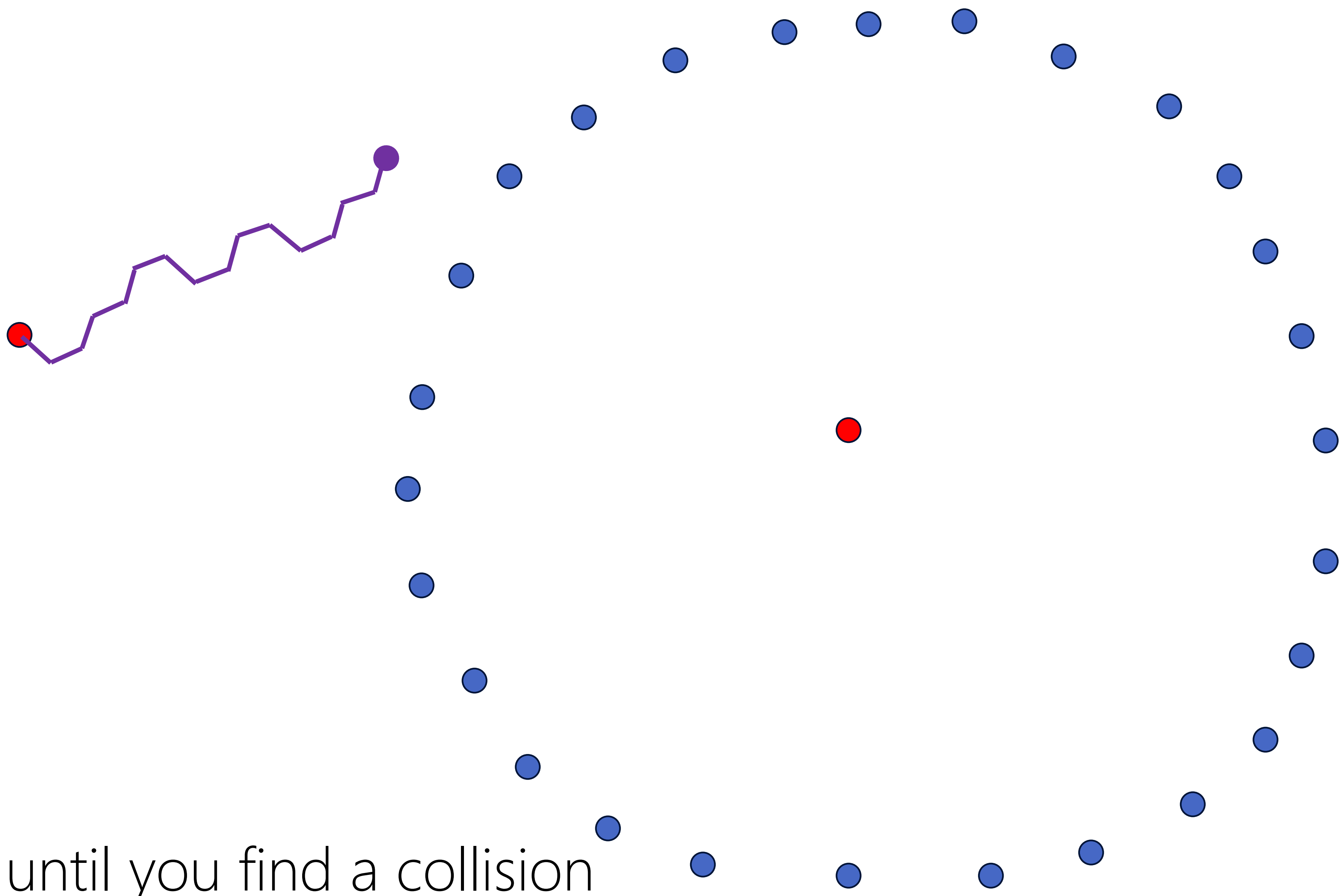
… until you have all of them

Claw algorithm: meet-in-the-middle



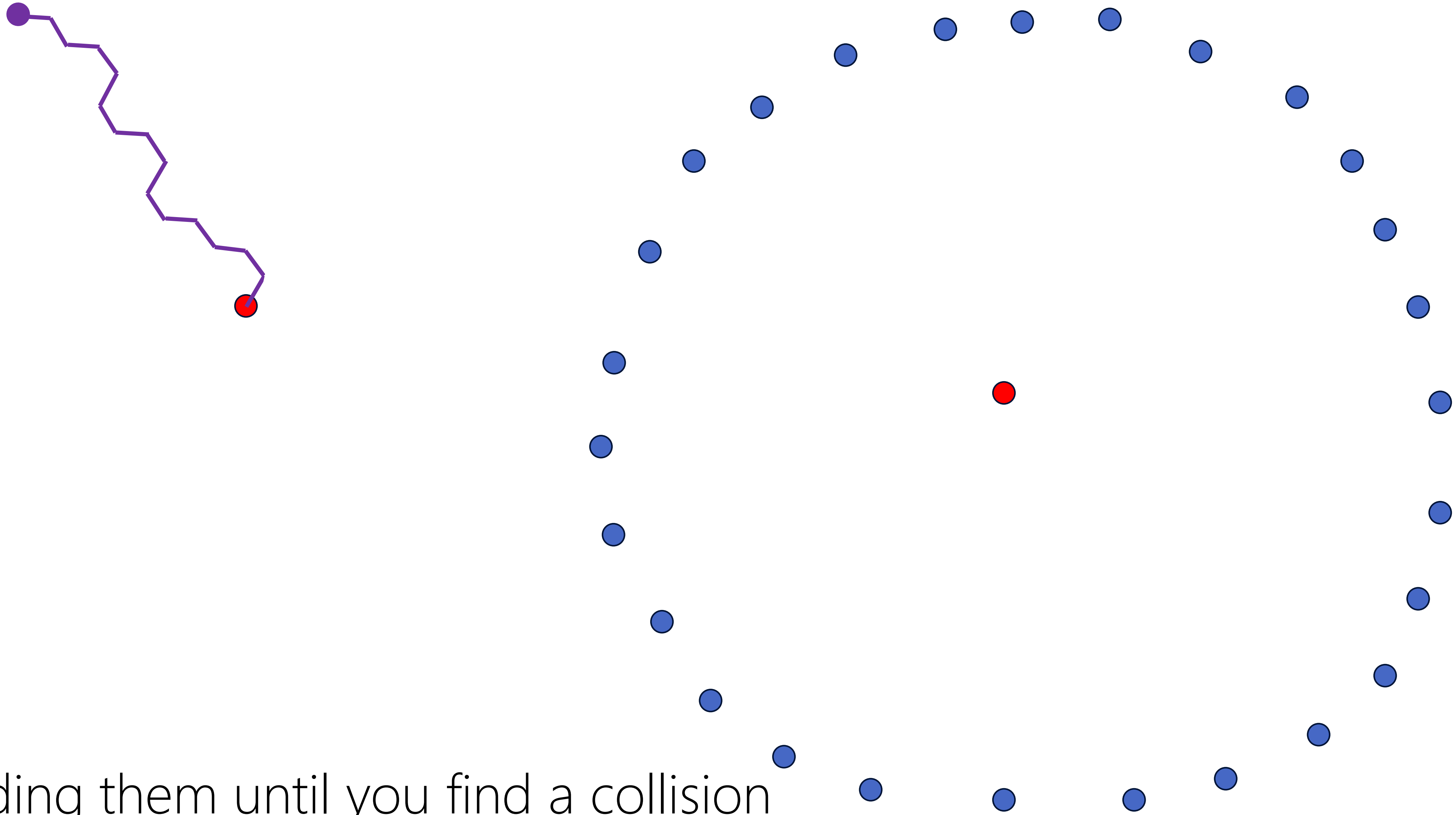
Now compute $\ell^{e/2}$-isogenies on the other side

Claw algorithm: meet-in-the-middle



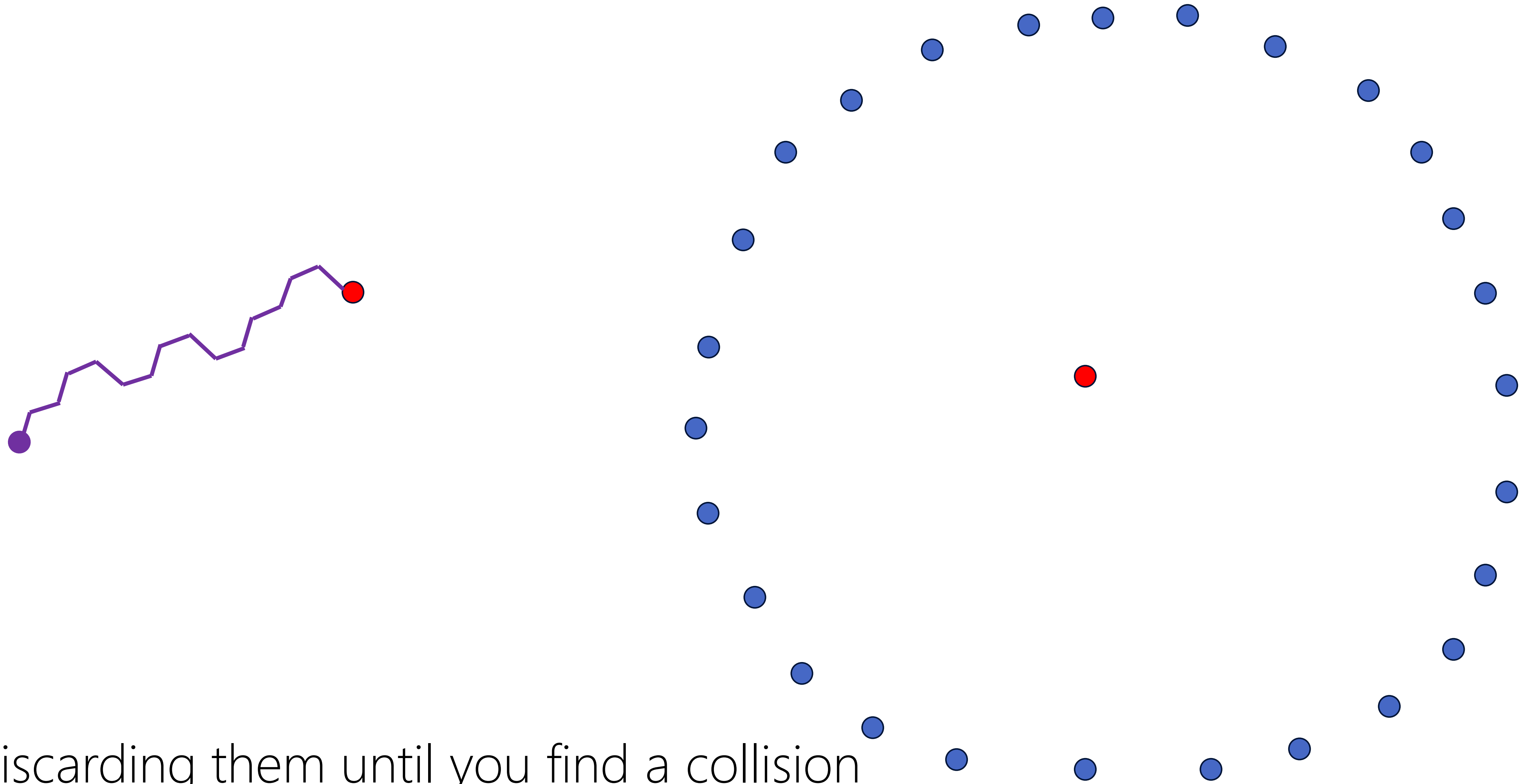
... discarding them until you find a collision

Claw algorithm: meet-in-the-middle



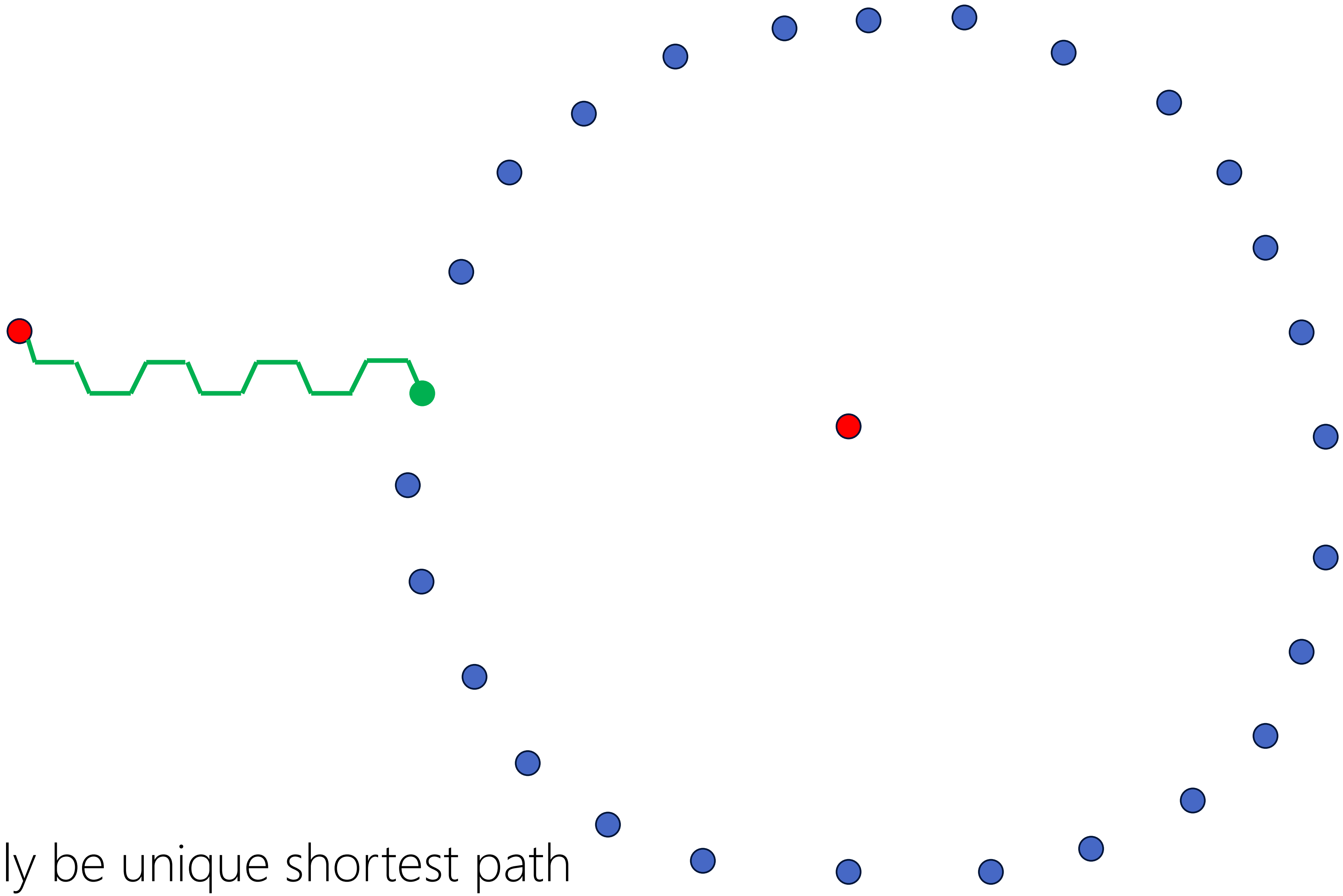
… discarding them until you find a collision

Claw algorithm: meet-in-the-middle



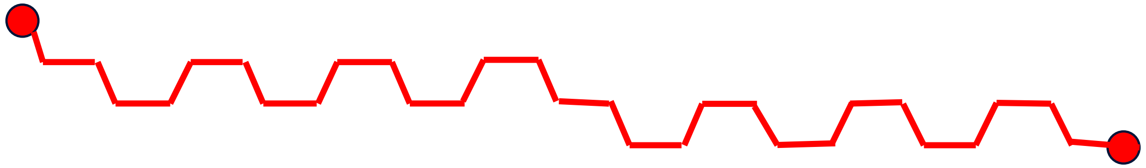
... discarding them until you find a collision

Claw algorithm: meet-in-the-middle



Collision will most likely be unique shortest path

Claw algorithm: meet-in-the-middle



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

NIST security levels

1) Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g. AES128)
2) Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for collision search on a 256-bit hash function (e.g. SHA256/ SHA3-256)
3) Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 192-bit key (e.g. AES192)
4) Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for collision search on a 384-bit hash function (e.g. SHA384/ SHA3-384)
5) Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 256-bit key (e.g. AES 256)

The curves and their security estimates

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

Target Security Level	**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** $(\sqrt[3]{2^{e_2}}, \sqrt[3]{3^{e_3}})$
NIST 1	SIKEp503	(250,159)	128	2^{127}	2^{125}	2^{64}	2^{83}
NIST 3	SIKEp761	(372,239)	192	2^{191}	2^{186}	2^{96}	2^{124}
NIST 5	SIKEp964	(486,301)	256	2^{255}	2^{238}	2^{128}	2^{159}

classical quantum

Apples and oranges

- Our proposed level 1 ($\boldsymbol{p} \approx 2^{512}$) requires $\approx 2^{128}$ time and $\approx 2^{128}$ memory for meet-in-the-middle

- Best attacks on AES128 either $\approx 2^{128}$ time and almost no memory or (bicliques) $\approx 2^{125}$ and $\approx 2^{32}$ memory

- Unfair comparison: 2^{128} memory is infeasible: fix an upper-bound on memory, then analyse runtime. (vOW, DJB, Adj et al...)

Van Oorschot – Wiener (vOW) meets isogenies

Parallel Collision Search with Cryptanalytic Applications

Paul C. van Oorschot and *Michael J. Wiener*

Nortel, P.O. Box 3511 Station C, Ottawa, Ontario, K1Y 4H7, Canada

1996 September 23

Abstract. A simple new technique of parallelizing methods for solving search problems which seek collisions in pseudo-random walks is presented. This technique can be adapted to a wide range of cryptanalytic problems which can be reduced to finding collisions. General constructions are given showing how to adapt the technique to finding discrete logarithms in cyclic groups, finding meaningful collisions in hash functions, and performing meet-in-the-middle attacks such as a known-plaintext attack on double encryption. The new technique greatly extends the reach of practical attacks, providing the most cost-effective means known to date for defeating: the small subgroup used in certain schemes based on discrete logarithms such as Schnorr, DSA, and elliptic curve cryptosystems; hash functions such as MD5, RIPEMD, SHA-1, MDC-2, and MDC-4; and double encryption and three-key triple encryption. The practical significance of the technique is illustrated by giving the design for three \$10 million custom machines which could be built with current technology: one finds elliptic curve logarithms in $GF(2^{155})$ thereby defeating a proposed elliptic curve cryptosystem in expected time 32 days, the second finds MD5 collisions in expected time 21 days, and the last recovers a double-DES key from 2 known plaintexts in expected time 4 years, which is four orders of magnitude faster than the conventional meet-in-the-middle attack on double-DES. Based on this attack, double-DES offers only 17 more bits of security than single-DES.

Key words. parallel collision search, cryptanalysis, discrete logarithm, hash collision, meet-in-the-middle attack, double encryption, elliptic curves.

1. Introduction

The power of parallelized attacks has been illustrated in work on integer factorization and cryptanalysis of DES. In the factoring of the RSA-129 challenge number and other factoring efforts (e.g. [26, 27]), the sieving process was distributed among a large number of workstations. Similar efforts have been undertaken on large parallel machines [14, 19]. In an exhaustive key search attack proposed for DES [44], a large number of inexpensive specialized processors were proposed to achieve a high degree of parallelism. In this paper, we provide a method for efficient parallelization of collision search techniques.[1]

[1] Preliminary versions of parts of this work have appeared in the proceedings of the Second ACM Conference on Computer and Communications Security [42] and in the proceedings of Crypto '96 [43].

