

# SIKE in Hardware

**Reza Azarderakhsh**

Florida Atlantic University and PQSecure Technologies

June 18, 2019

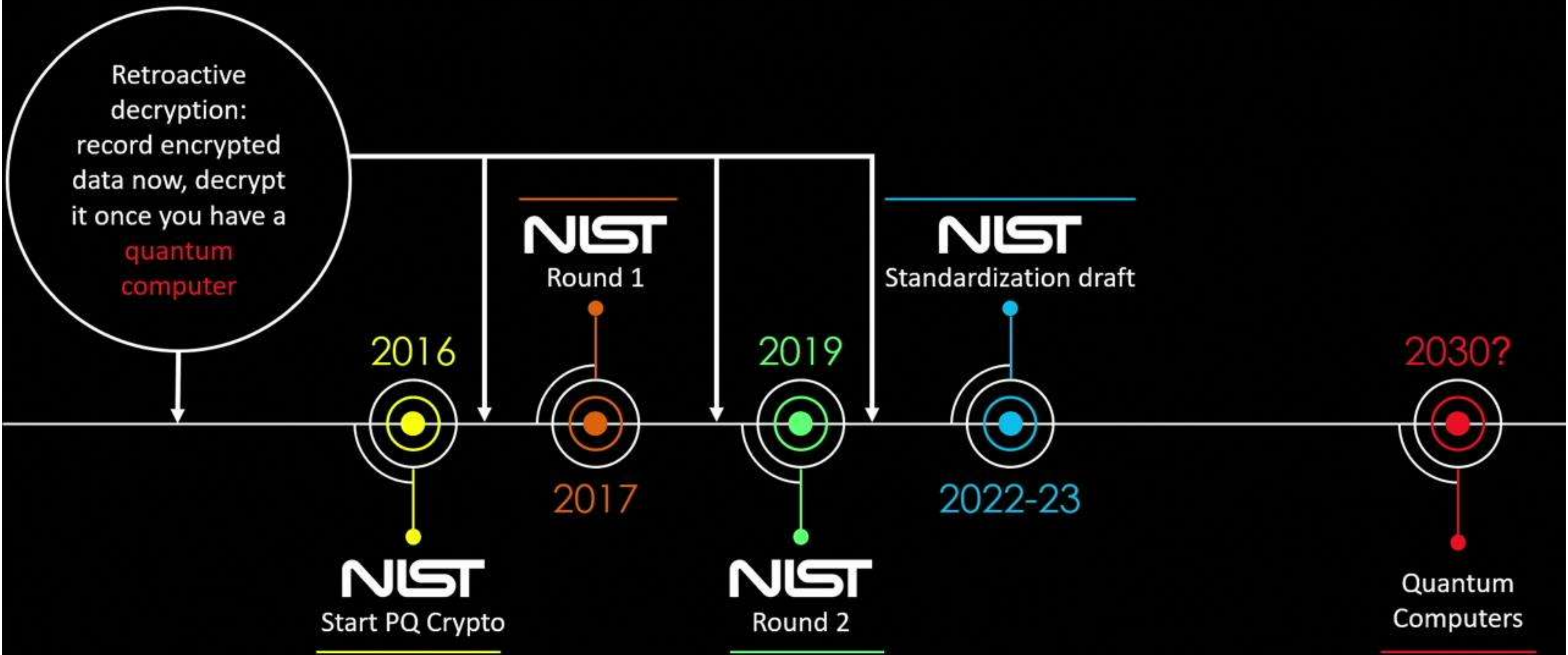
# Quantum Threat to Information Security

Large-scale quantum computers could break some encryption schemes

Need to migrate encryption to quantum-resistant algorithms

When we should start the process?

# Timeline



- Design **better** post-quantum cryptosystems
- Improve classical and quantum **attacks**
- Pick **parameter sizes**
- Develop fast, efficient, and **secure** implementations
- Integrate them into the **existing** infrastructures

## Post-Quantum **Key-Exchange**

Lattice-  
based

Code-  
based

Isogeny-  
based

## Post-Quantum **Signatures**

Lattice-  
based

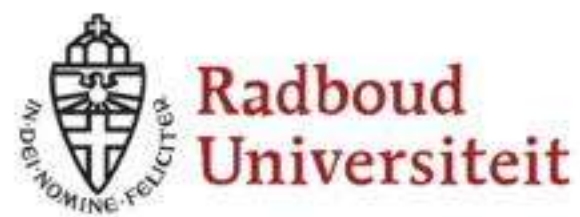
Hash-  
based

Multivariate-  
based

Zero-Knowledge  
based

- **[2006]**: Birth of a **supersingular** isogeny-based cryptosystem
  - Charles – Goren – Lauter
  - built hash function from supersingular isogeny graph
- **[2011]**: Supersingular isogeny key exchange
  - Jao – De Feo
- **[2017]**: Supersingular isogeny key encapsulation
  - SIKE Team

# SIKE Team

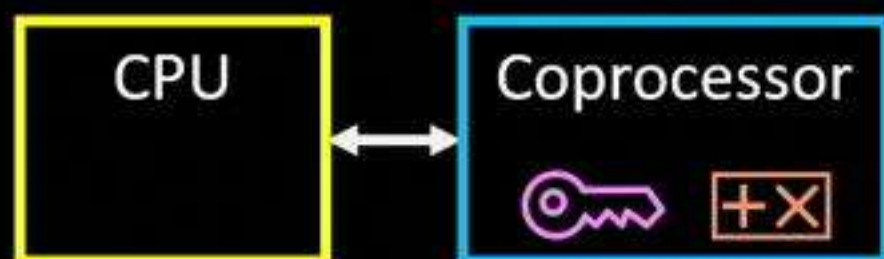


Microsoft Research



# Architecture Selection for Cryptographic Design

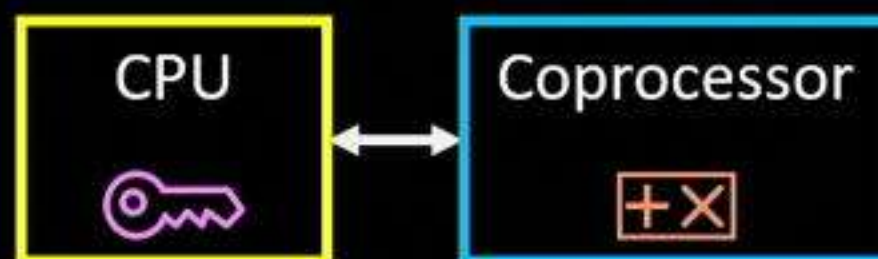
## HW only



+ Highly **optimized** for dedicated purpose (power consumption, execution time, security)