ON THE COST OF COMPUTING ISOGENIES BETWEEN SUPERSINGULAR ELLIPTIC CURVES

GORA ADJ, DANIEL CERVANTES-VÁZQUEZ, JESÚS-JAVIER CHI-DOMÍNGUEZ, ALFRED MENEZES, AND FRANCISCO RODRÍGUEZ-HENRÍQUEZ

ABSTRACT. The security of the Jao-De Feo Supersingular Isogeny Diffie-Hellman (SIDH) key agreement scheme is based on the intractability of the Computational Supersingular Isogeny (CSSI) problem — computing $\mathbb{F}_{p^2}$-rational isogenies of degrees 2^e and 3^e between certain supersingular elliptic curves defined over $\mathbb{F}_{p^2}$. The classical meet-in-the-middle attack on CSSI has an expected running time of $O(p^{1/4})$, but also has $O(p^{1/4})$ storage requirements. In this paper, we demonstrate that the van Oorschot-Wiener golden collision finding algorithm has a lower cost (but higher running time) for solving CSSI, and thus should be used instead of the meet-in-the-middle attack to assess the security of SIDH against classical attacks. The smaller parameter p brings significantly improved performance for SIDH.

1. INTRODUCTION

The Supersingular Isogeny Diffie-Hellman (SIDH) key agreement scheme was proposed by Jao and De Feo [12] (see also [7]). It is one of 69 candidates being considered by the U.S. government's National Institute of Standards and Technology (NIST) for inclusion in a forthcoming standard for quantum-safe cryptography [11]. The security of SIDH is based on the difficulty of the Computational Supersingular Isogeny (CSSI) problem, which was first defined by Charles, Goren and Lauter [3] in their paper that introduced an isogeny-based hash function. The CSSI problem is also the basis for the security of isogeny-based signature schemes [9, 28] and an undeniable signature scheme [13].

Let p be a prime, let ℓ be a small prime (e.g., $\ell \in \{2, 3\}$), and let E and E' be two supersingular elliptic curves defined over $\mathbb{F}_{p^2}$ for which a (separable) degree-ℓ^e isogeny $\phi : E \to E'$ defined over $\mathbb{F}_{p^2}$ exists. The CSSI problem is that of constructing such an isogeny. In [7], the CSSI problem is assessed as having a complexity of $O(p^{1/4})$ and $O(p^{1/6})$ against classical and quantum attacks [23], respectively. The classical attack is a meet-in-the-middle attack (MITM) that has time complexity $O(p^{1/4})$ and space complexity $O(p^{1/4})$. We observe that the (classical) van Oorschot-Wiener golden collision finding algorithm [16, 17] can be employed to construct ϕ. Whereas the time complexity of the van Oorschot-Wiener algorithm is higher than that of the meet-in-the-middle attack, its space requirements are smaller. Our cost analysis of these two CSSI attacks leads to the conclusion that, despite its higher running time, the golden collision finding CSSI attack has a lower cost than the meet-in-the-middle attack, and thus should be used to assess the security of SIDH against (known) classical attacks.

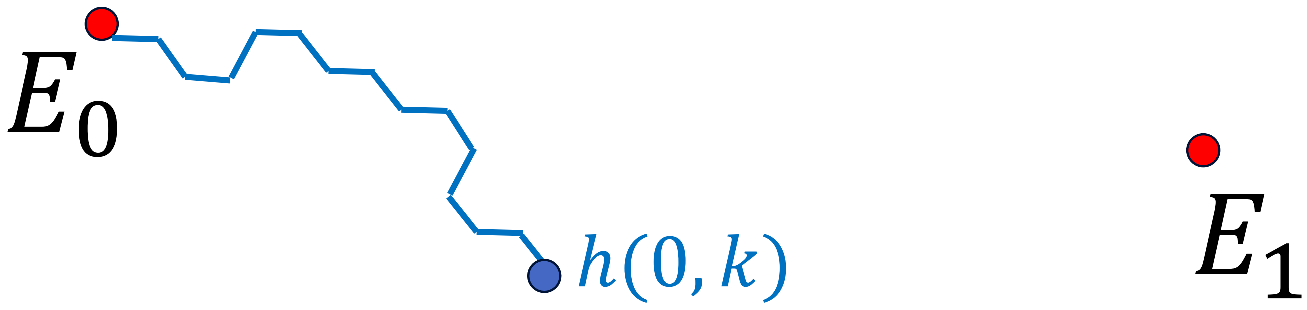
Date: April 3, 2018; updated on July 18, 2018.

This work

Let P_0, Q_0 be a basis for $E_0[2^e]$, and P_1, Q_1 be a basis for $E_1[2^e]$

Define $S = \{0,1\} \times \{0,1,\dots,2^{e/2}-1\}$

$(b,k) \in S$ fixes curve E_b, and k fixes subgroup $P_b + [k]Q_b$



Define h: $S \to \mathbb{F}_{p^2}$, $(b,z) \to j(E_b / \langle [2^{e/2}](P_b + [k]Q_b) \rangle)$

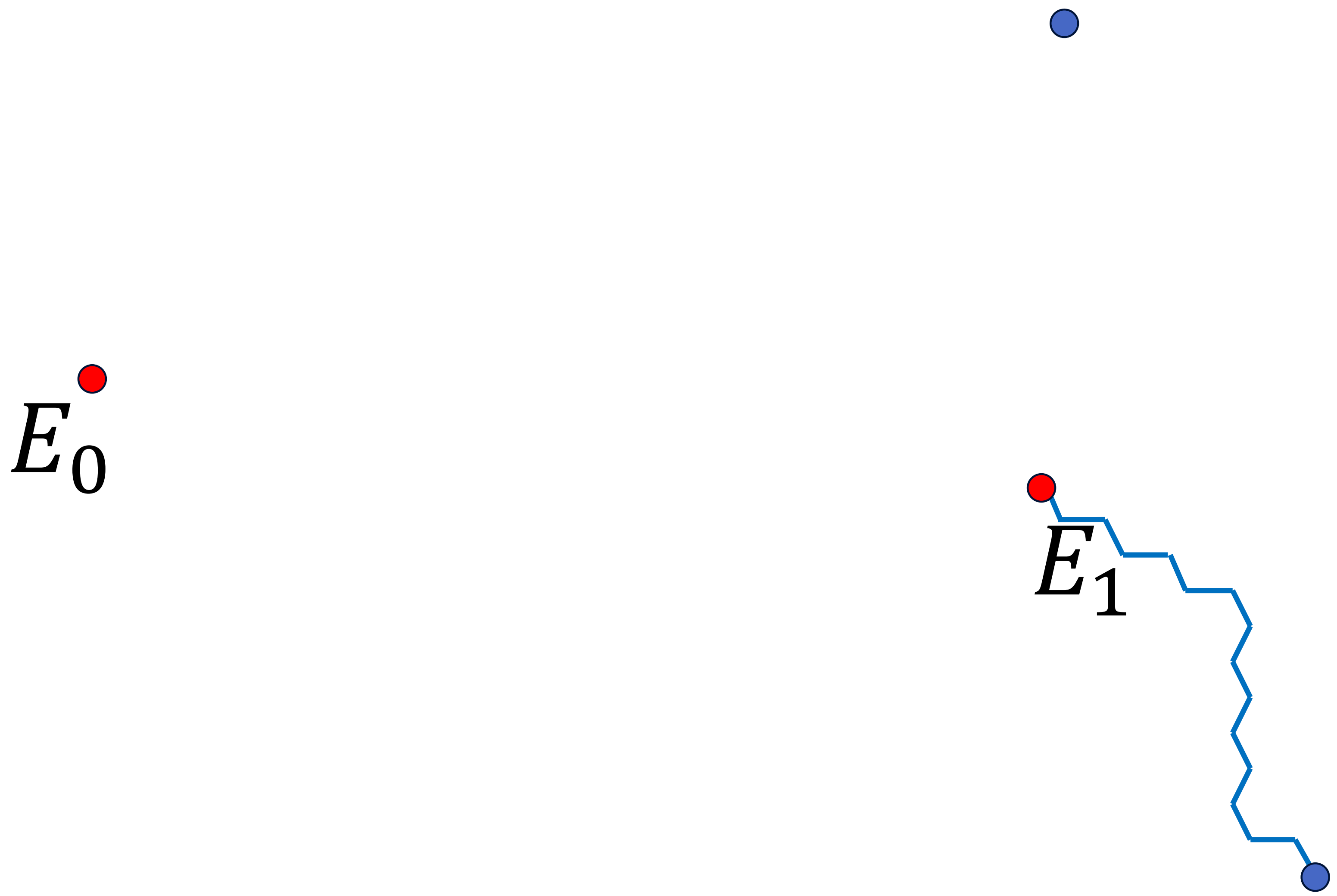
Define g_n: $\mathbb{F}_{p^2} \to S$, Merkle-Damgard based on AES with $IV = n$

Define $f_n : S \to S$, $(b,k) \mapsto (g_n \circ h)(b,k)$, simplifying notation...

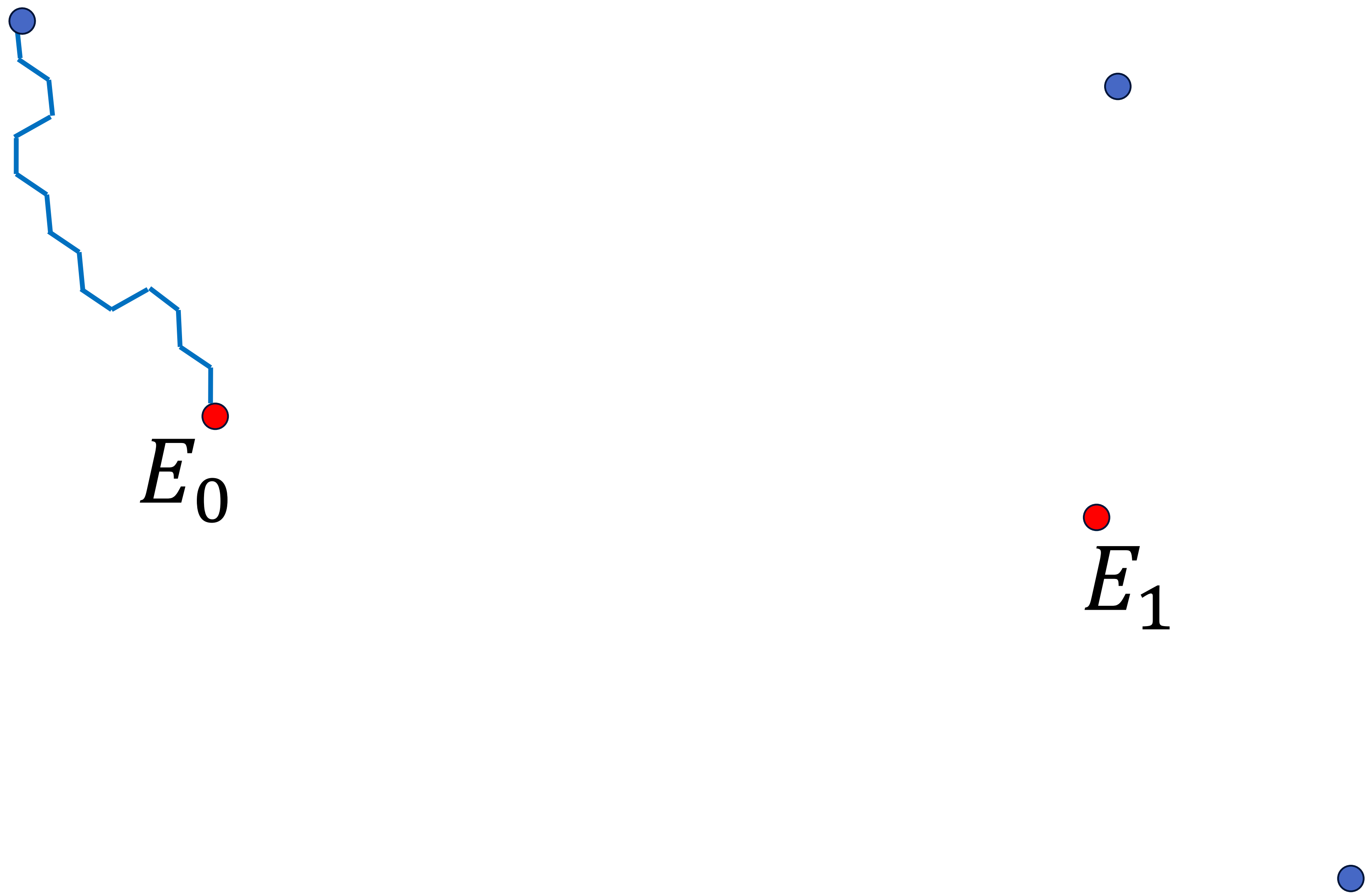
$$f_n\colon \quad S \to S$$

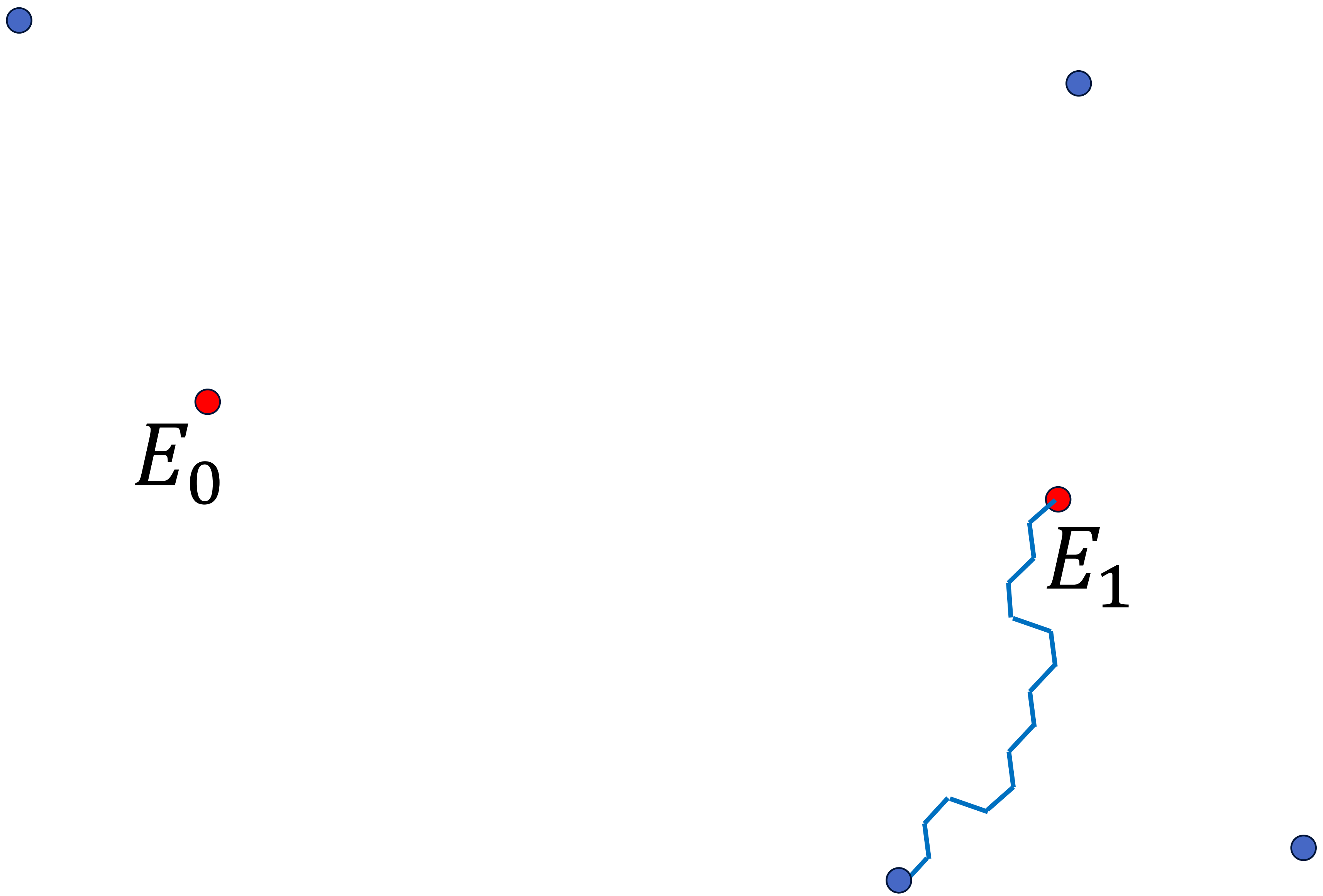
$$x_i \mapsto x_{i+1}$$

E_0
E_1

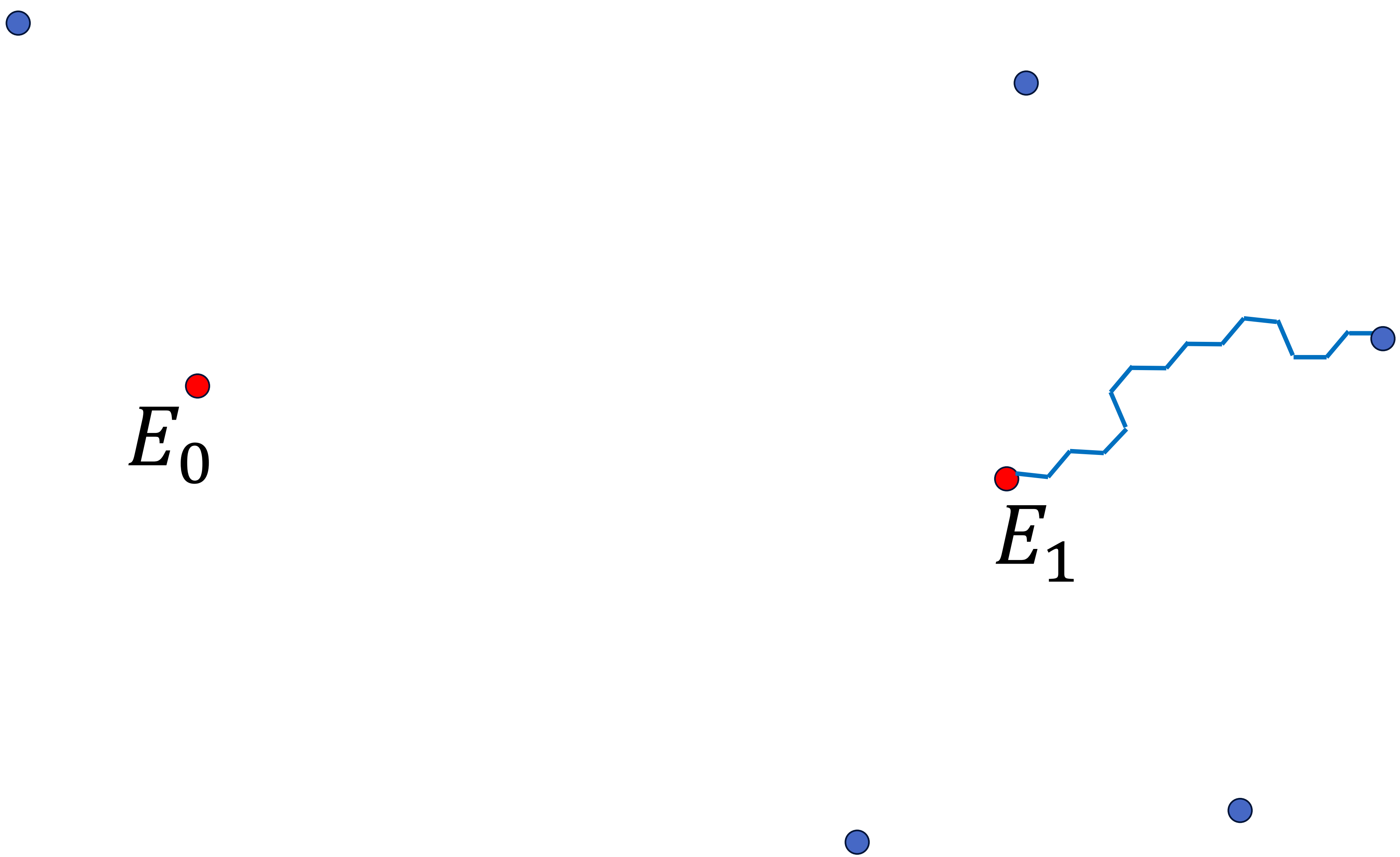


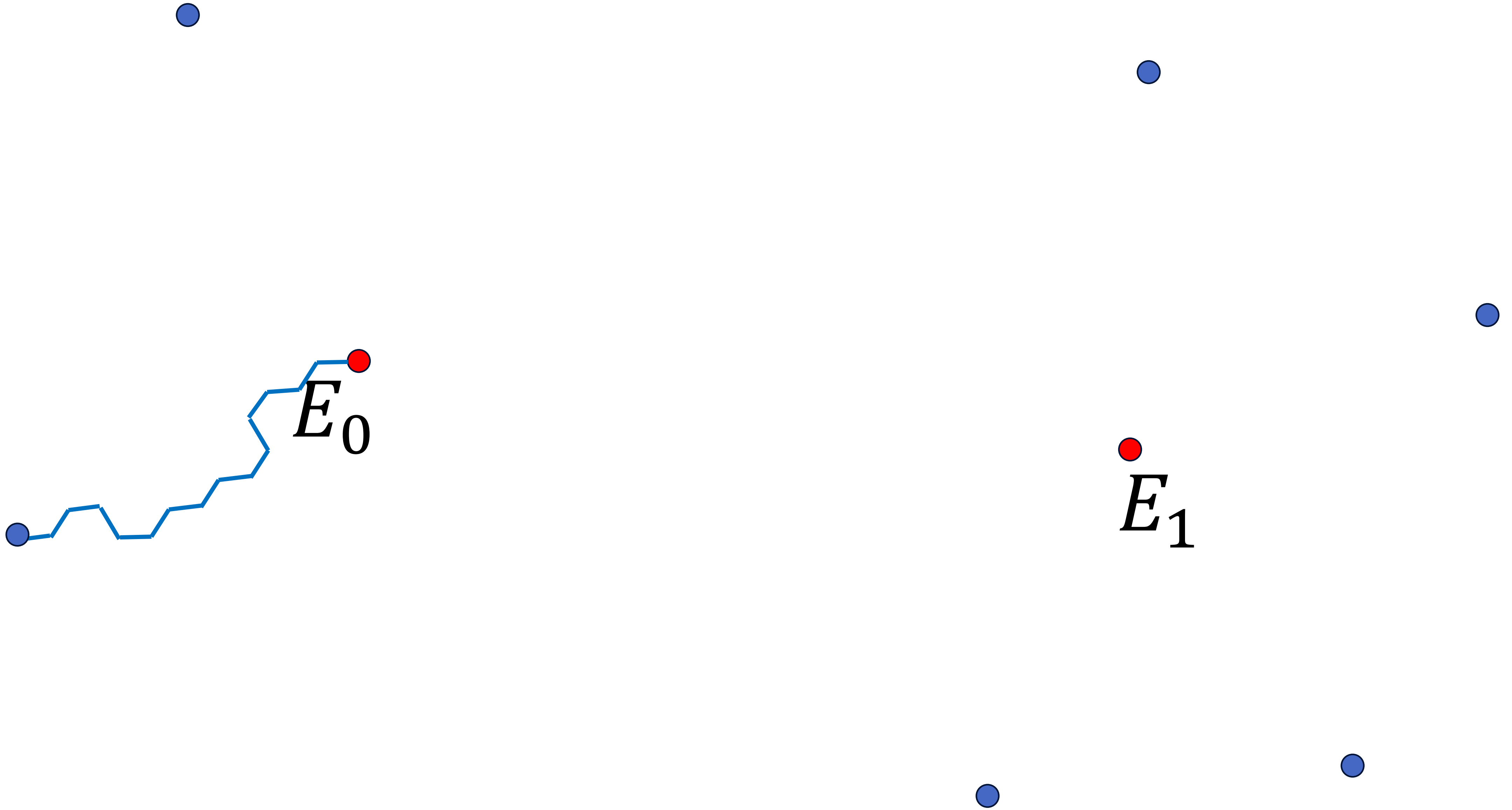
E_0
E_1


E_0
E_1

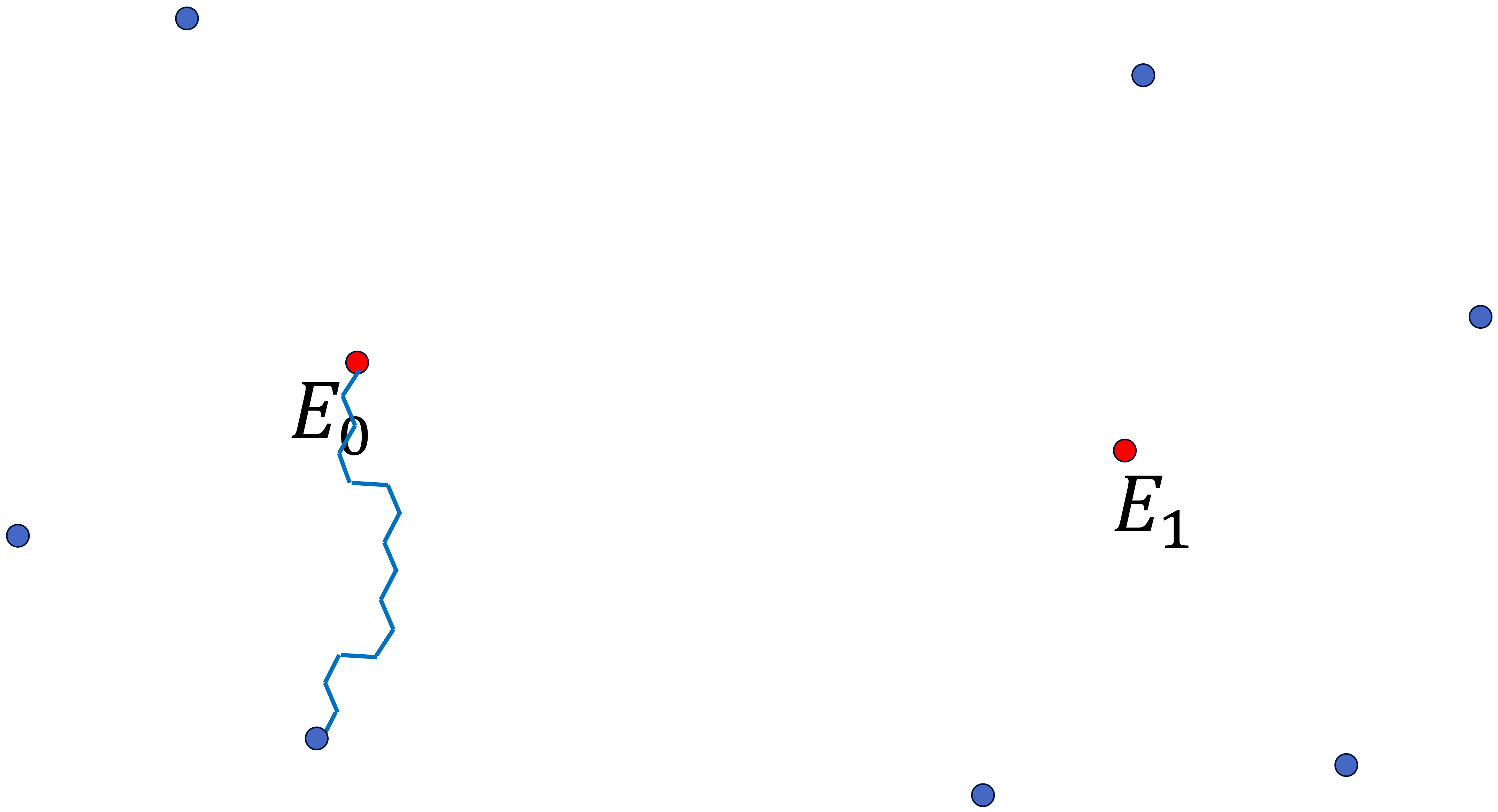


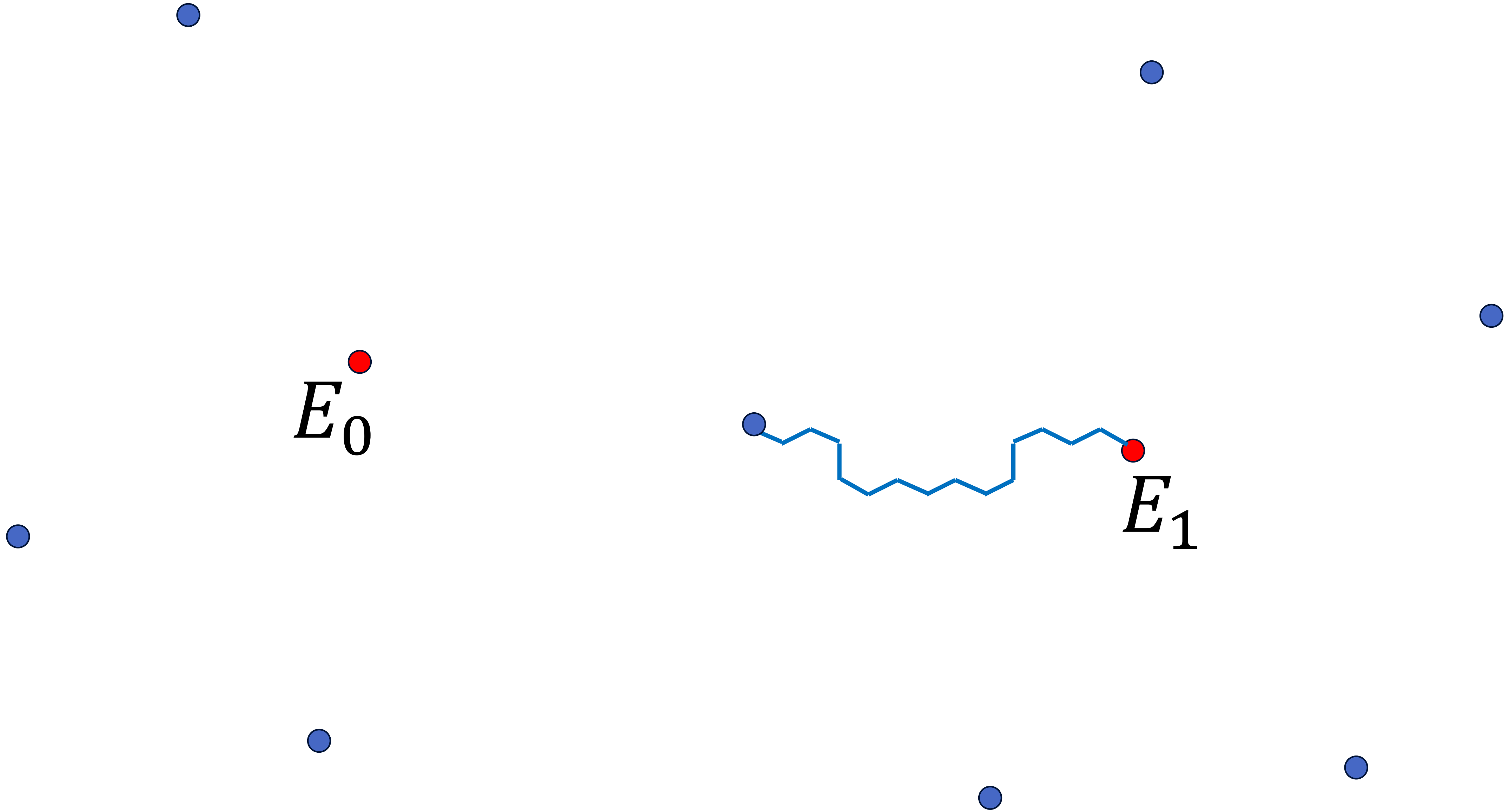
E_0
E_1

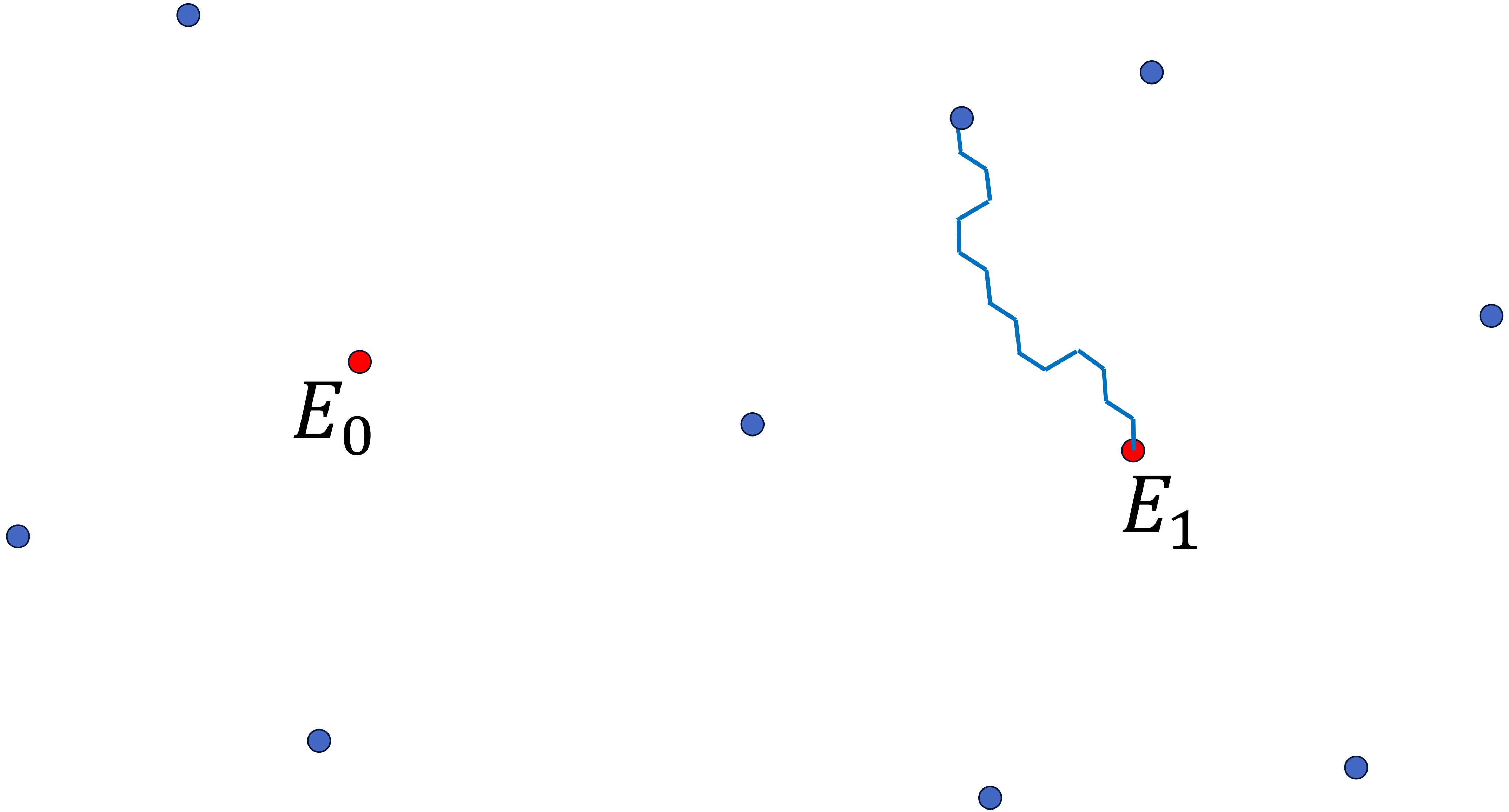

E_0
E_1

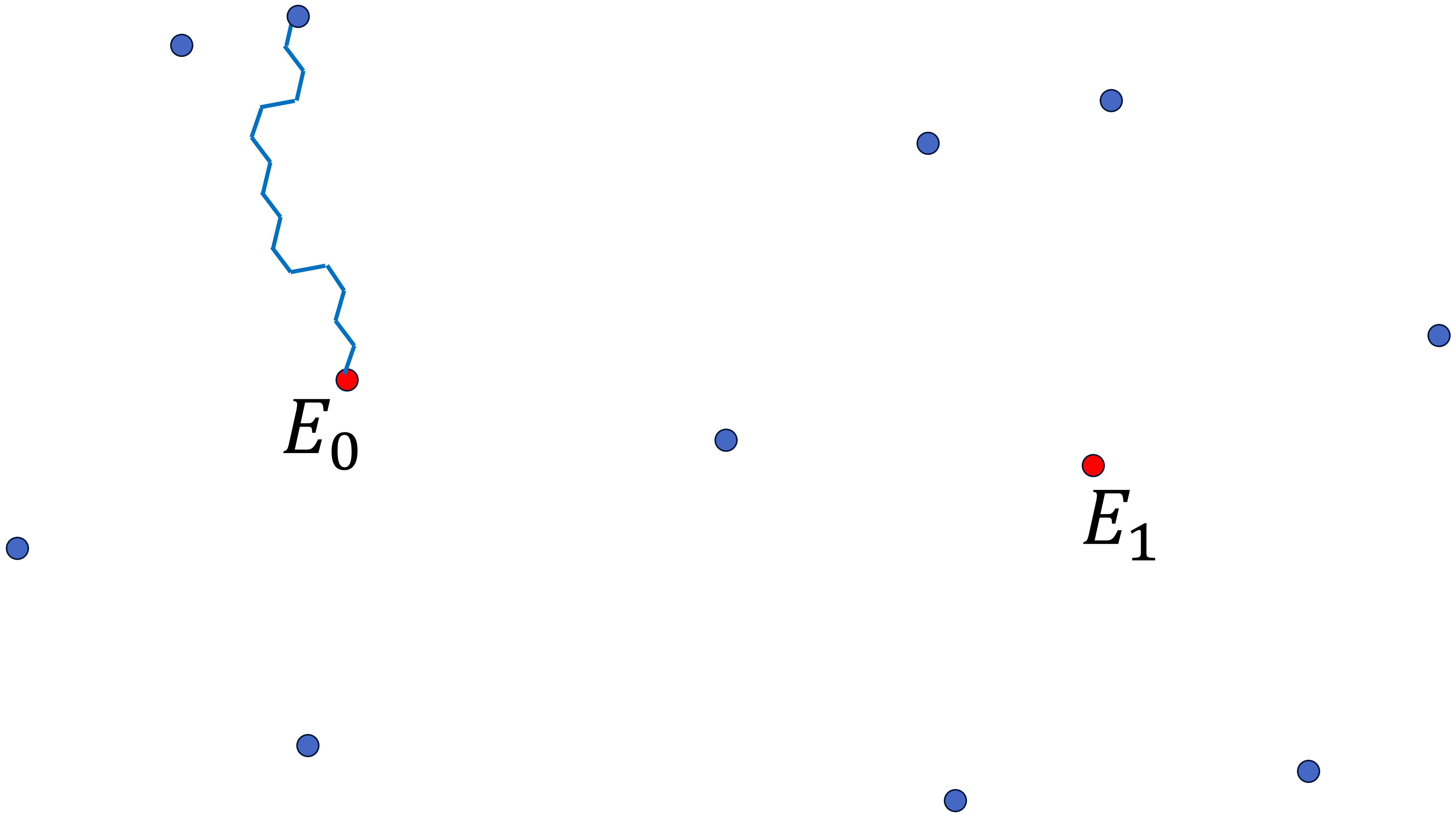


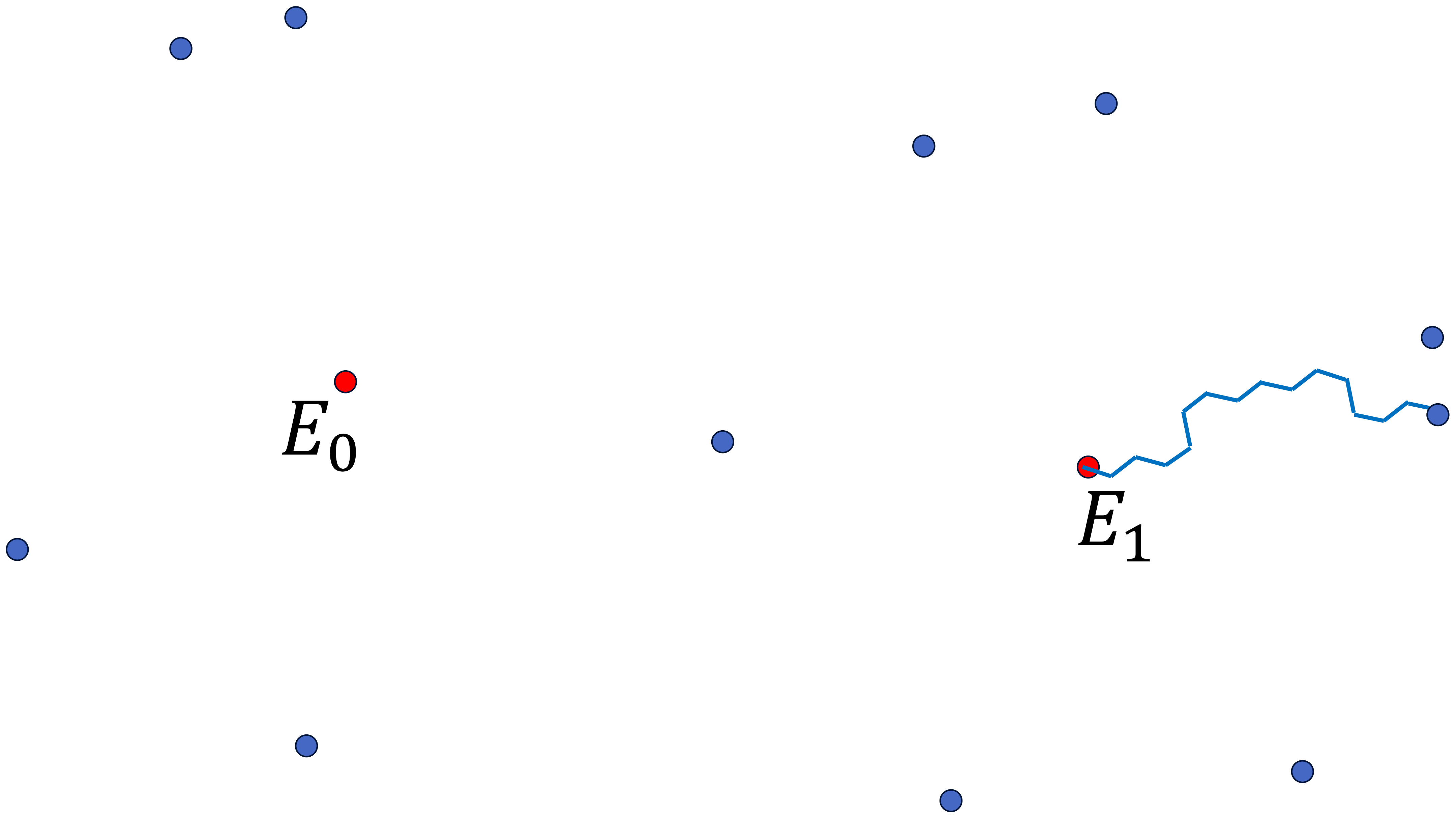
E_0
E_1


E_0
E_1

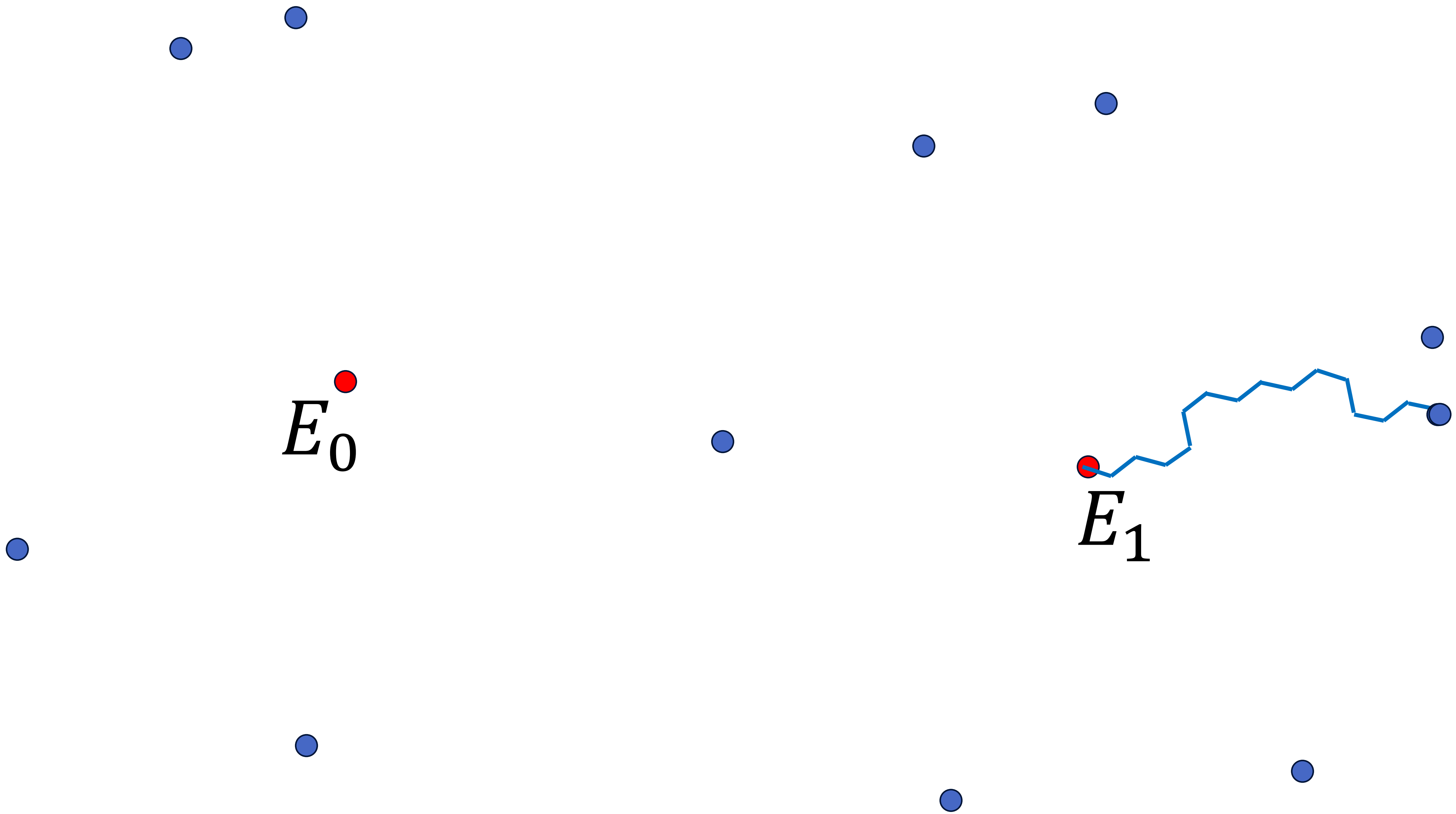

E_0