- Extra HW costs
- limited flexibility
- HW design effort/complexity

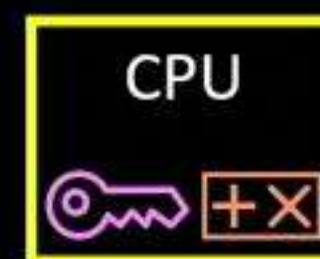
## HW/SW



+ Good **trade-off** between optimization/costs (still fast but less design effort/complexity easier to handle)

- + Higher flexibility
- Not straight-forward to find optimal HW/SW partitioning
- Extra HW costs
- Less optimized than HW-only

## SW only



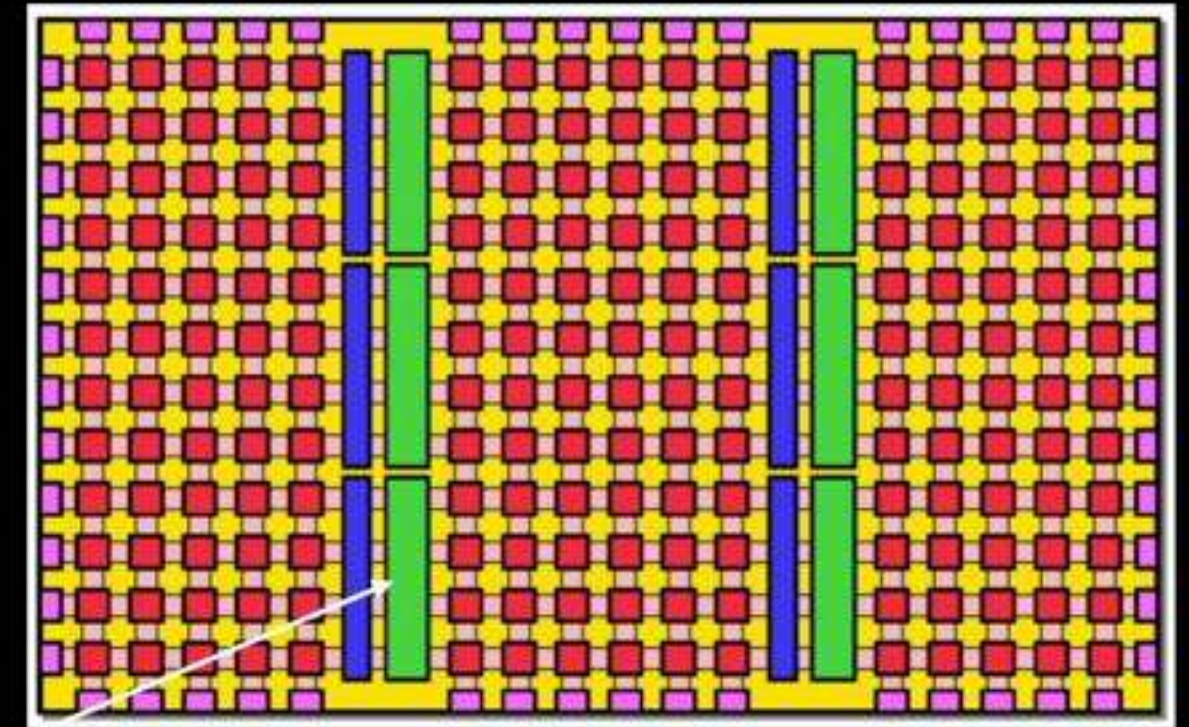
+ Limited HW costs (code/data storage)

- + High **flexibility**
- + Minimal HW design effort/eases handling of complexity (programming)
- Not optimized (energy, consumption, performance)



# FPGAs: Field Programmable Gate Arrays

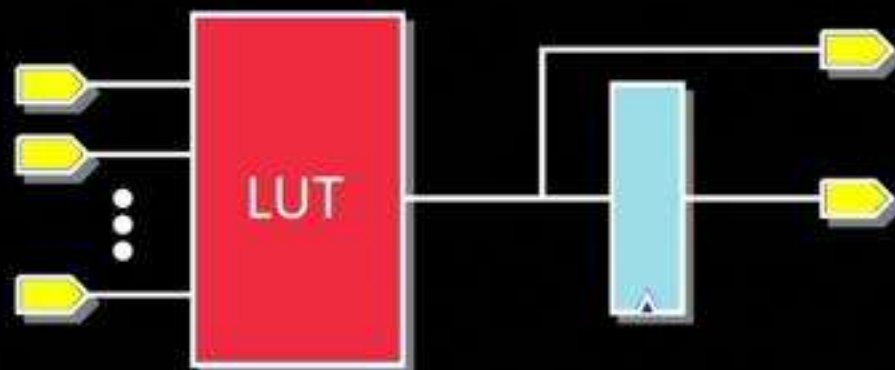
- FPGAs are composed of:
- Programmable **logic cells**
- A configurable **routing matrix**
- configurable **input/output** cells
- Embedded **memory blocks**
- Small embedded **multipliers**
- etc.



18-bit×18-bit **multiplier blocks**

Inside a logic cell:

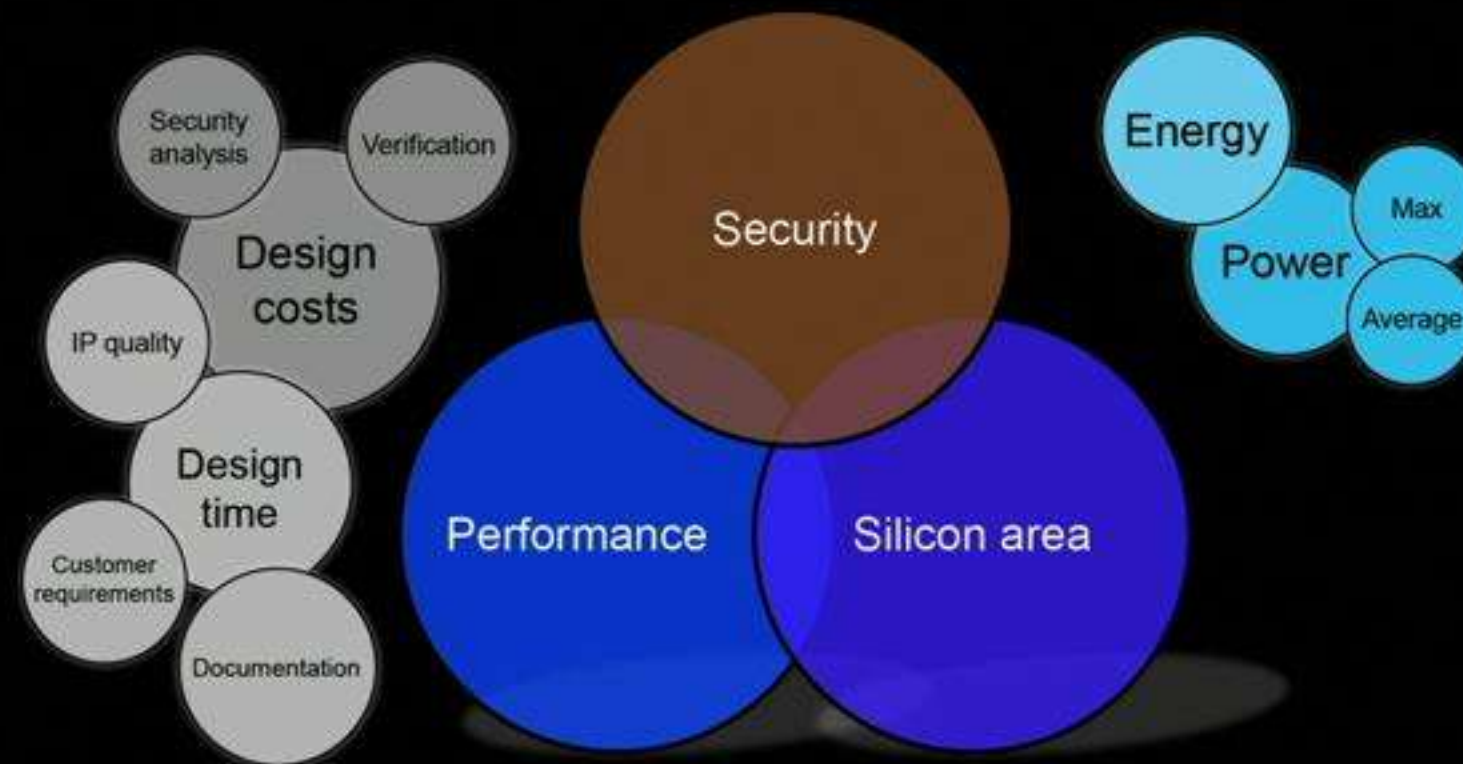
- Connections to the routing matrix
- Programmable **lookup-tables**
  - 4 inputs, 1 output
  - 6 inputs, 1 output
  - 6 inputs, 2 outputs
- optional **registers**
  - free **pipelining**
- more logic for **fast carry-propagation**