E_1


E_0
E_1

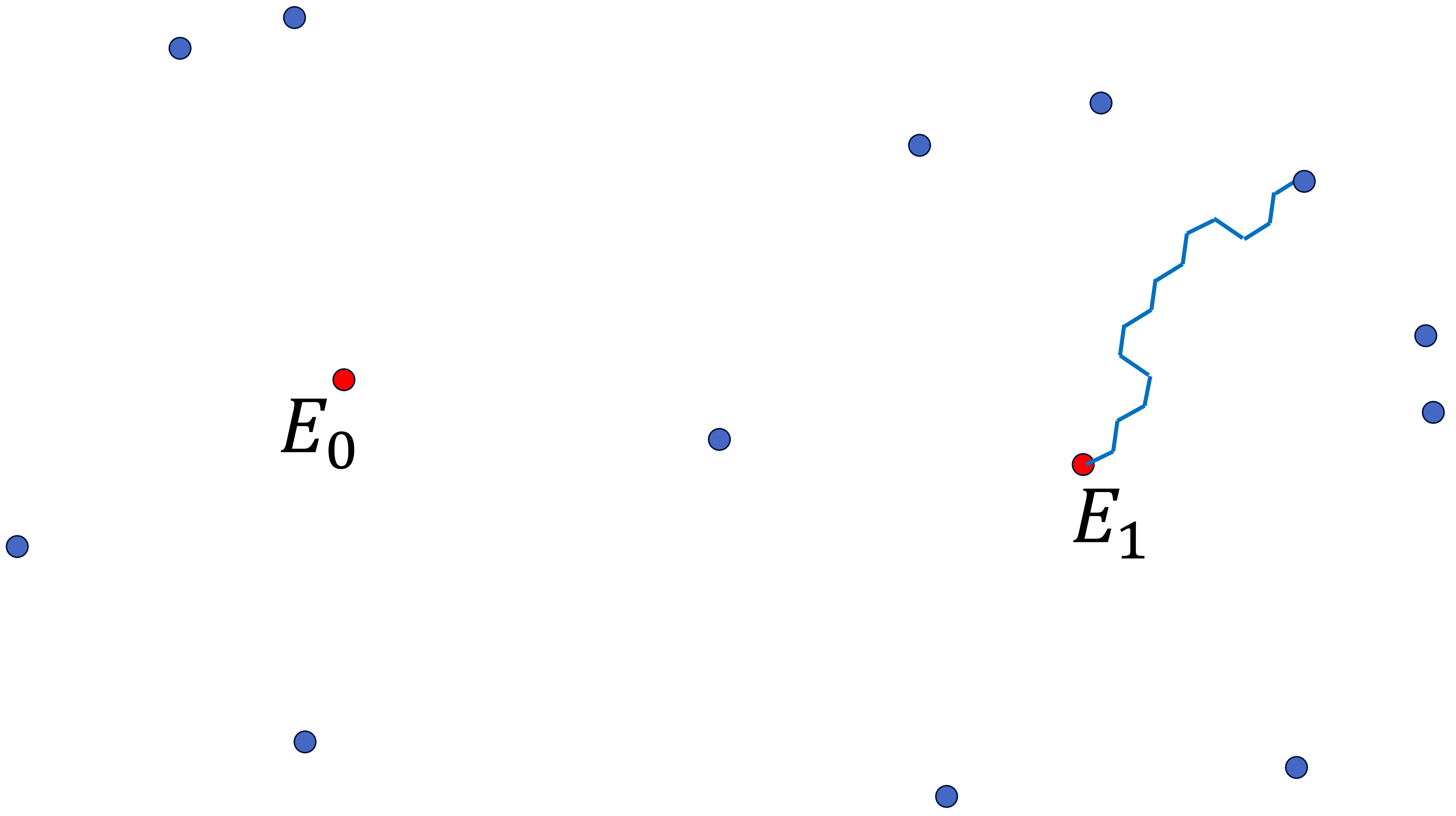

E_0
E_1



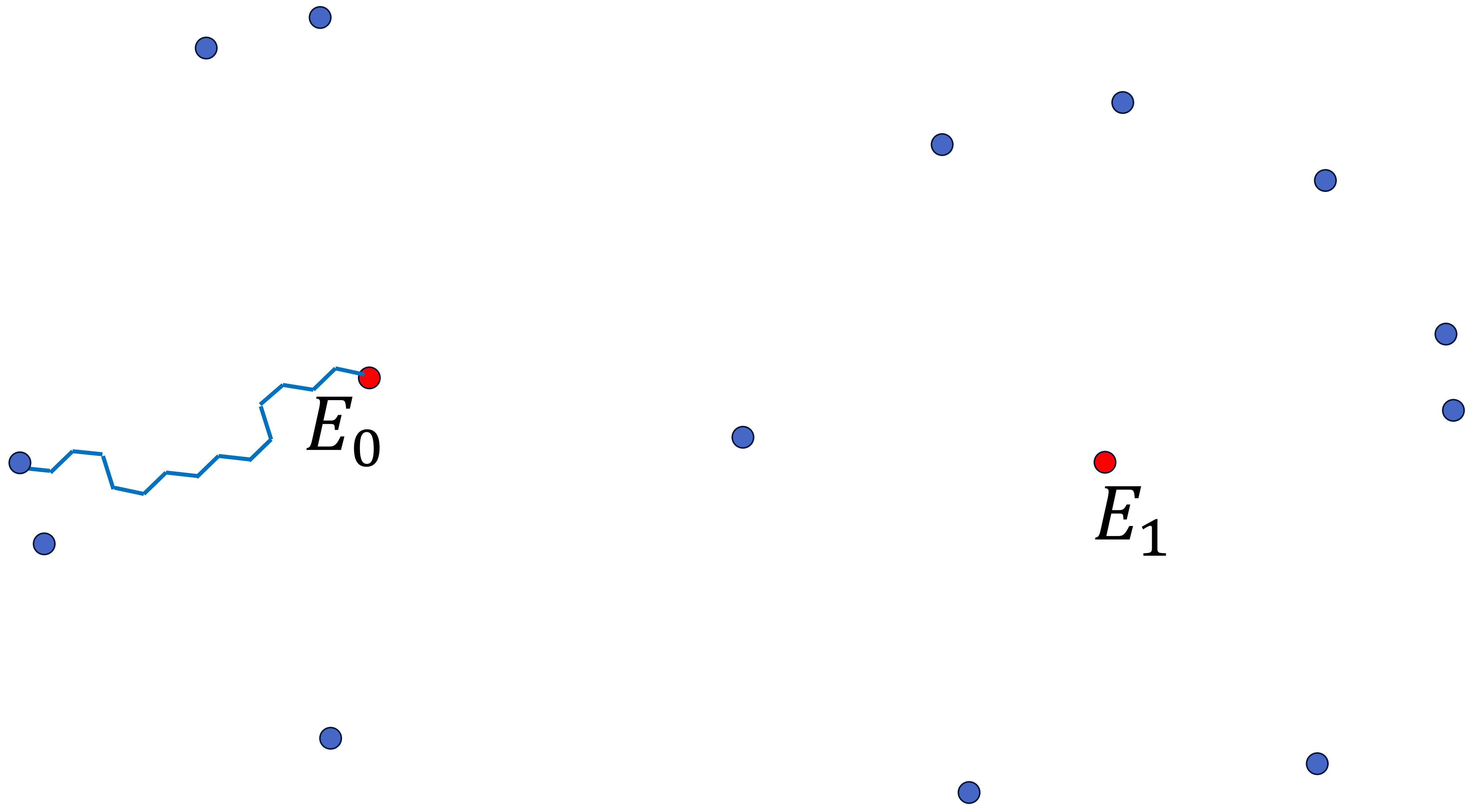
E_0
E_1

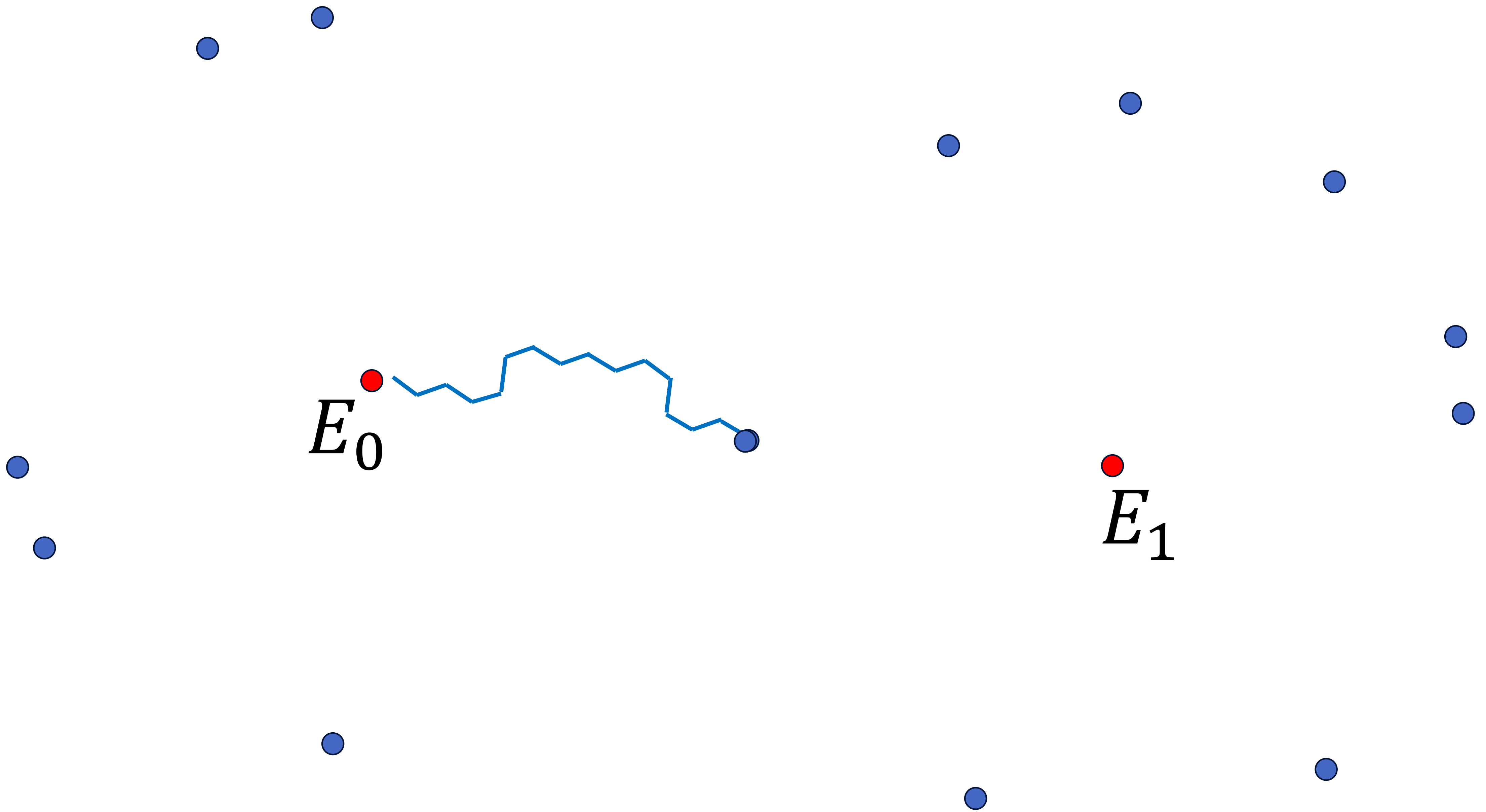


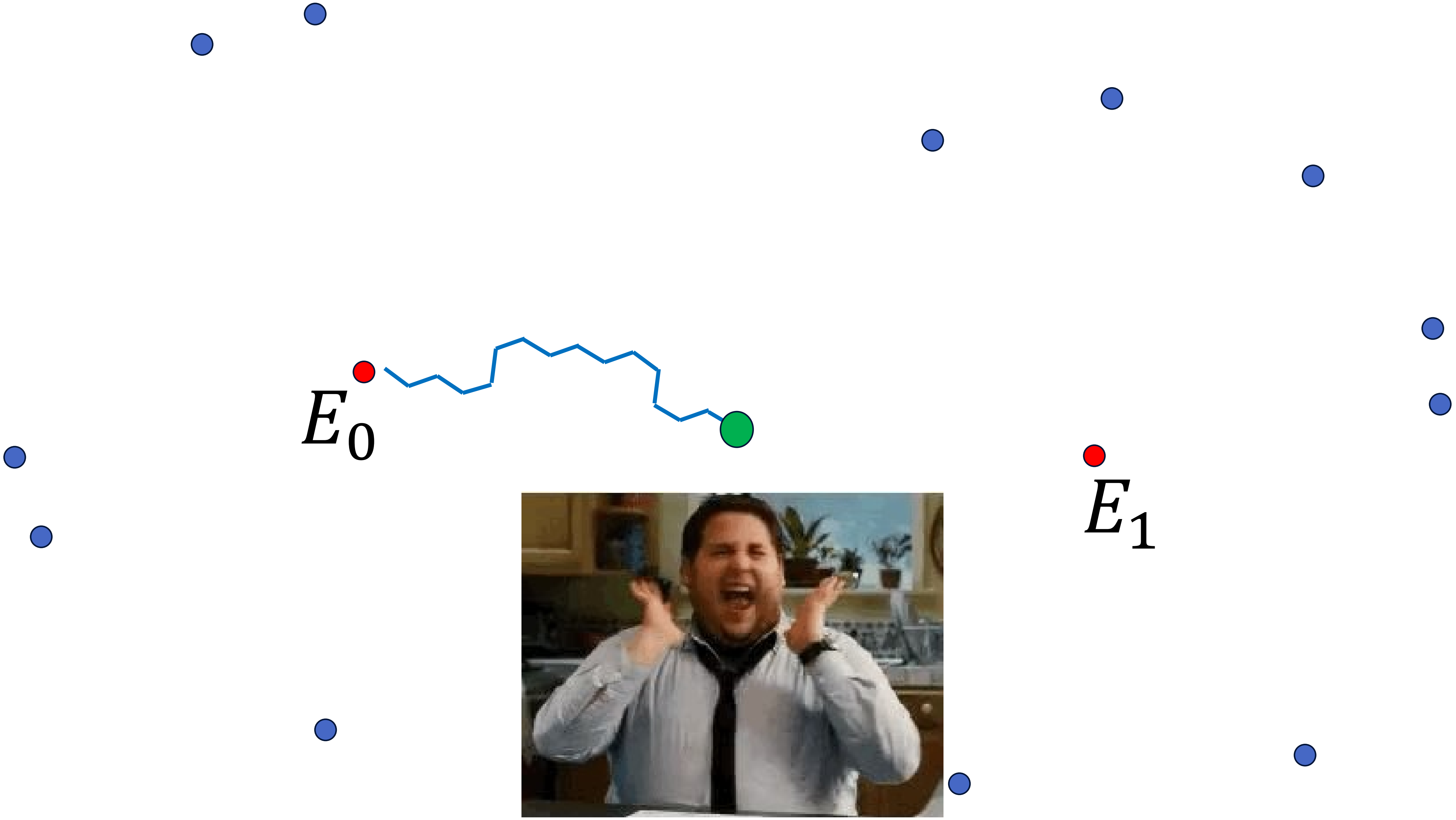
E_0
E_1



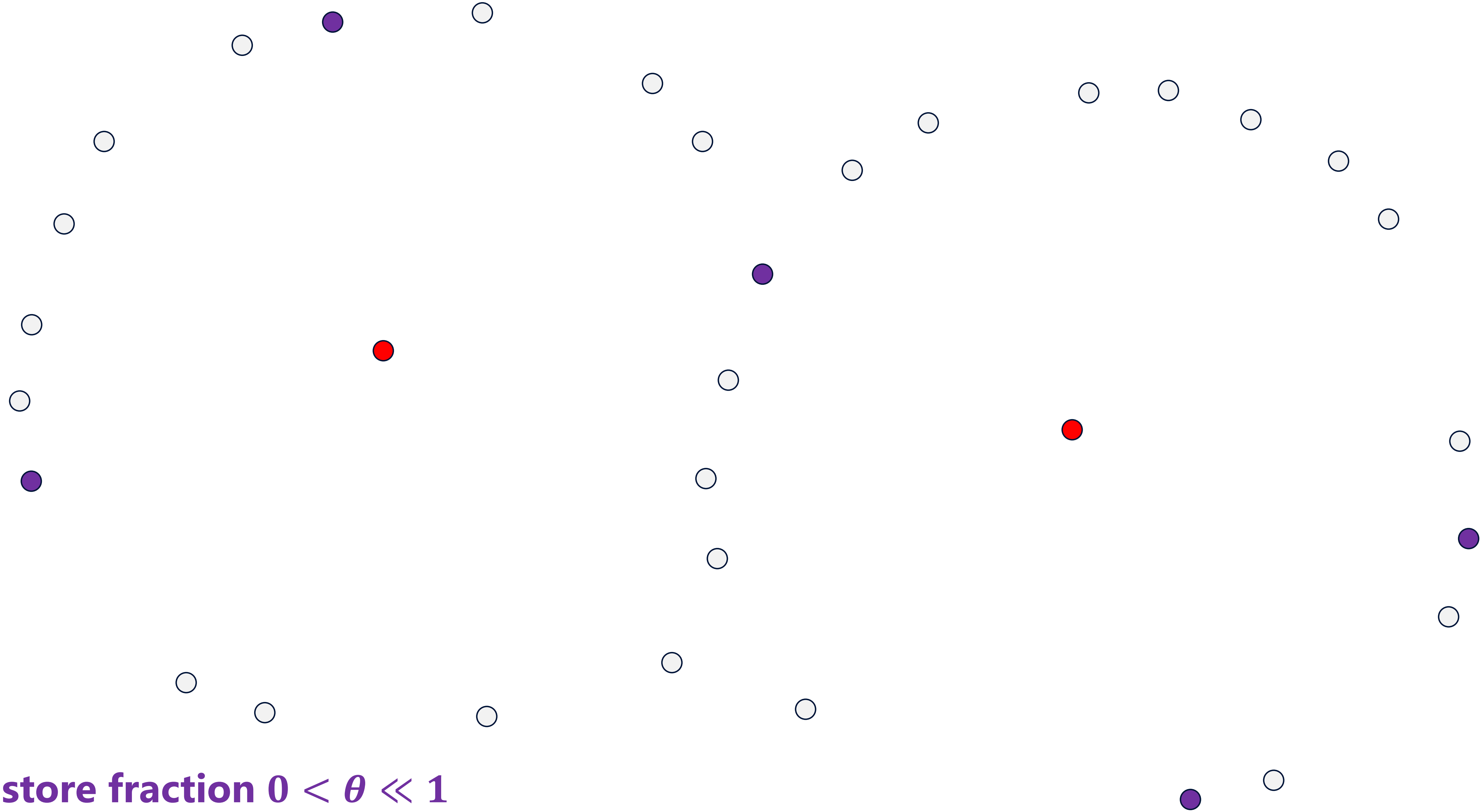
E_0
E_1


E_0
E_1


E_0
E_1


E_0
E_1

can’t possibly store all these: fix w as upper bound on $\#x_i$ storage



store fraction $0 < \theta \ll 1$

vOW

$f_n: \quad S \to S$

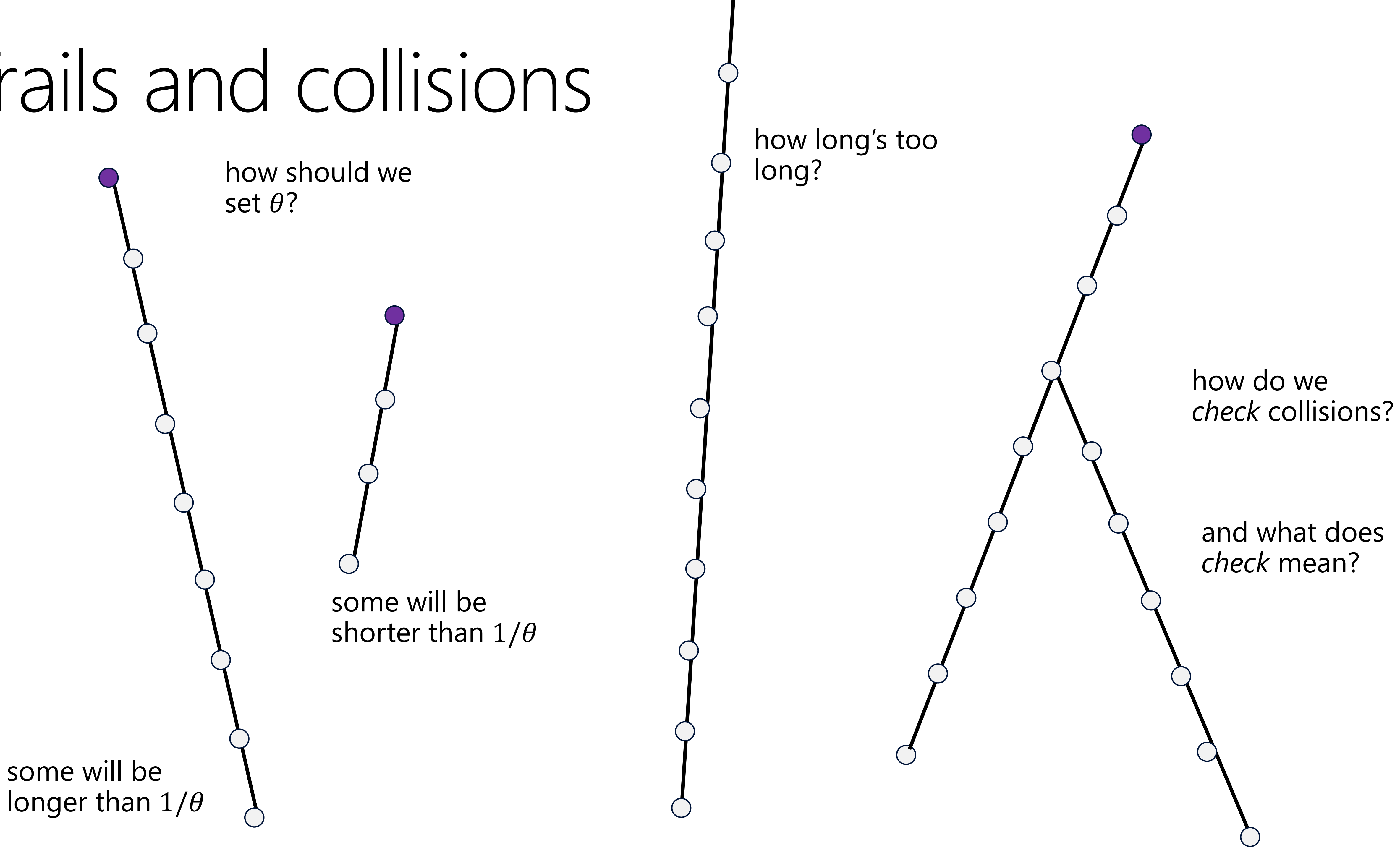
- f_n is a deterministic *random* function, different for each $IV = n$

- For a fixed n, each processor does the following:

  - pick a random starting point x_0
  - produce trail $x_i = f_n(x_{i-1})$, for $i = 1,2 \ldots$
  - stop when x_d is "distinguished" $(1/\theta)$.

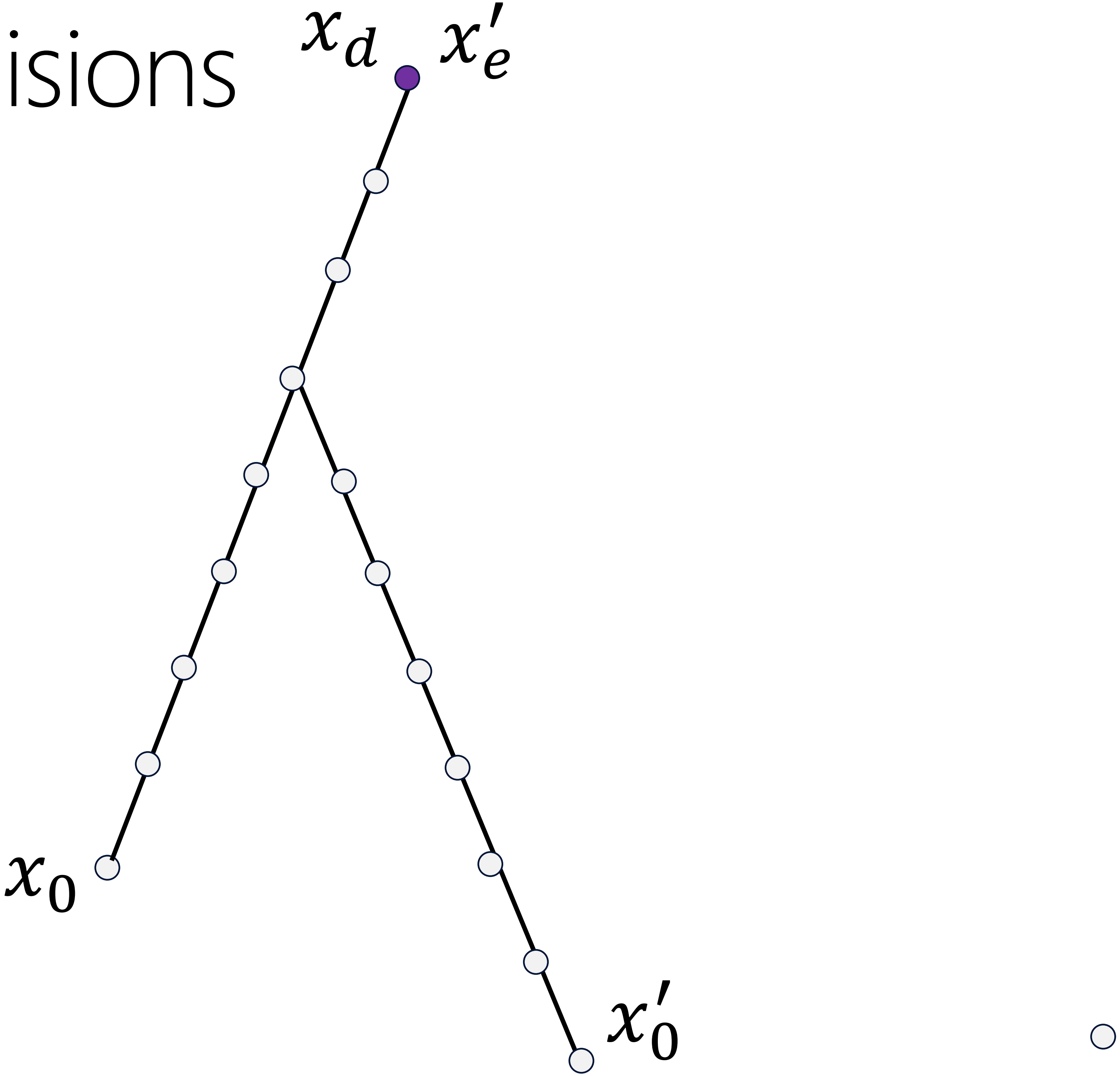
if (x_d has not been seen yet) then
store triple (x_0, x_d, d) and resample
else
if (collision not "golden") then
overwrite previous triple (x_0, x_d, d) and resample
else 

Trails and collisions



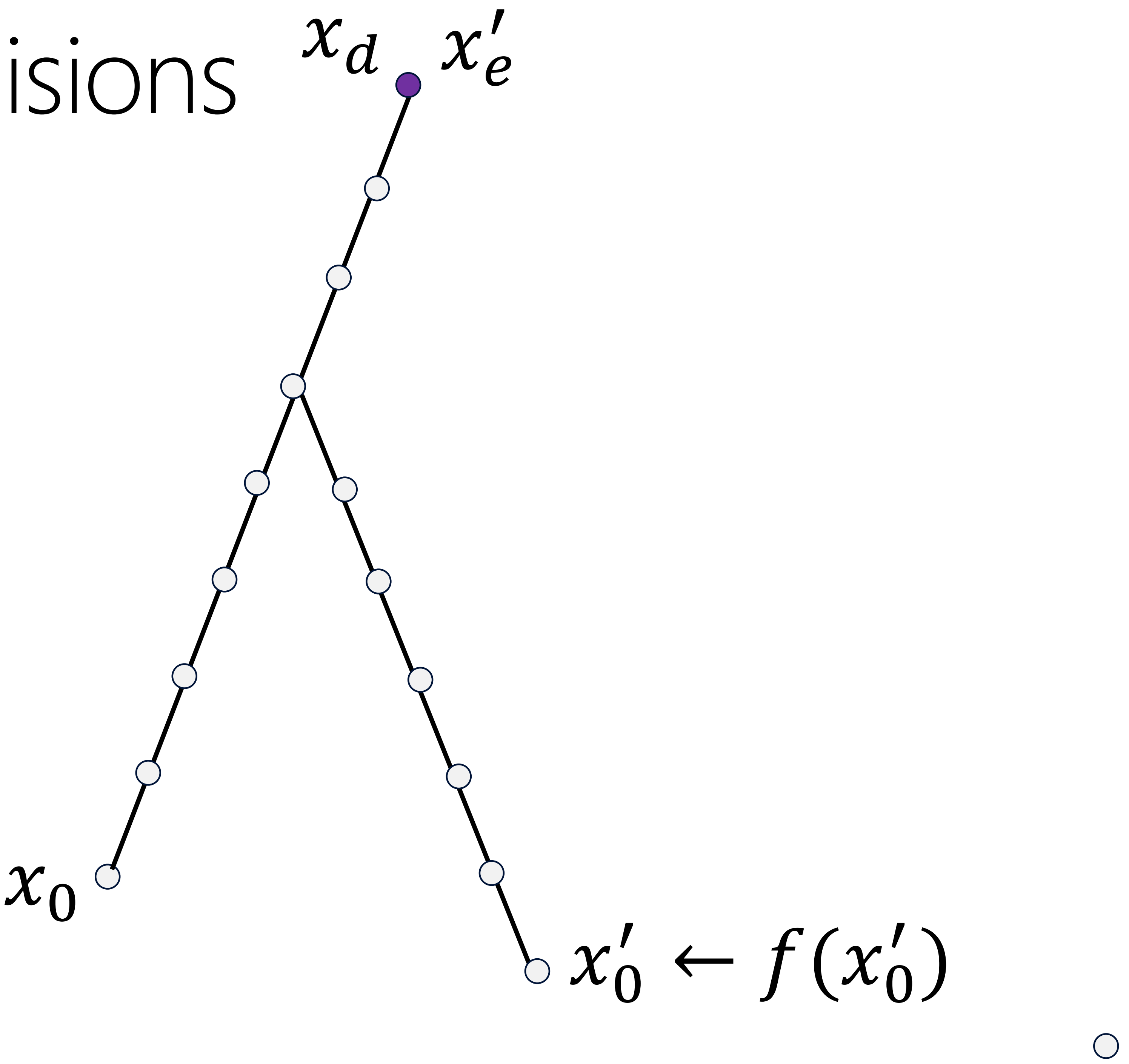
Checking collisions

memory
(x_0, x_d, d)
(x'_0, x'_e, e)