# FPGAs vs. ASIC

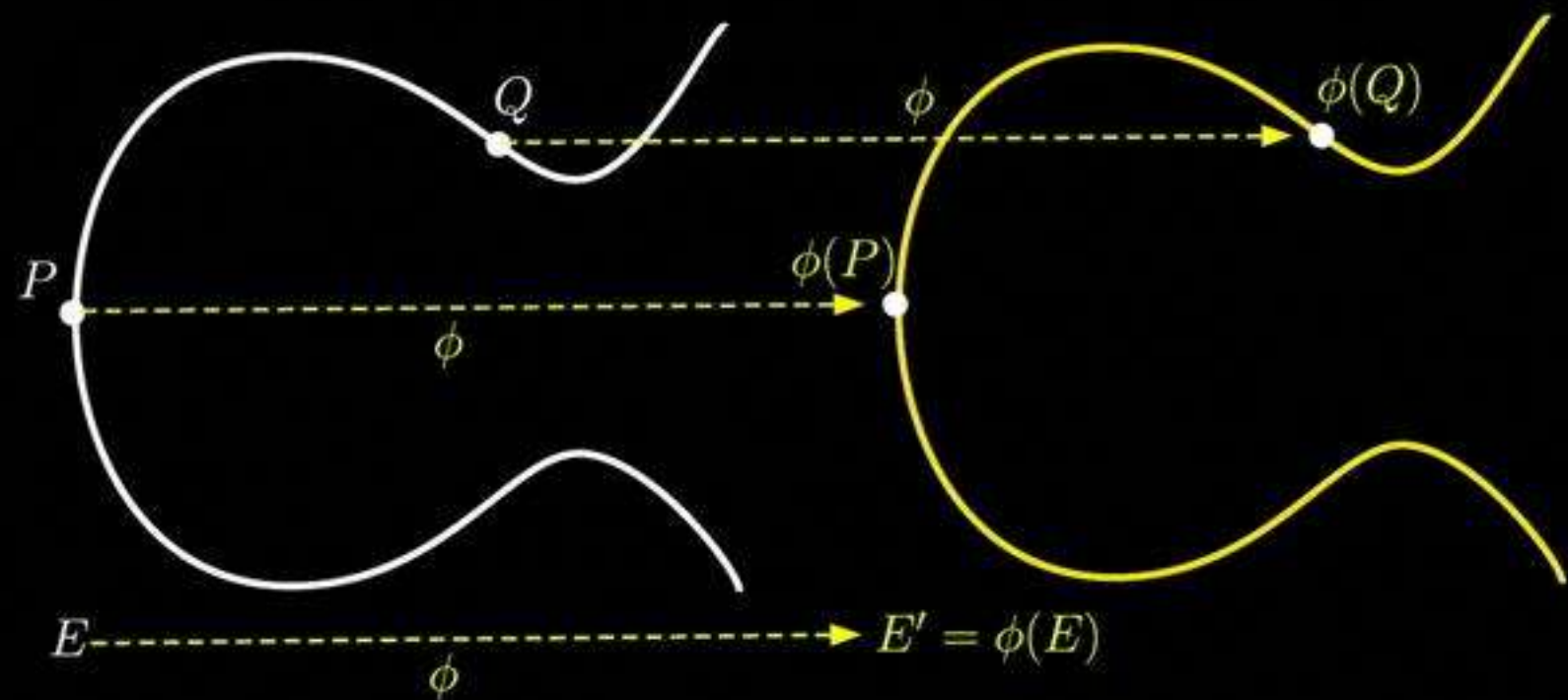
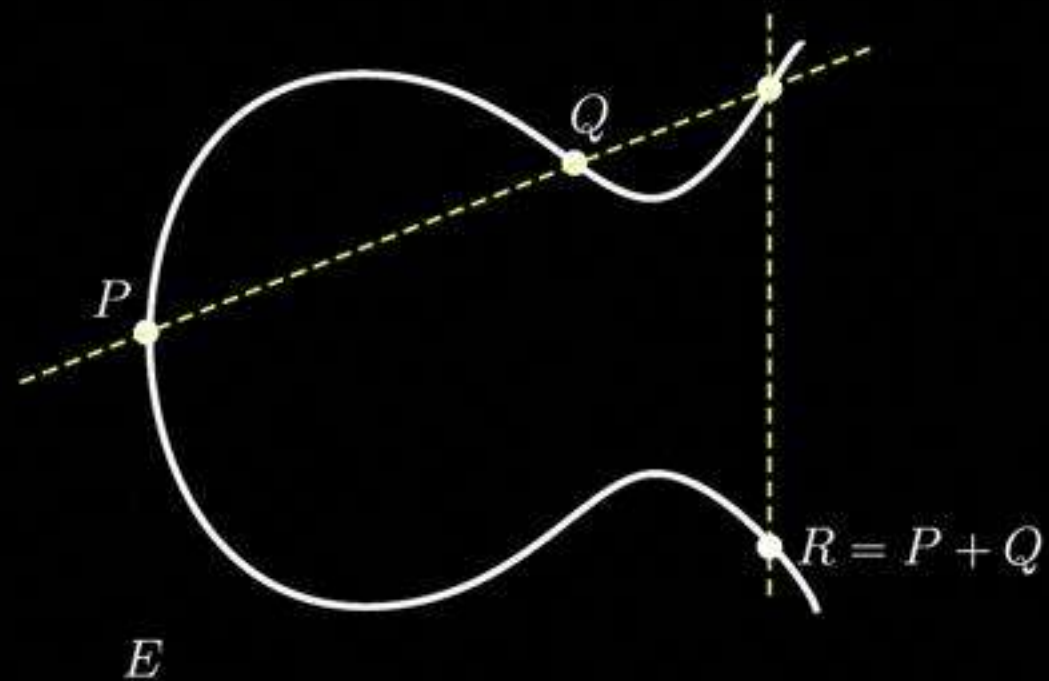
- + prototyping
- + re-usability
- + short time to market
- + simpler design cycle
- + Programmable in the field
- + hardware/software co-design