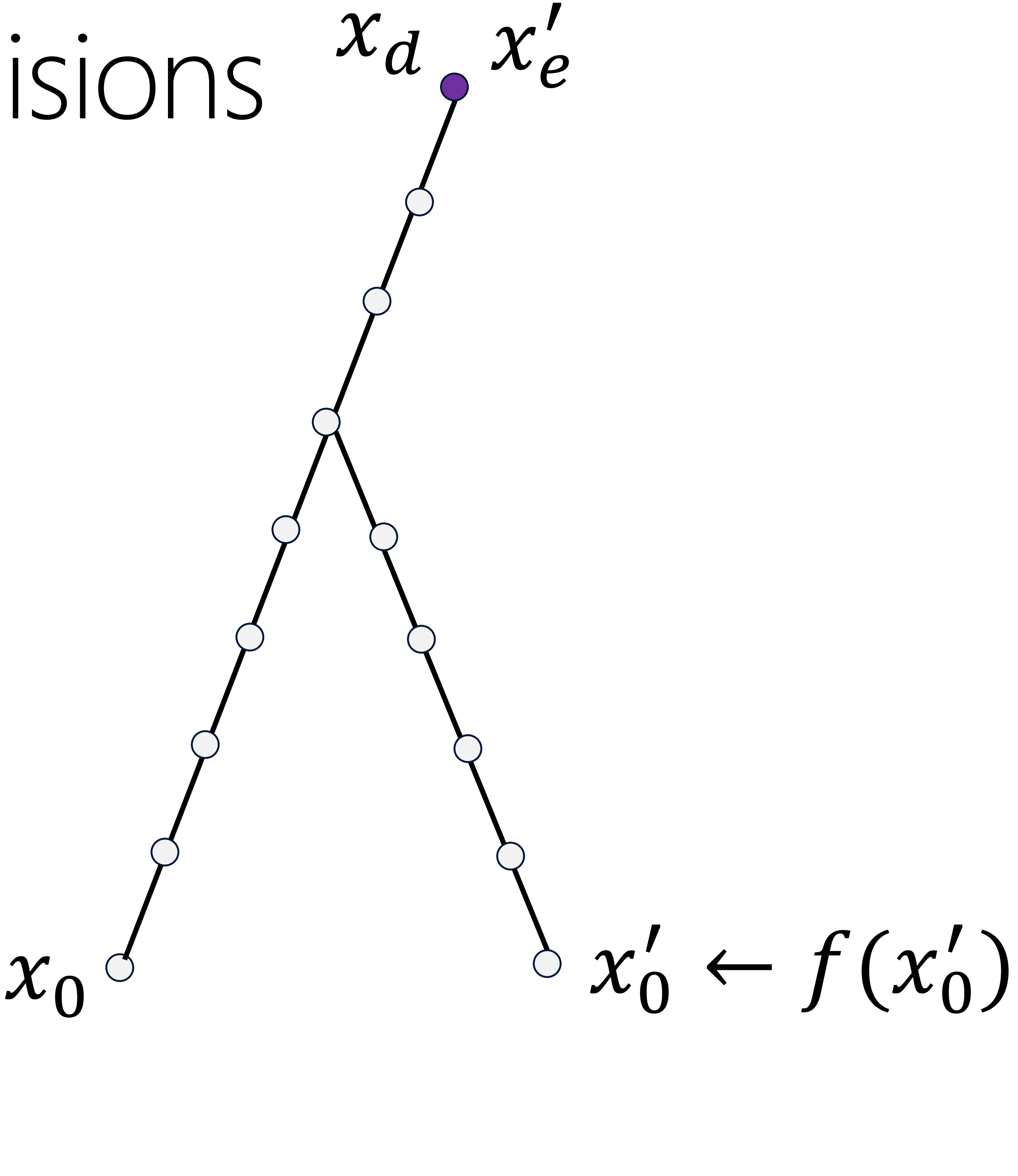
Checking collisions

memory
(x_0, x_d, d)
(x'_0, x'_e, e)



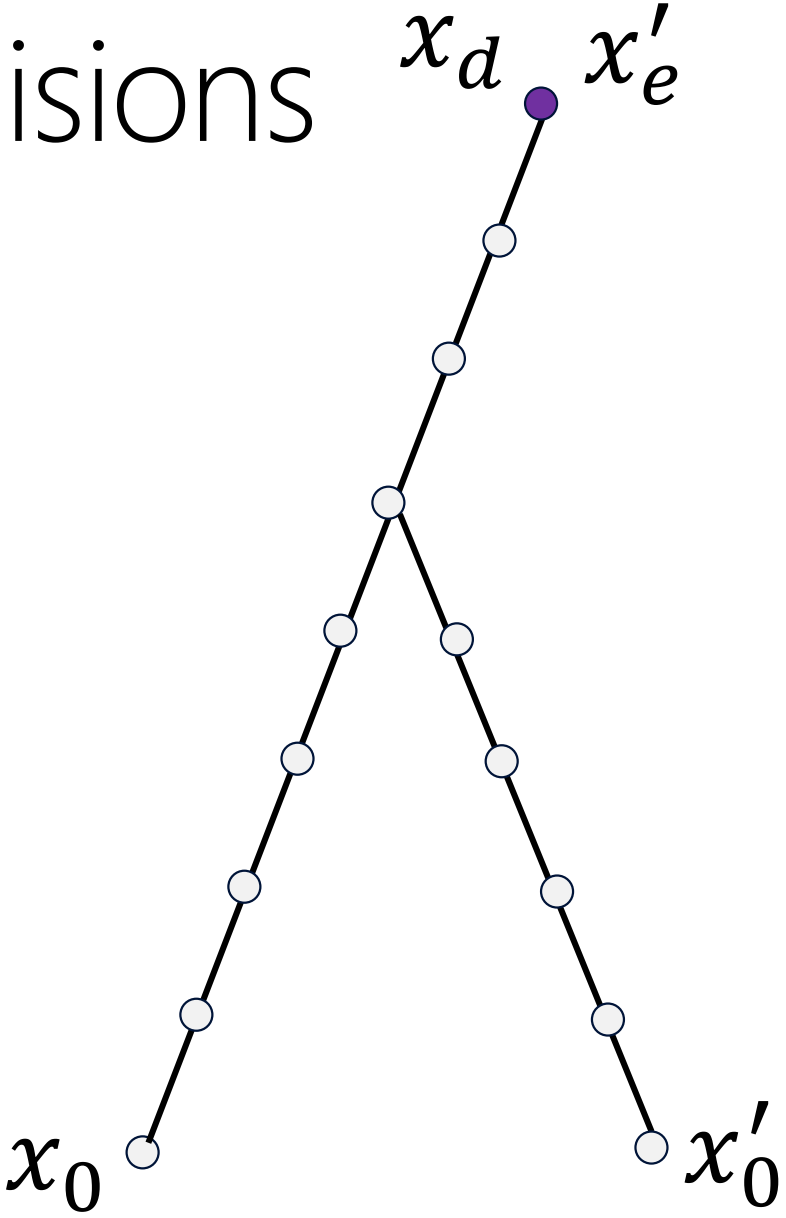
Checking collisions

memory
(x_0, x_d, d)
(x'_0, x'_e, e)



Checking collisions

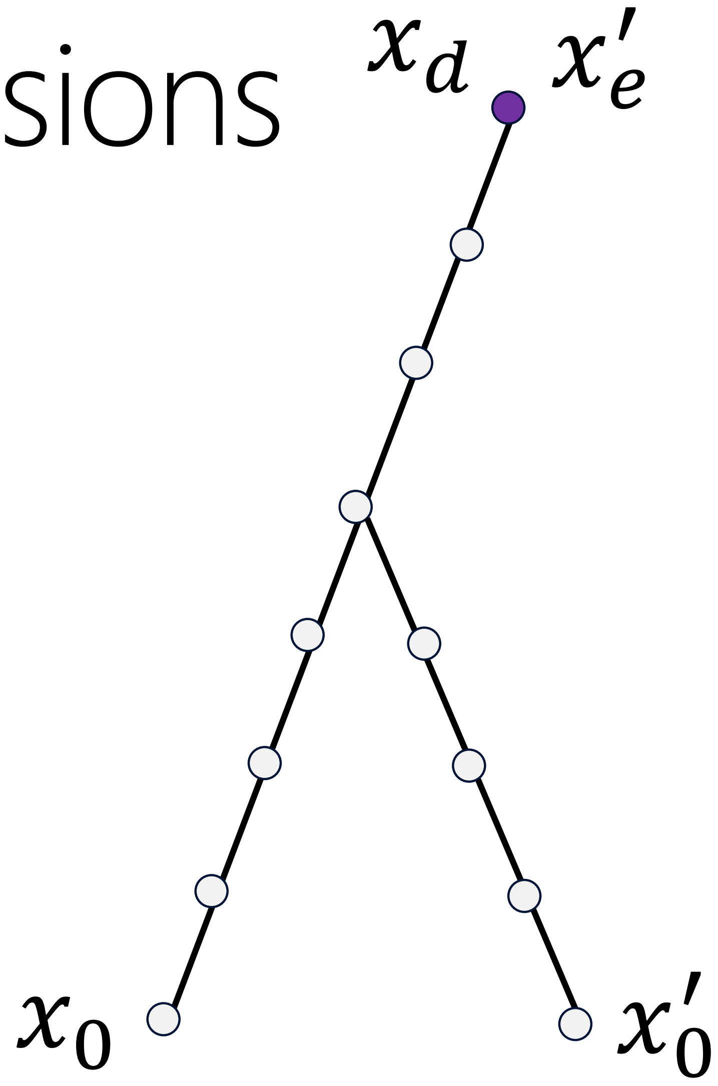
memory
(x_0, x_d, d)
(x'_0, x'_e, e)



$f_n(x_0) \neq f_n(x'_0)$

Checking collisions

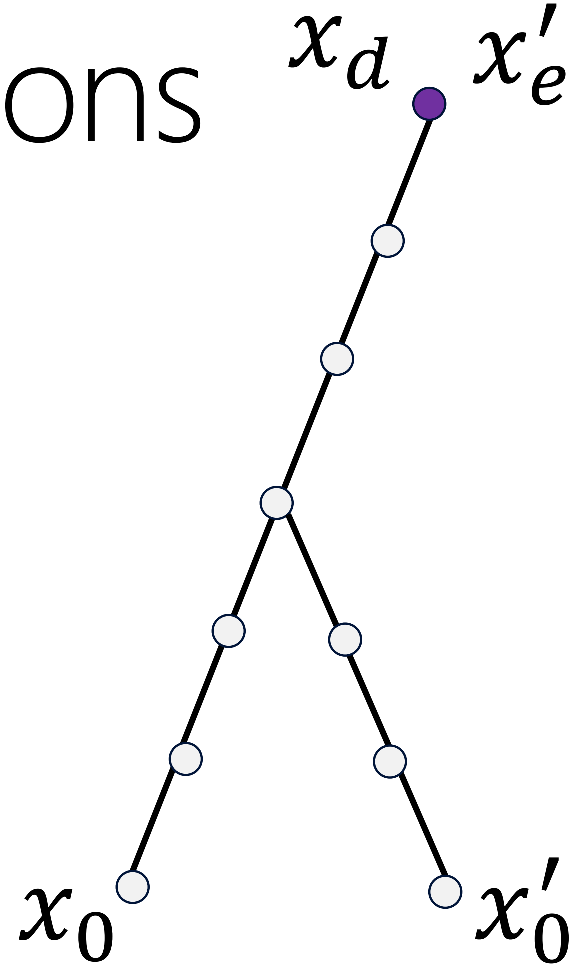
memory
(x_0, x_d, d)
(x'_0, x'_e, e)



$f_n(x_0) \neq f_n(x'_0)$

Checking collisions

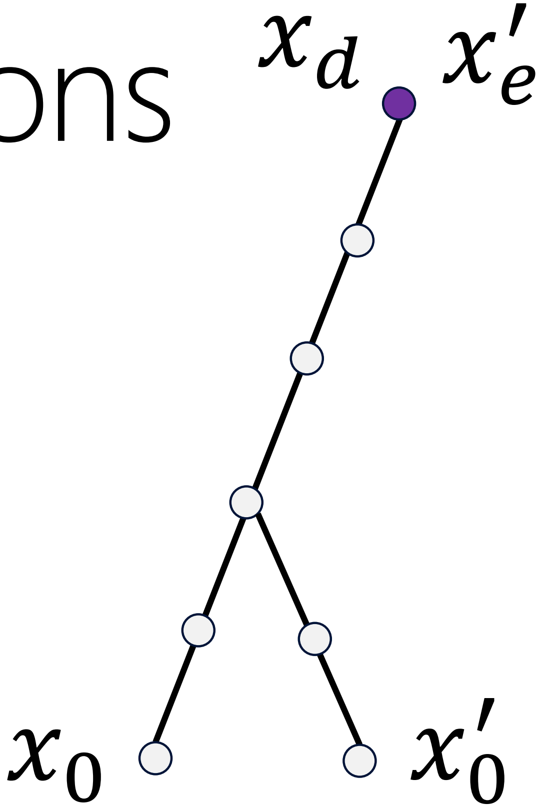
memory
(x_0, x_d, d)
(x'_0, x'_e, e)



$f_n(x_0) \neq f_n(x'_0)$

Checking collisions

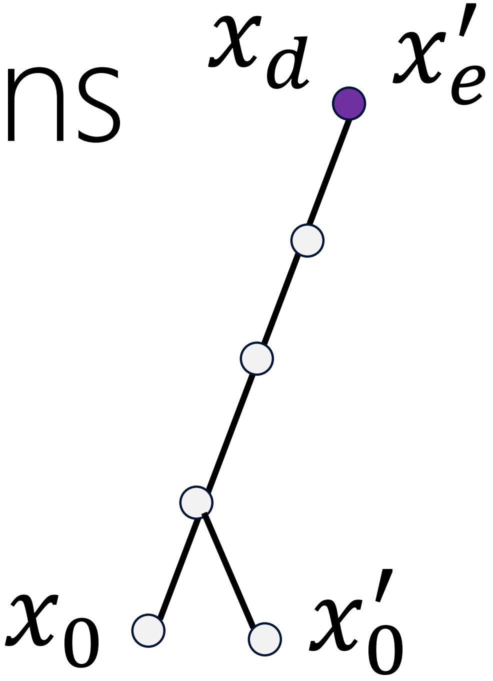
memory
(x_0, x_d, d)
(x'_0, x'_e, e)



$f_n(x_0) \neq f_n(x'_0)$

Checking collisions

memory
(x_0, x_d, d)
(x'_0, x'_e, e)

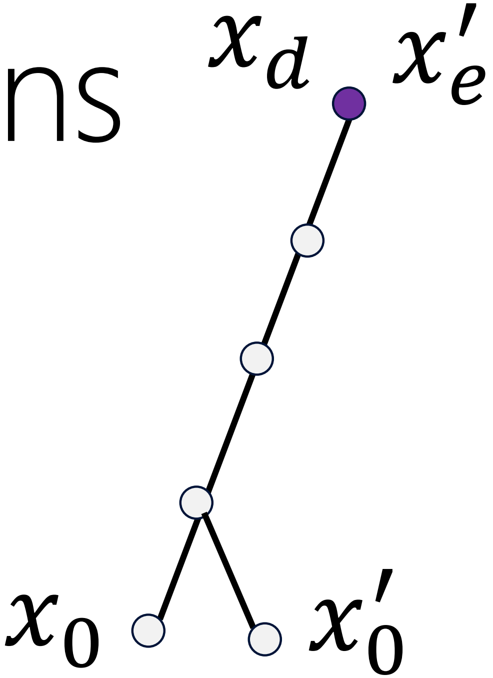


$f_n(x_0) = f_n(x'_0)$
$x_0 \neq x'_0$

DONE?

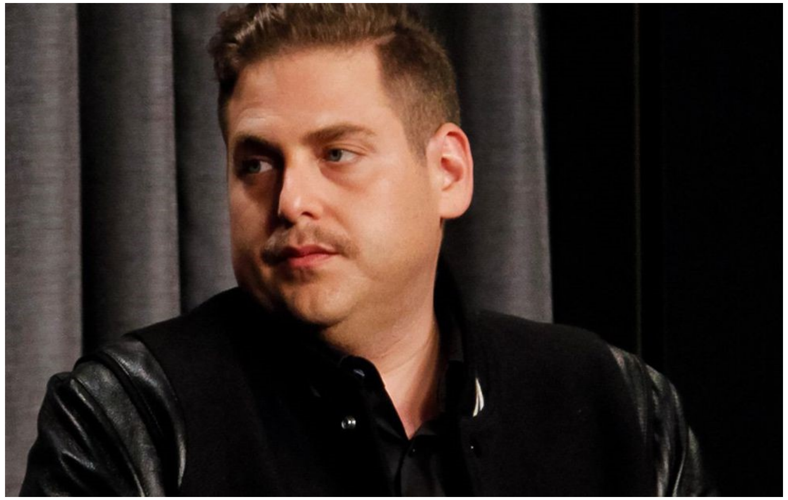
Checking collisions

memory
(x_0, x_d, d)
(x'_0, x'_e, e)



$f_n(x_0) = f_n(x'_0)$

Nope! False alarm



Random collisions vs. the golden collision

- A random function $f_n : S \to S$ has many collisions, e.g., think of the random function as a hash function (it kinda is anyway)

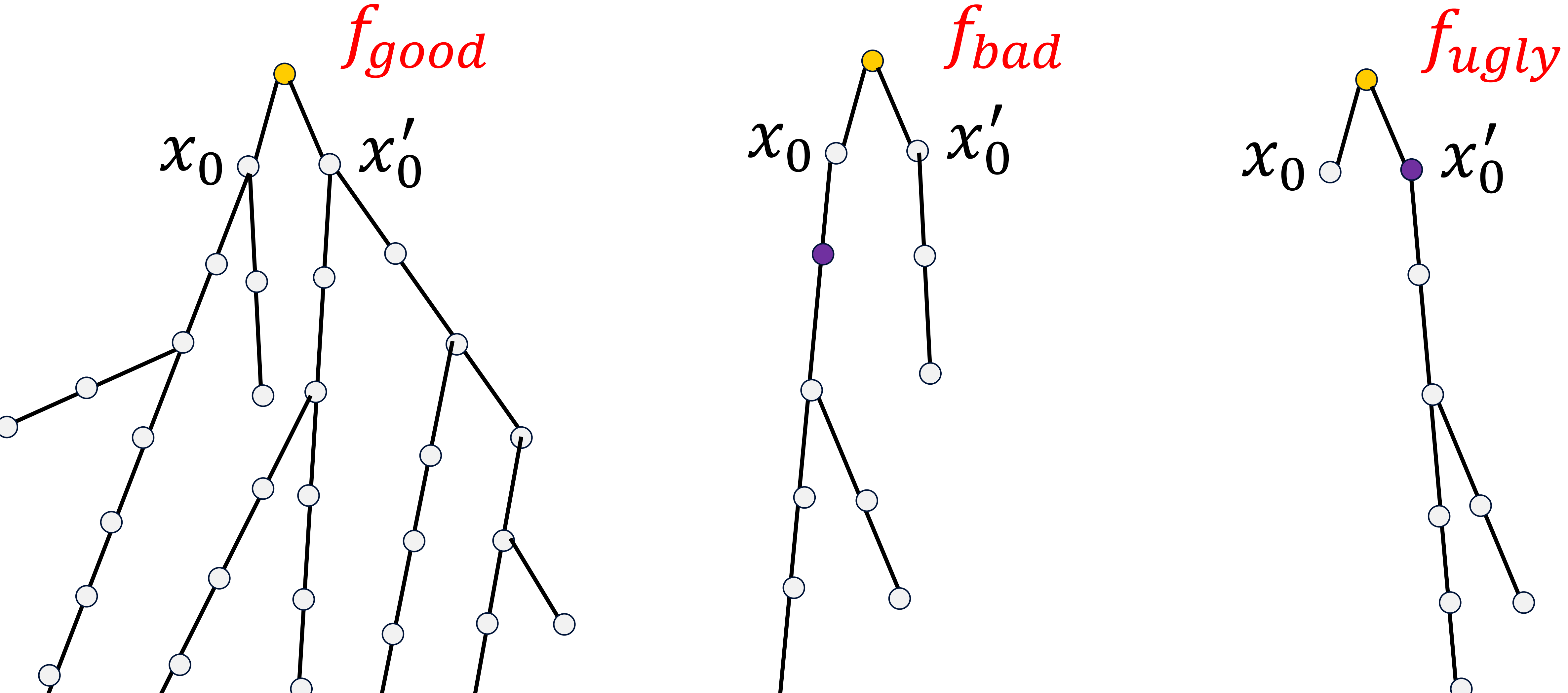
- We will encounter many of these before we hit the one we want, i.e., the "golden collision"

- Much of the algorithm is spent walking, much is spent checking useless annoying collisions

- Ideally there'll be many paths that take us to the golden collision…

Random f_n: the good, the bad and the ugly…

- Even more annoying is that we have to restart the whole algorithm, time and time again…



Analysis (vOW, Adj et al, us…)

SIDH: $|S| \approx p^{1/4}$
Adj et al: $w \approx 2^{80}$

- How many distinguished elements? $\theta \approx 2.25\sqrt{w/|S|}$
- How long before switching functions? $\approx \mathbf{10}w$ distinguish points
- How long before giving up on a trail? $\approx \mathbf{20}/\theta$ function iterations
- With these params, what's the runtime? $\approx O\left(\frac{|S|^{\frac{3}{2}}}{\sqrt{w}}\right)$
- Compared to MitM? $\approx O\left(\frac{|S|^2}{w}\right)$