- speed
- silicon footprint
- power and energy consumption
- low cost for high volumes
- better performance
- reconfigurability and redundancy



# Isogeny-Based Cryptography

- Isogeny-based cryptography is constructed on **a set of curves**.
- Given two curve  $E$  and  $E' = \phi(E)$  find  $\phi$ ?



# Supersingular Isomorphism Classes

- We are interested in the set of **supersingular** curves (up to isomorphism) over a specific field
- Prime  $p = 2^{e_A} \cdot 3^{e_B} \cdot f \pm 1$
- Elliptic curves over  $\mathbb{F}_{p^2}$ ,  $\#E = (p \mp 1)^2$
- Supersingular  **$j$ -invariants**:  $\#S_{p^2} \approx \left\lfloor \frac{p}{12} \right\rfloor$   
(isogenous elliptic curves)



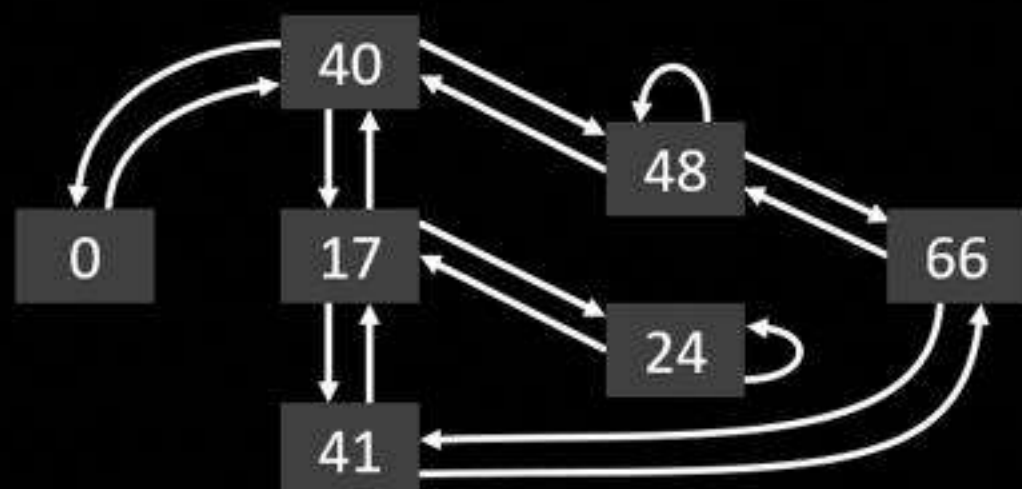
Prime  $p = 2^3 \cdot 3^2 - 1 = 71$ ,  $\#E = 72^2$ ,  $\#S_{p^2} = 7$

# Isogeny Graphs

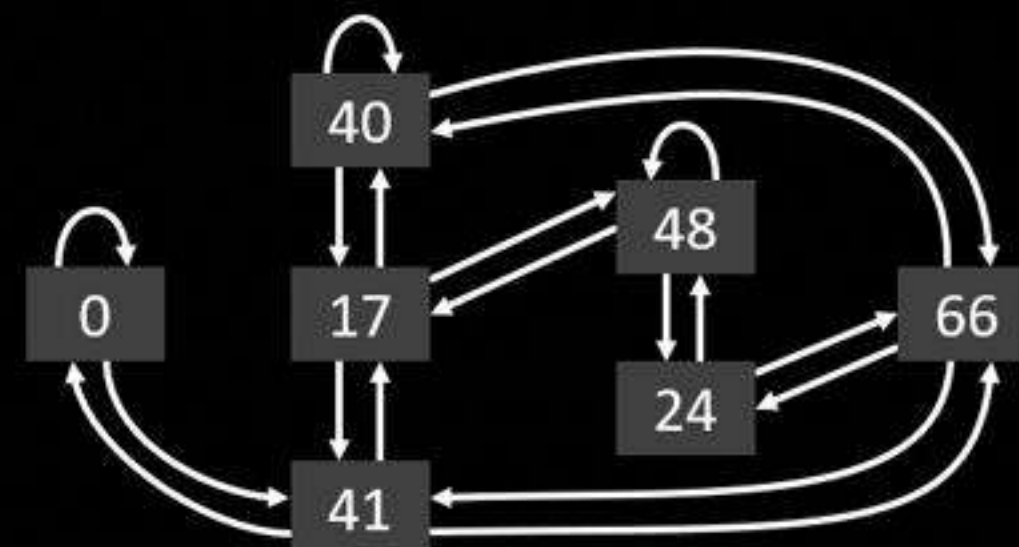
**Vertices:** All isogenous elliptic curves over  $\mathbb{F}_{p^2}$ .

**Edges:** Isogenies of degree  $\ell$

With isogeny of degree  $\ell$ , we get a **connected**  $(\ell + 1)$ -regular graph.



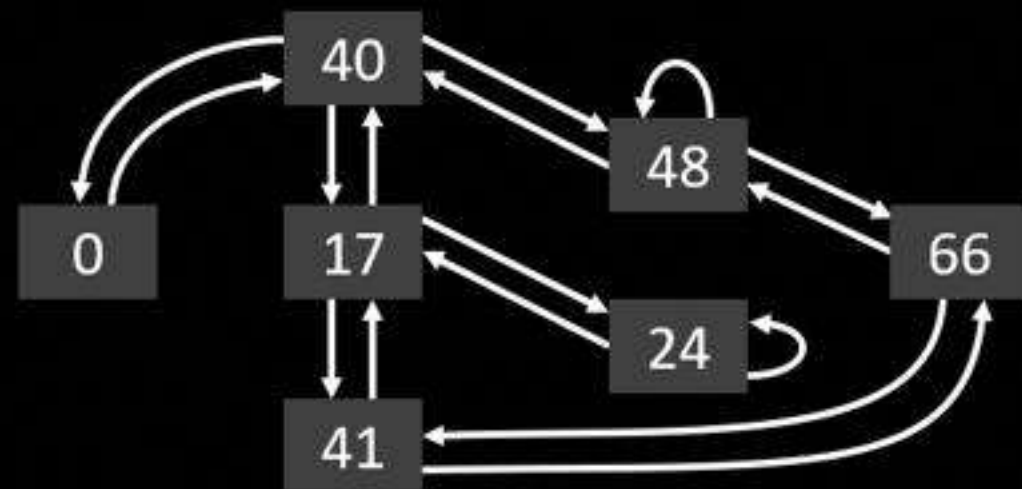
2-isogeny graph



3-isogeny graph

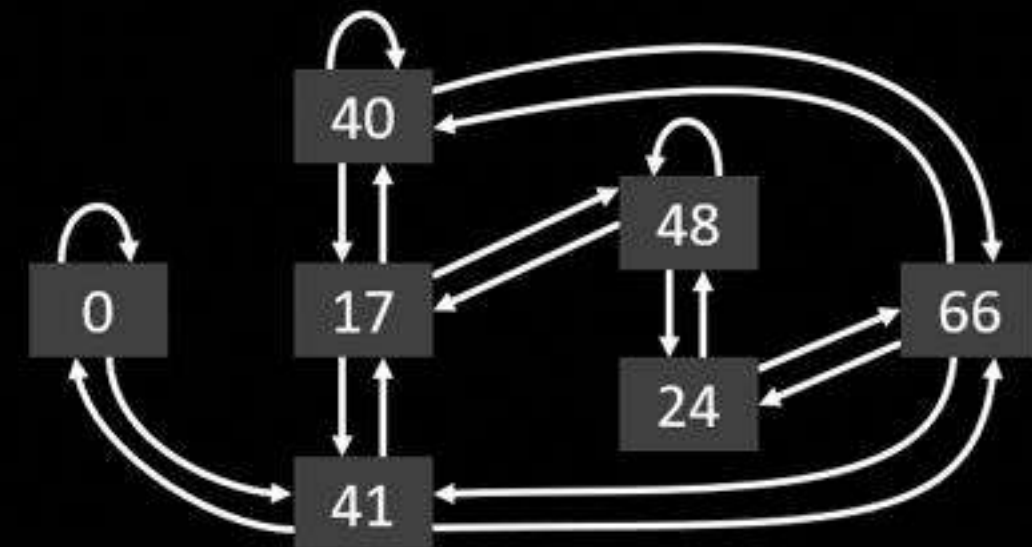
# Key Exchange based on Isogeny Graphs

Alice



2-isogeny graph

Bob



3-isogeny graph

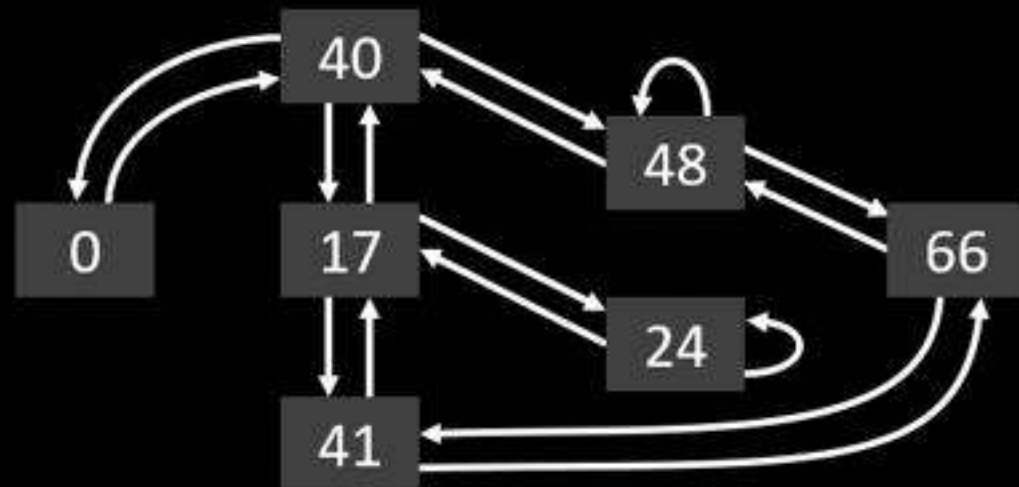
# Public Parameters

$$E_0/\mathbb{F}_{p^2}$$

$$\{P_A, Q_A\} \in E_0[2^{e_A}]$$

$$\{P_B, Q_B\} \in E_0[3^{e_B}]$$

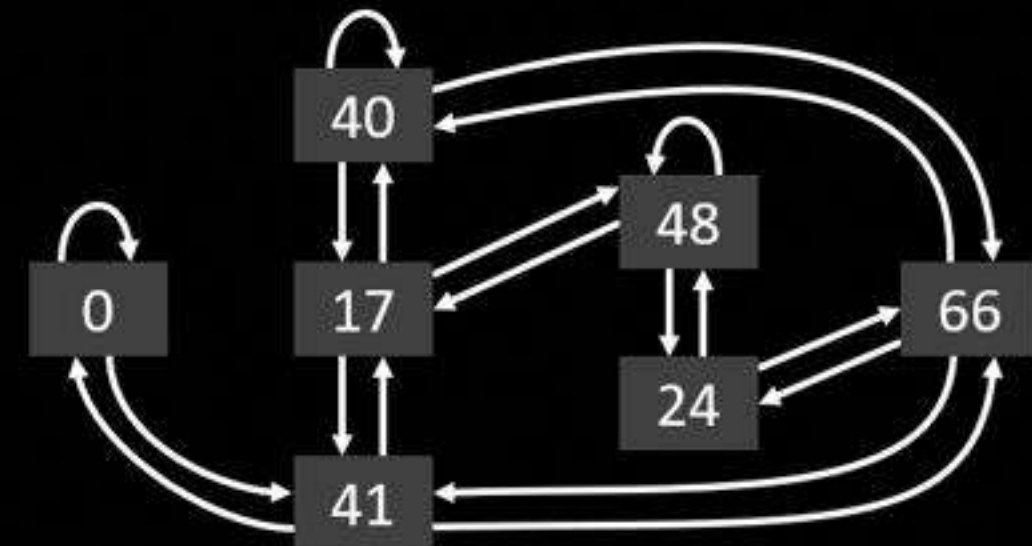