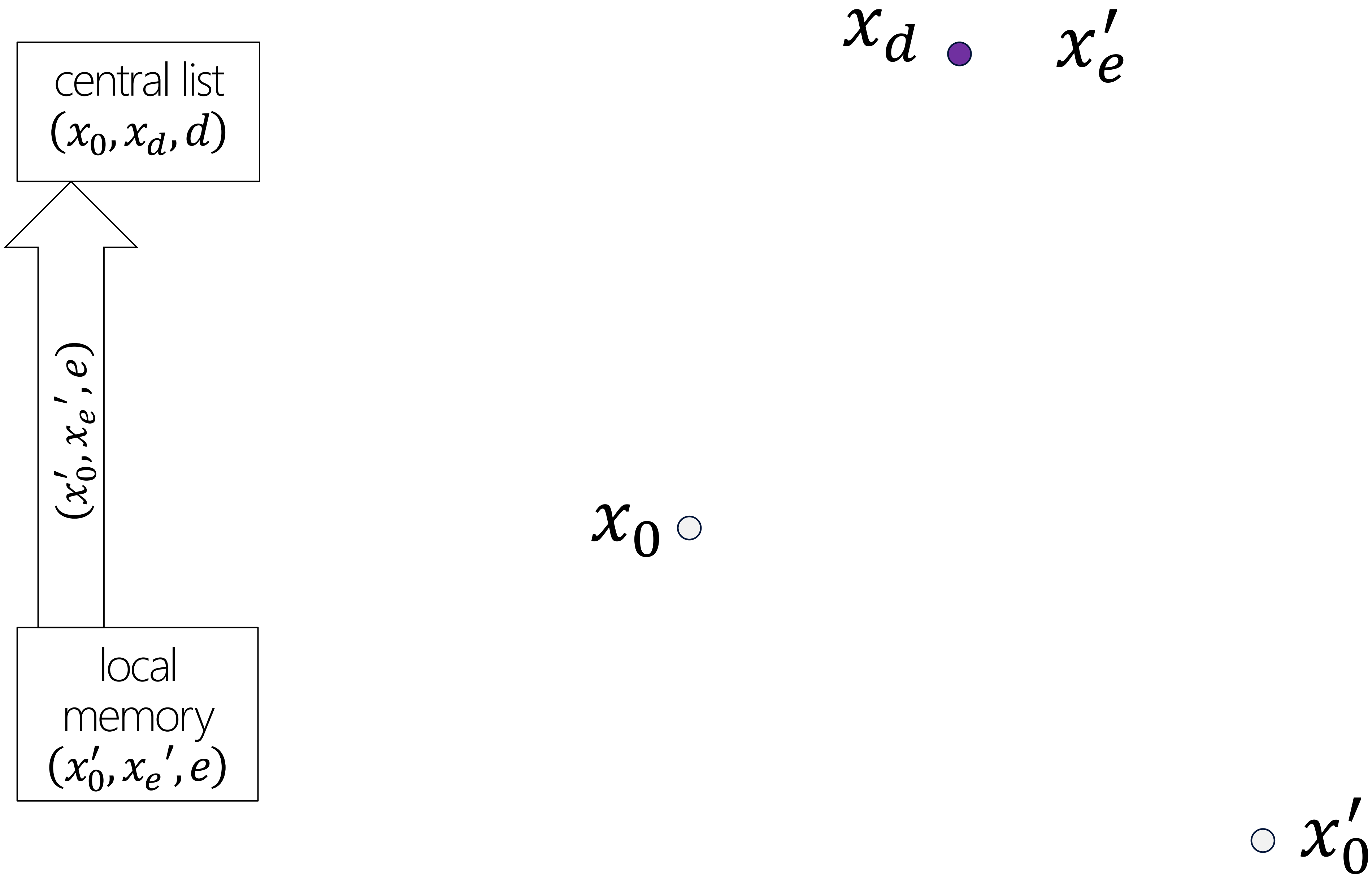
This work

- Fast(er) collision checking
- Real-world/distributed analysis
- SIKE-specific optimisations: conjugates, fixed-bits, …

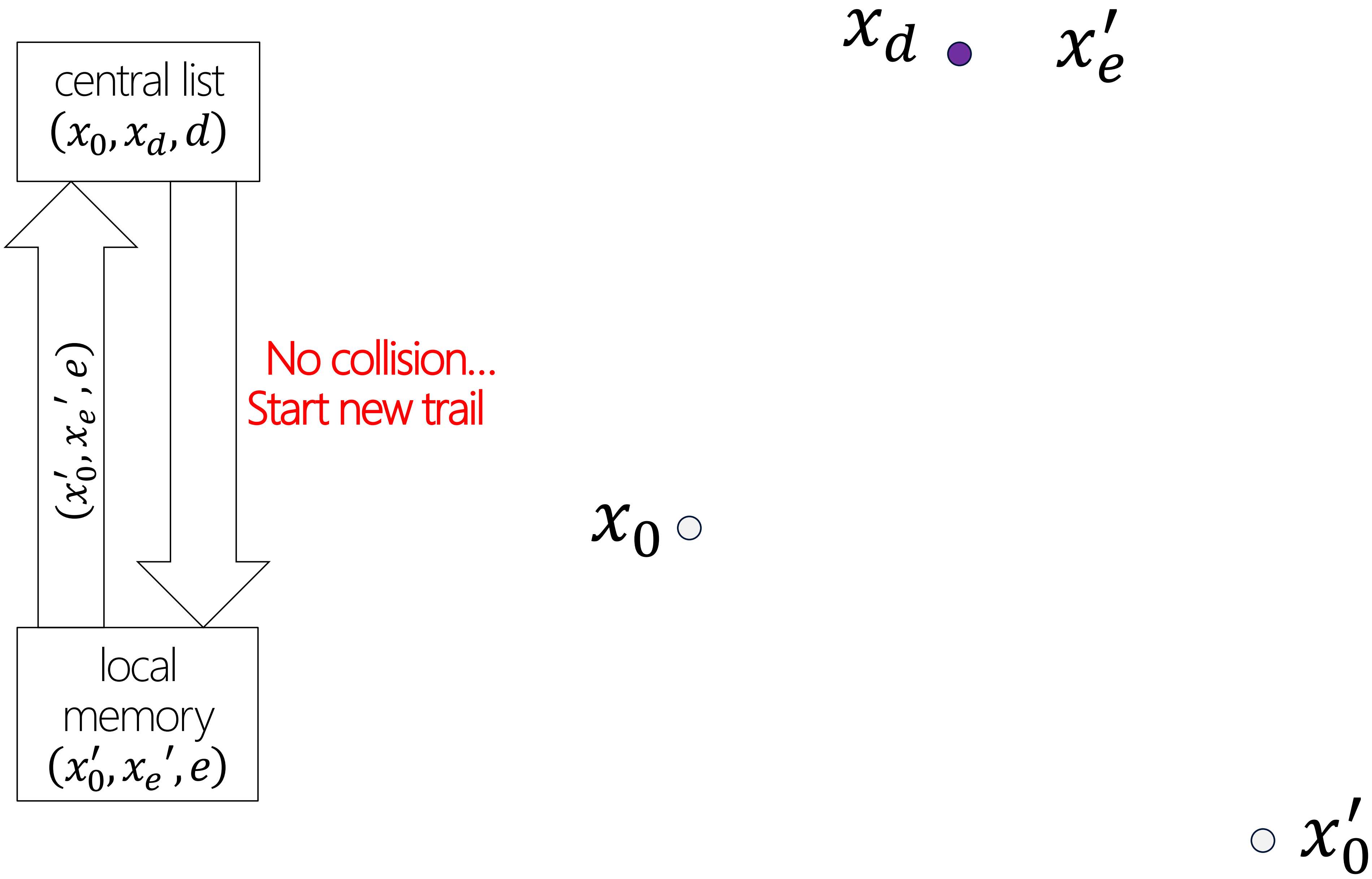
- Precomputation
- Compressed distinguished points
- Optimised isogeny computations
- Multi-target attacks