Alice



$$P_A = (53, 55)$$

$$Q_A = (18, 27w + 44)$$

Bob



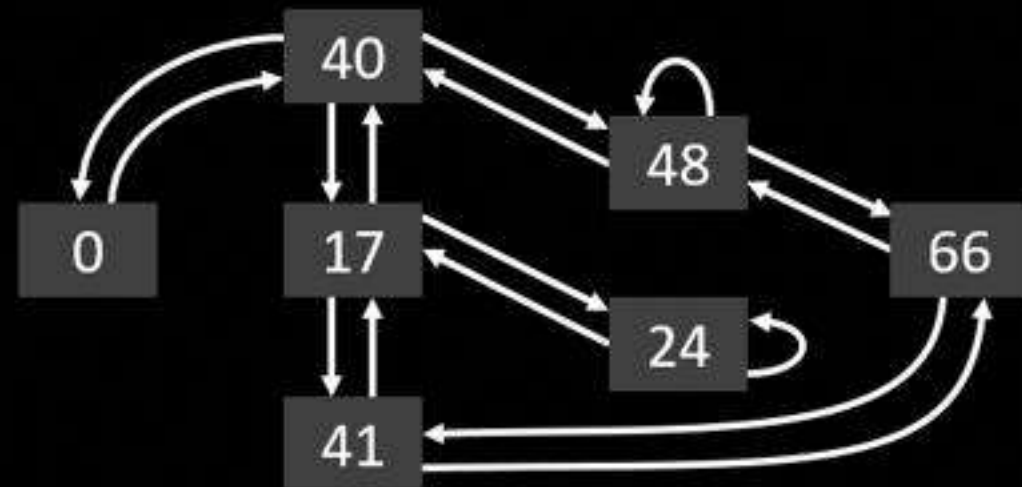
$$P_B = (7w + 20, 31w + 50)$$

$$Q_B = (21w + 64, 38w + 13)$$

$$E_0: y^2 = x^3 + x$$

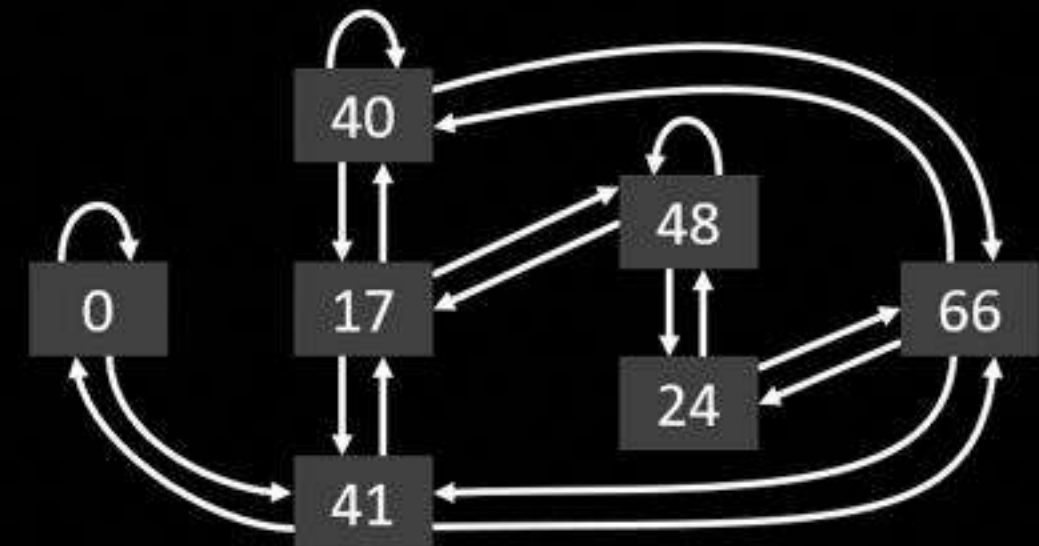
# Key Exchange based on Isogeny Graphs

Alice



2-isogeny graph

Bob



3-isogeny graph



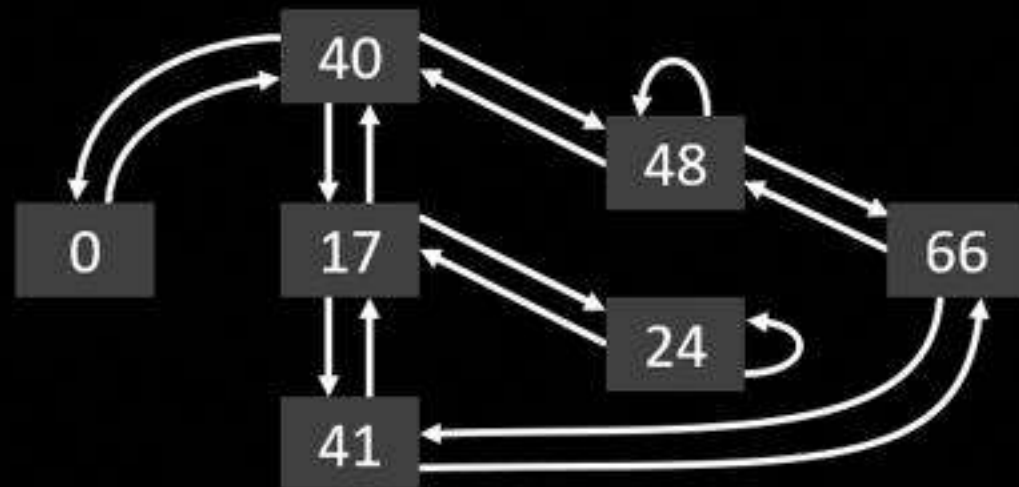
# Public Parameters

$$E_0/\mathbb{F}_{p^2}$$