… thus, (more) precise concrete SIDH/SIKE parameters

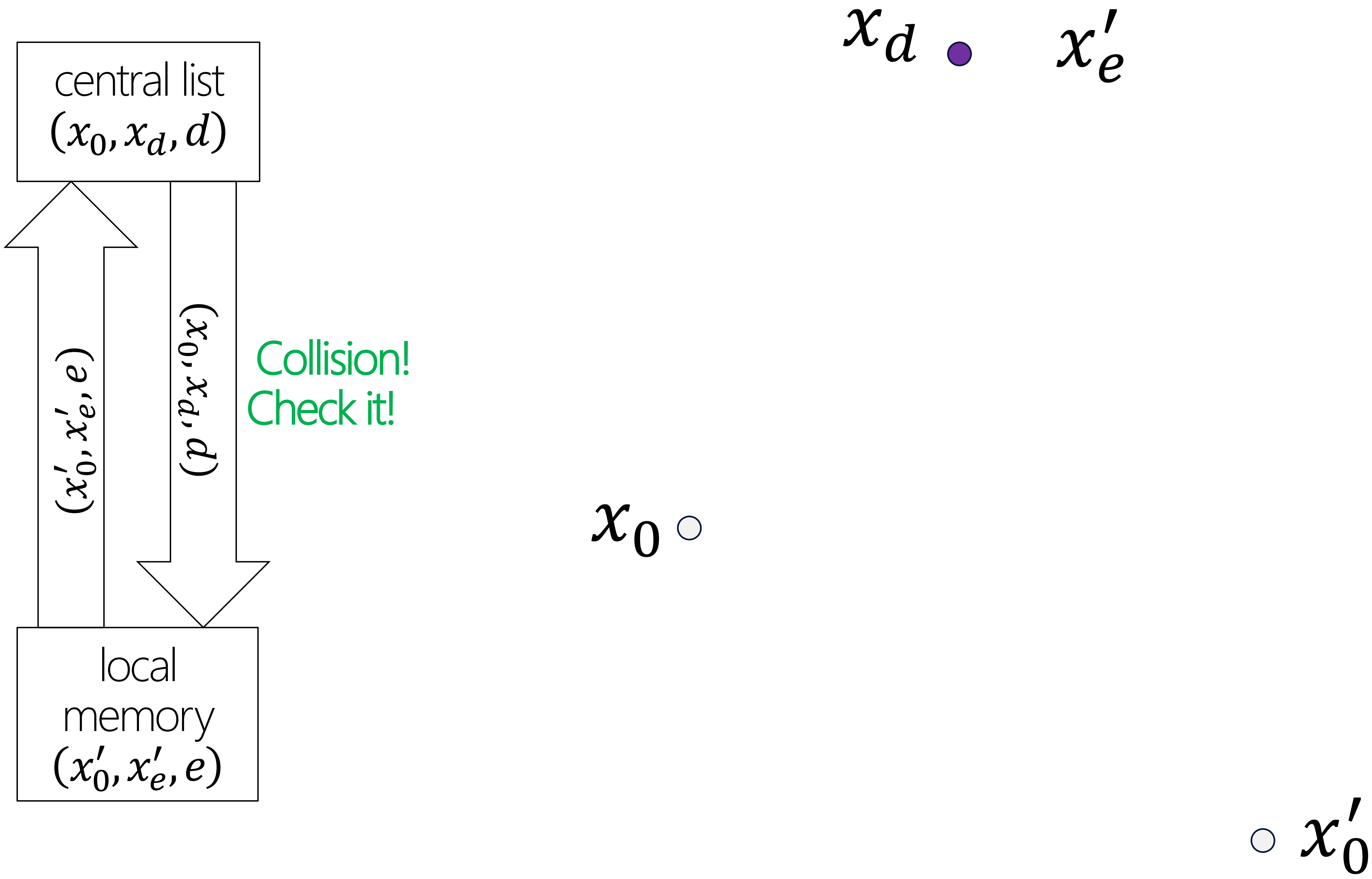
Fast collision checking



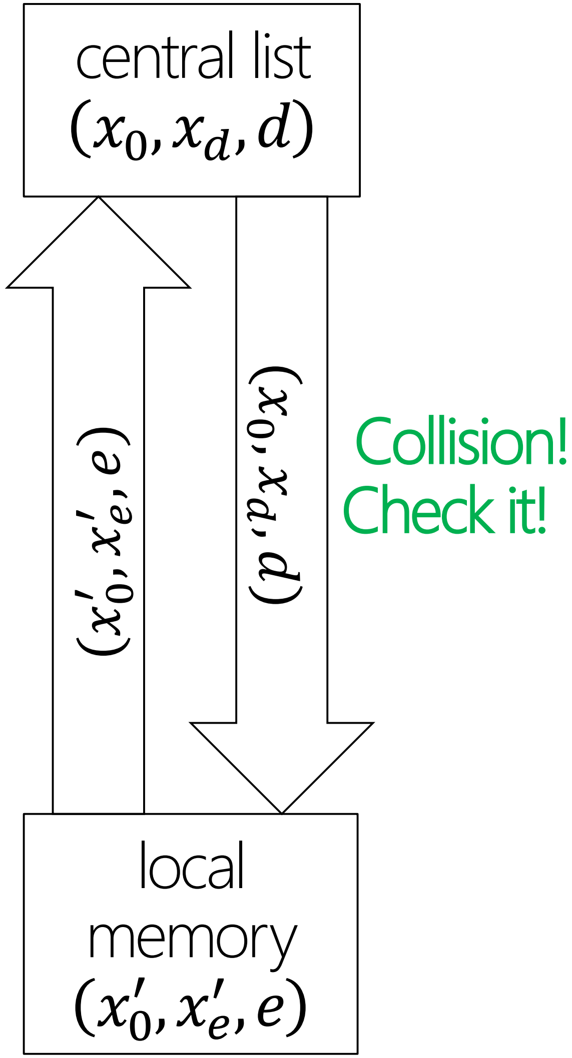
Fast collision checking

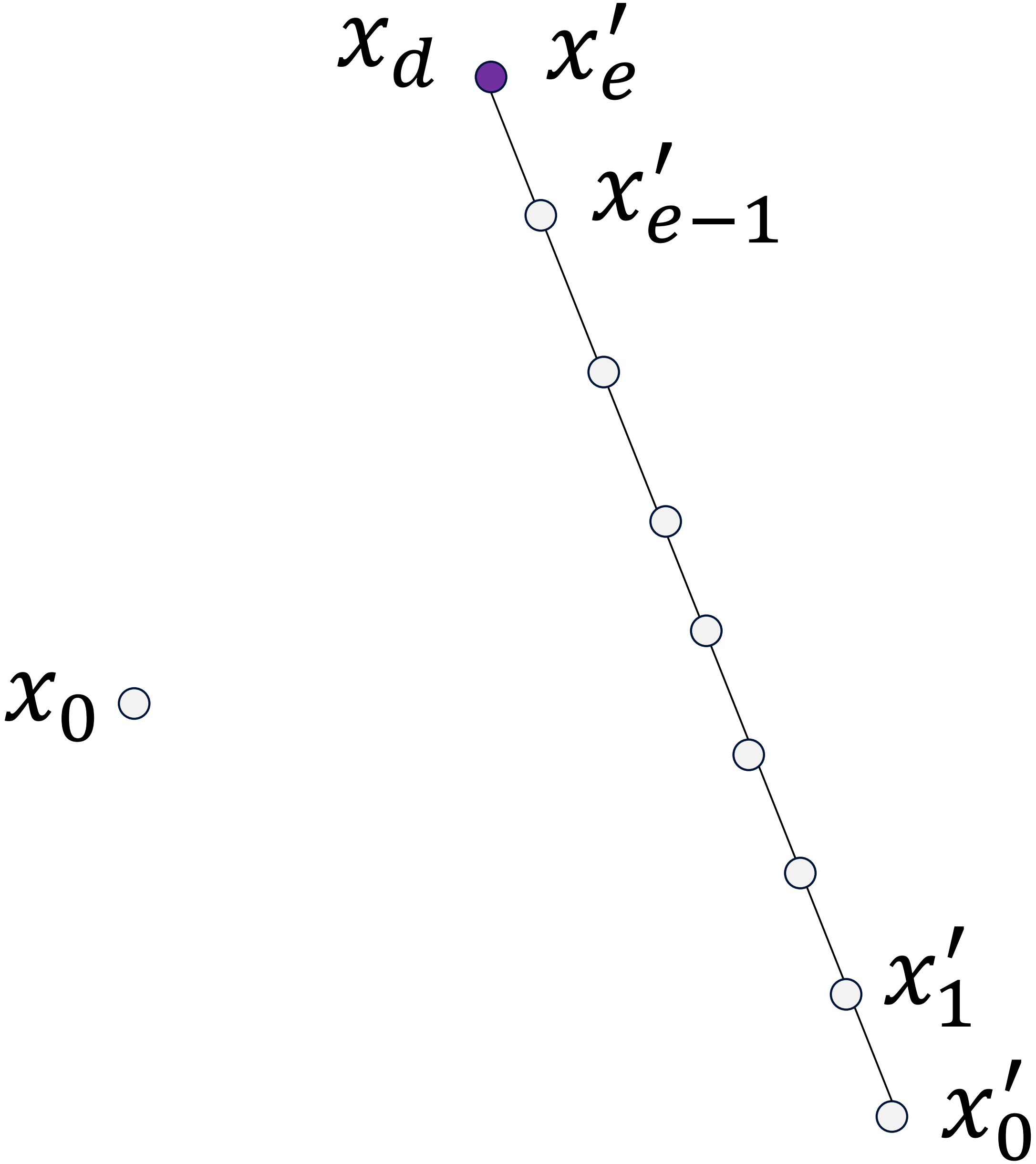


Fast collision checking

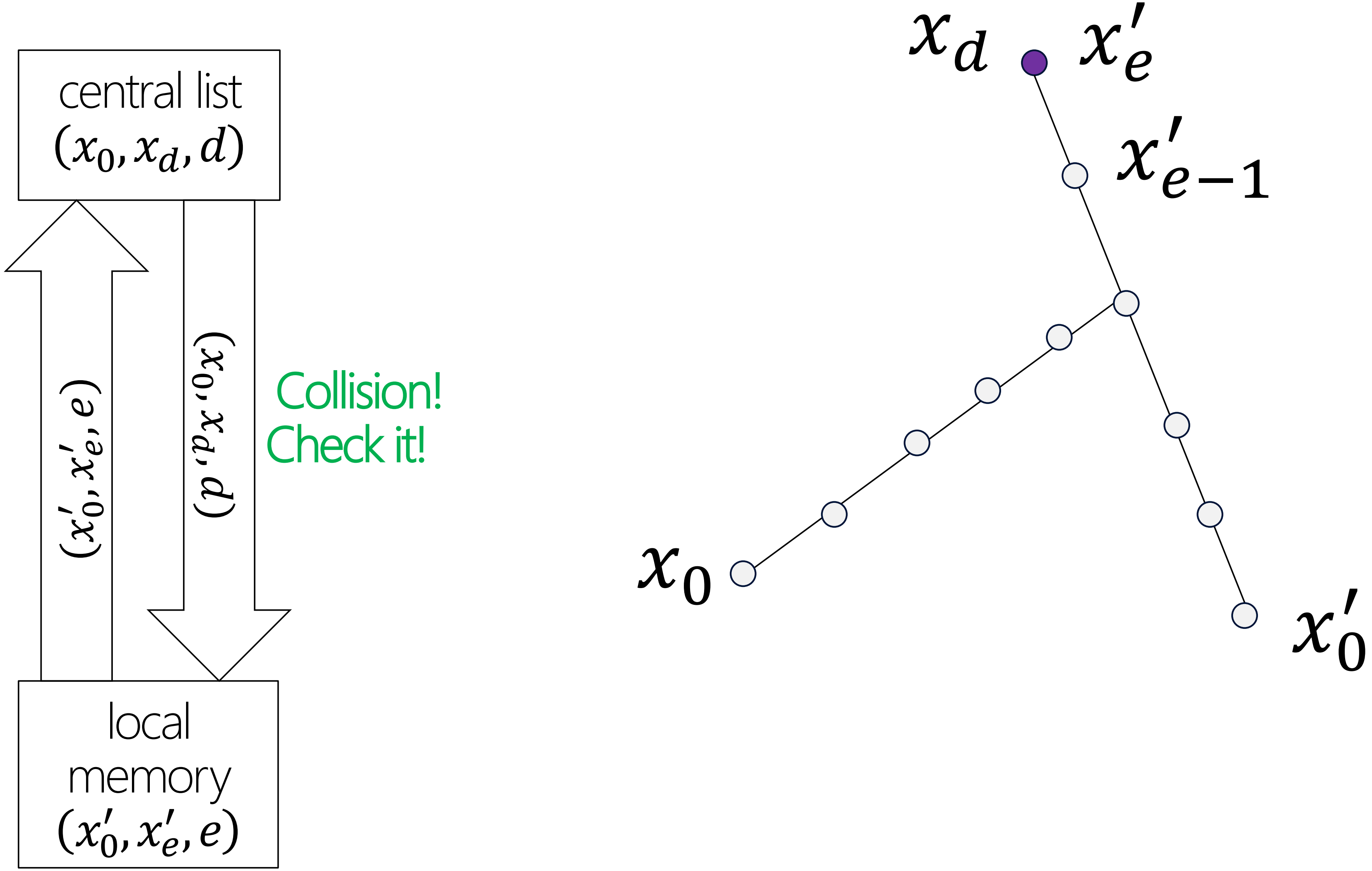


Fast collision checking

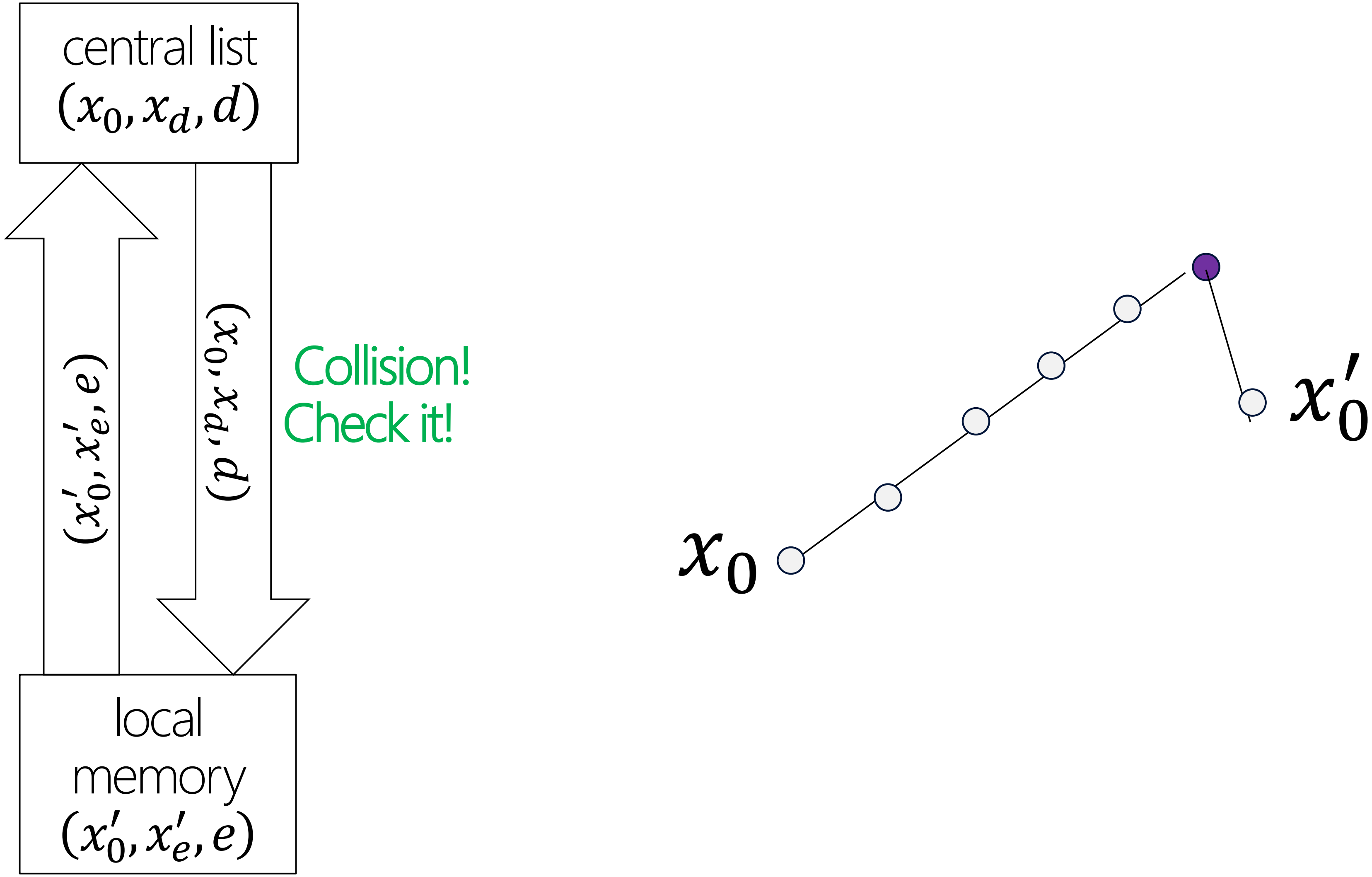




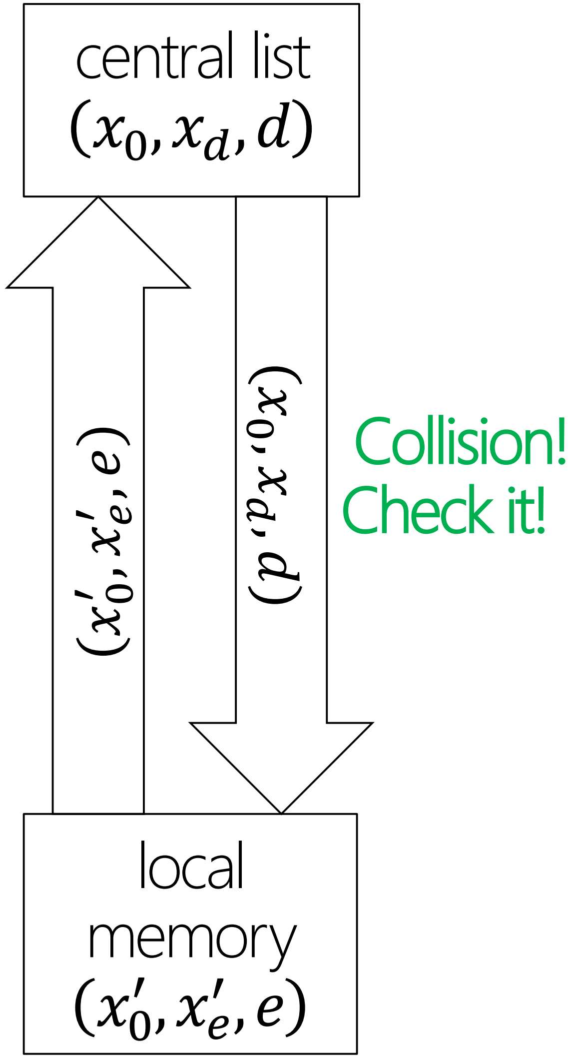
Fast collision checking



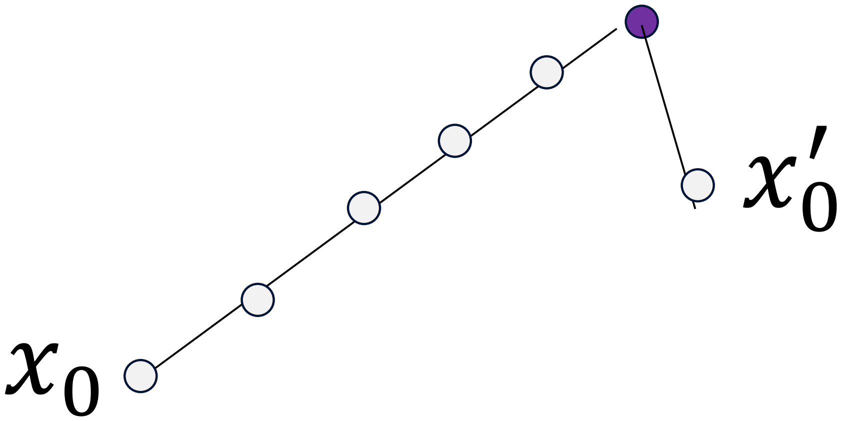
Fast collision checking



Fast collision checking



Now swap sides and repeat



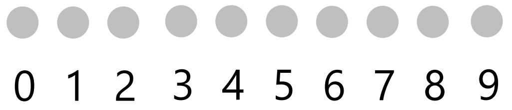
How to leave the trail?

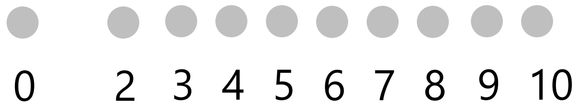
- Sedgewick, Szymanski and Yao., e.g., suppose we can store 10 points…

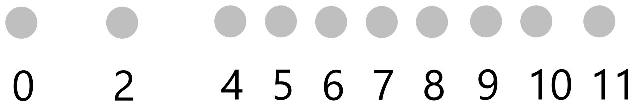
- 0

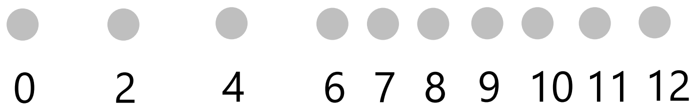

0 1

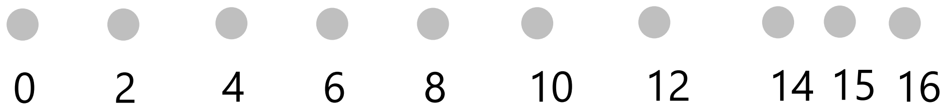
0 1 2

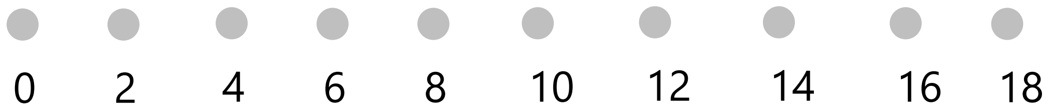

0 1 2 3 4 5 6 7 8 9

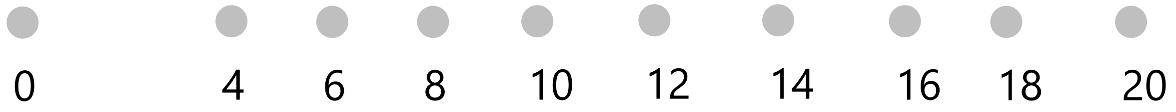

0 2 3 4 5 6 7 8 9 10


0
2
4
5
6
7
8
9
10
11

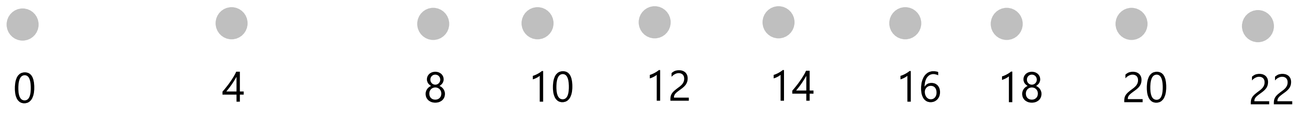

0
2
4
6
7
8
9
10
11
12


0
2
4
6
8
10
12
14
15
16


0
2
4
6
8
10
12
14
16
18

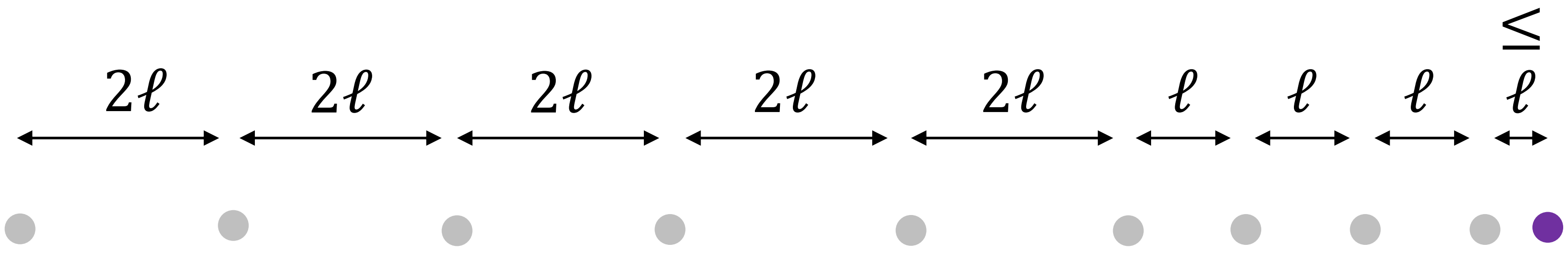

0
4
6
8
10
12
14
16
18
20


0
4
6
8
10
12
14
16
18
20


0
4
8
10
12
14
16
18
20
22


0
4
8
12
16
20
24
28
32
36

Hansel & Gretel a la Sedgewick-Szymanski-Yao…

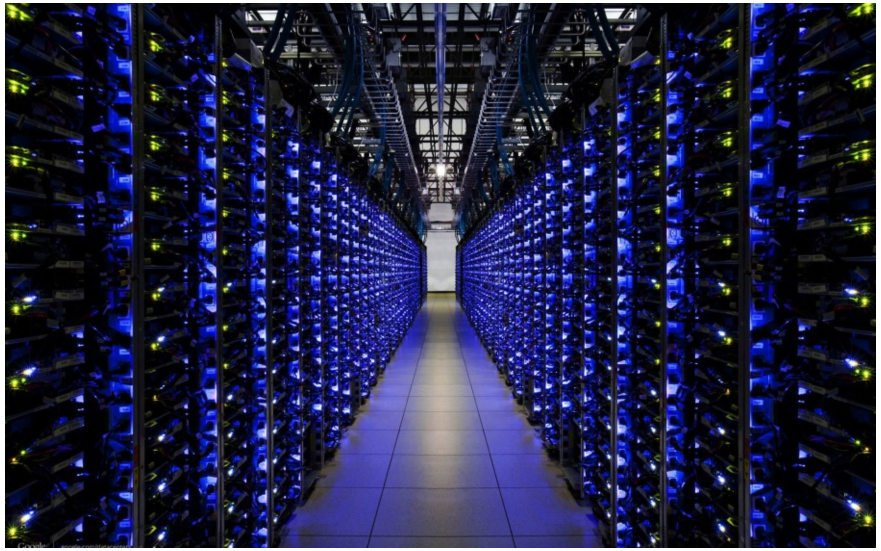


- Hard to analyse average case, but (easy-to-analyse) worst case is way better than previous average collision checking

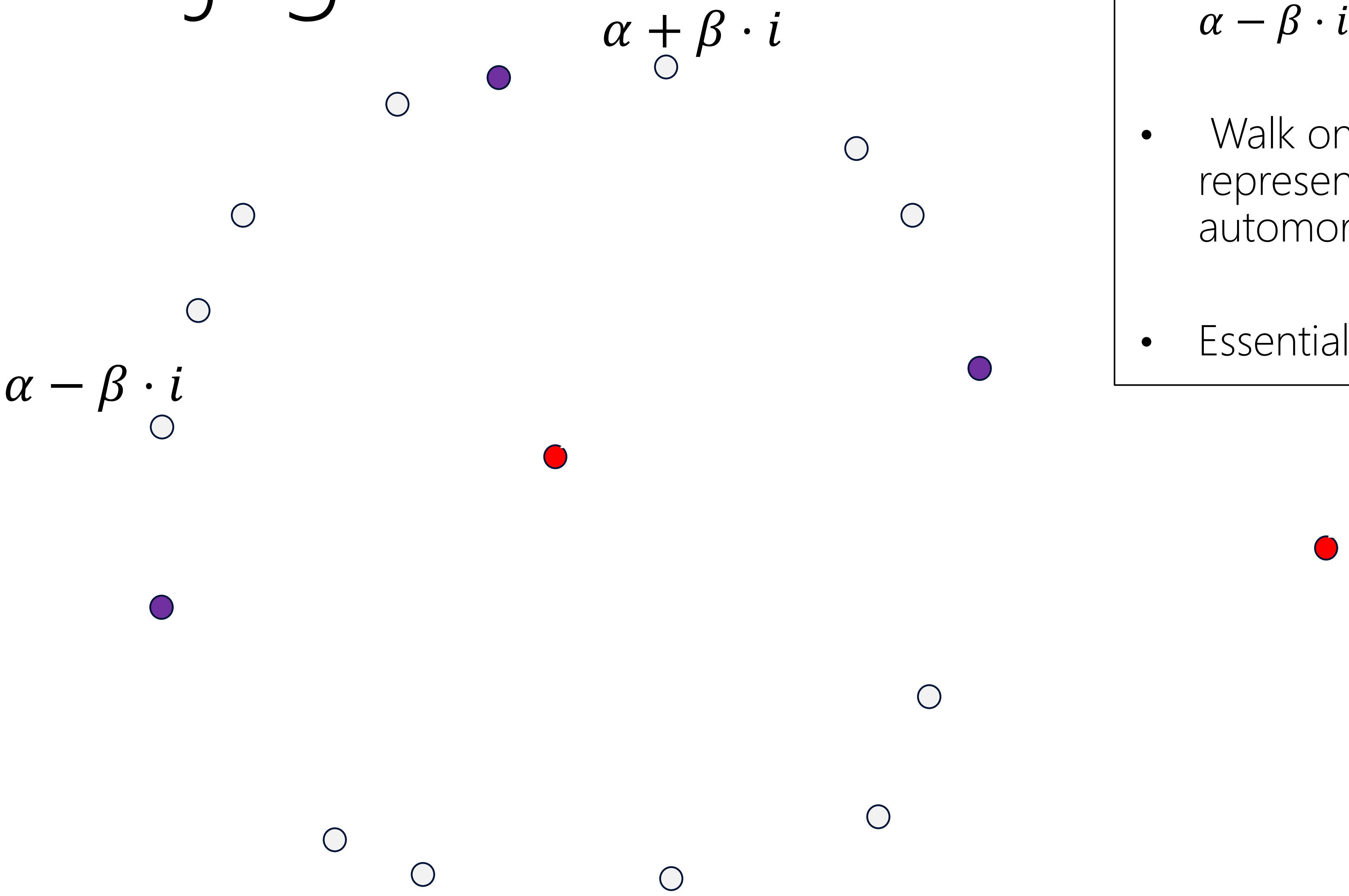
- In practice solid savings…

vOW at scale

- How best to orchestrate a real attack?
- Communication costs are non-trivial. Overhead? Synchronise f_n changes...?
- When/how to check for incoming distinguished points? At both ends? Overhead?
- Large-scale vOW is non-trivial
- This is ongoing...



Conjugates



- For every $\alpha + \beta \cdot i$ reached from left, $\alpha - \beta \cdot i$ is also a possible j-invariant
- Walk on pairs by choosing canonical representative (same as Pollard rho automorphisms/negation map)
- Essentially shrinks set size $|S|$ by 25%

Implications

Target Security Level	**SIKE spec** $\log_2(p)$	**Adj et al SAC 2018** $\log_2(p)$	**SIKE future spec** $\log_2(p)$
NIST 1 (AES128)	503	-	?
NIST 2 (SHA256)	-	434	?
NIST 3 (AES192)	751	-	?
NIST 4 (SHA384)	-	610	?
NIST 5 (AES256)	964	-	?

- ePrint 2018/313: Adj, Cervantes-Vazquez, Chi-Dominguez, Menezes, Rodriguez-Henriquez

Questions?