$$\{P_A, Q_A\} \in E_0[2^{e_A}]$$

$$\{P_B, Q_B\} \in E_0[3^{e_B}]$$

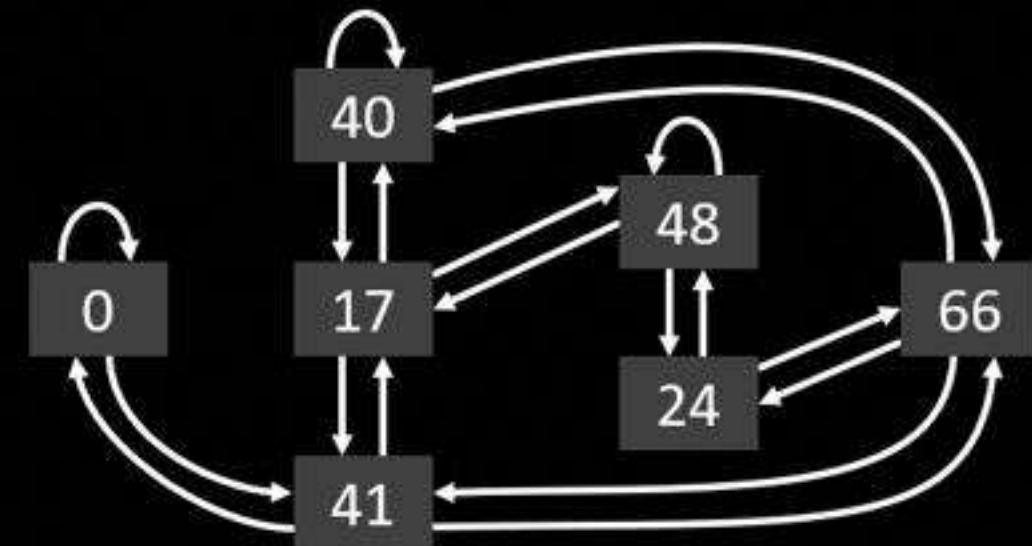
Alice



$$P_A = (53, 55)$$

$$Q_A = (18, 27w + 44)$$

Bob



$$P_B = (7w + 20, 31w + 50)$$

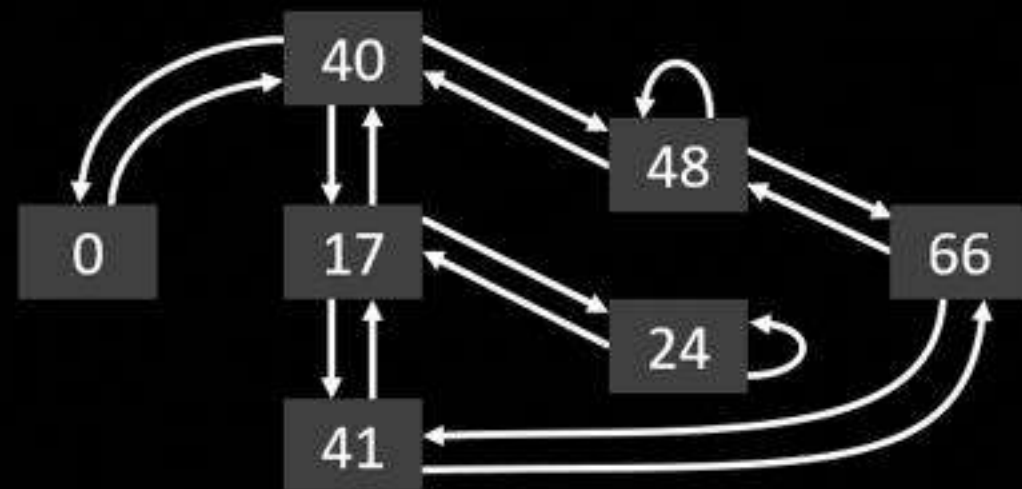
$$Q_B = (21w + 64, 38w + 13)$$

$$E_0: y^2 = x^3 + x$$

# Secret Key

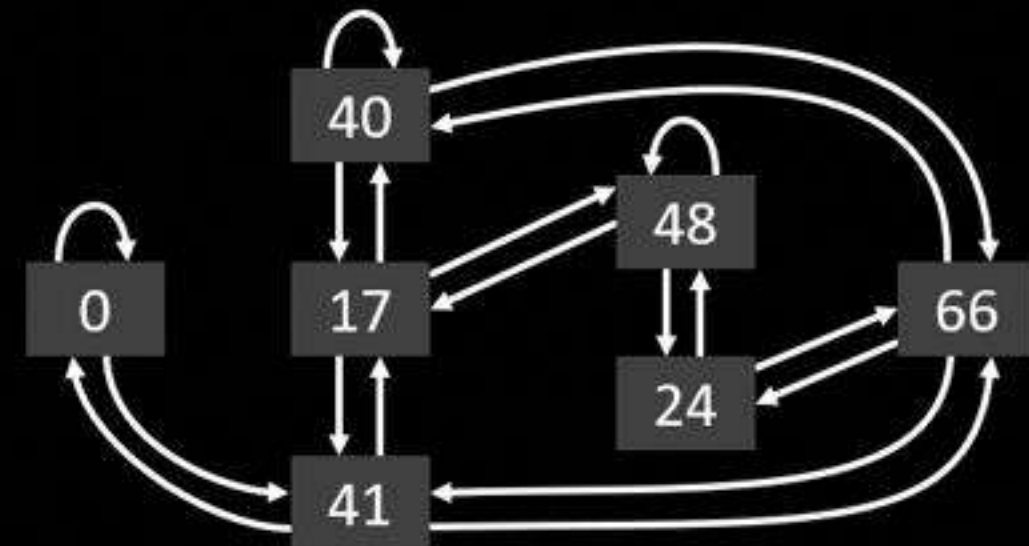
$$s_A \in [0, 2^{e_A})$$
$$s_B \in [0, 3^{e_B})$$

Alice



$$s_A = 6$$

Bob

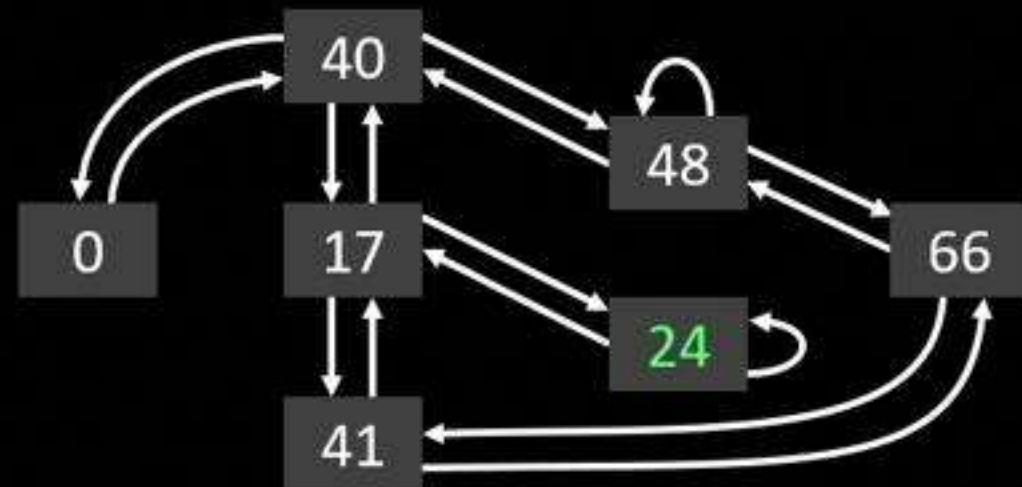


$$s_B = 3$$

# Public Key Generation

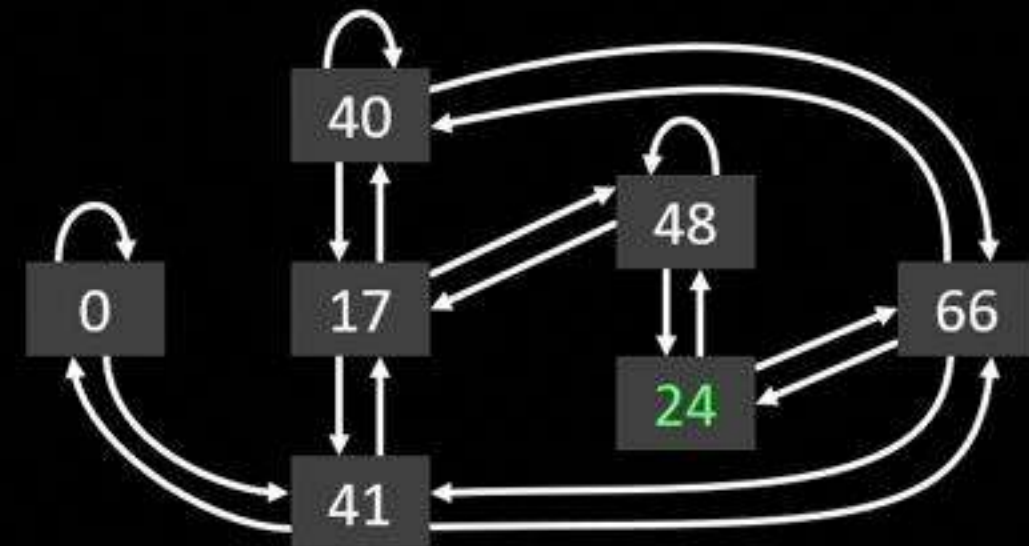
$E_0$

Alice



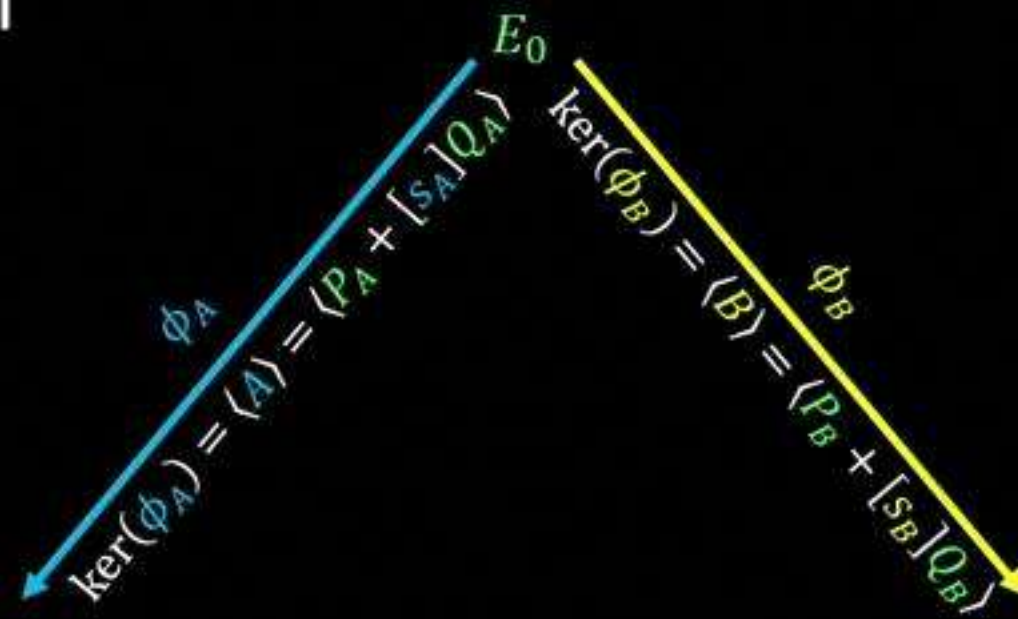
$$E_0: y^2 = x^3 + x$$

Bob

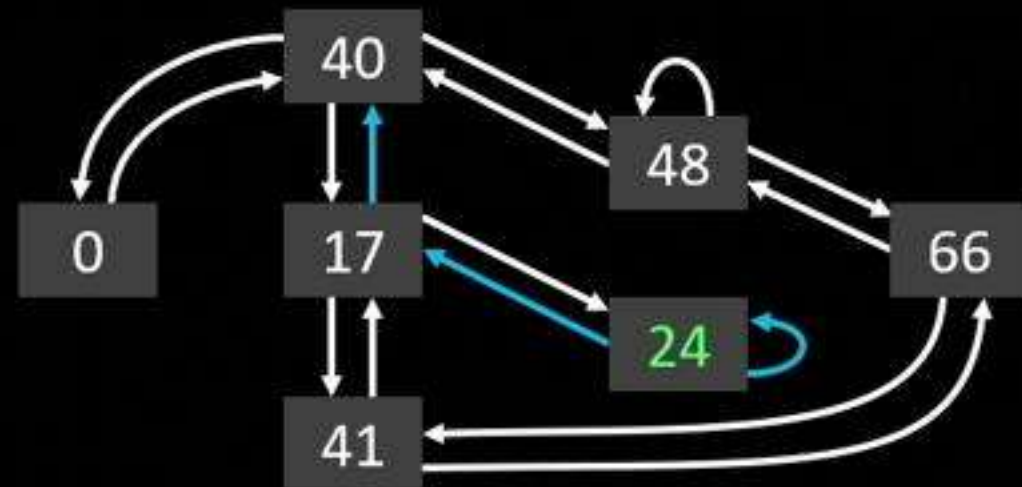


$$E_0: y^2 = x^3 + x$$

# Public Key Generation



Alice

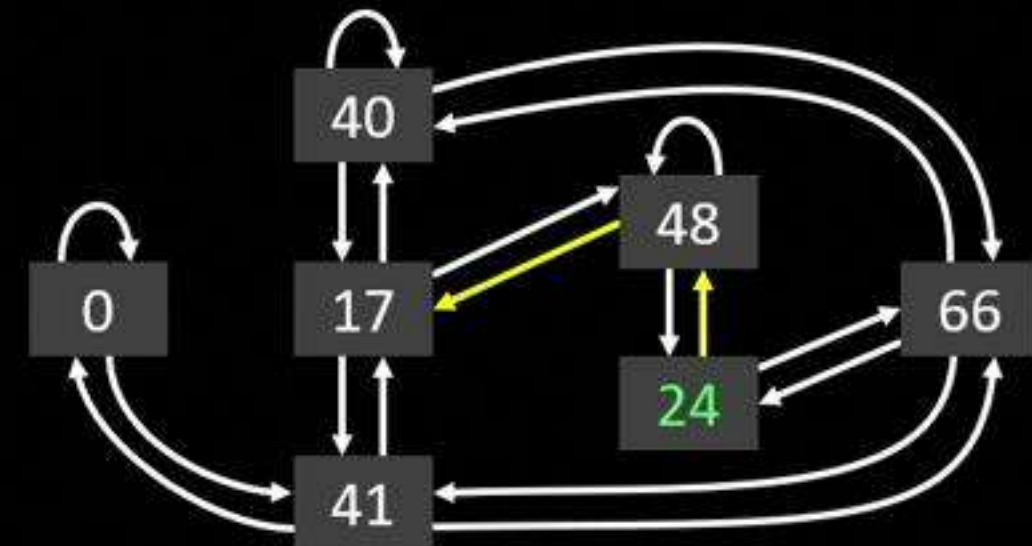


$$E_0: y^2 = x^3 + x$$

$$\phi_A: E_0 \rightarrow E_A$$

$$p = 2^3 \cdot 3^2 - 1 = 71$$

Bob



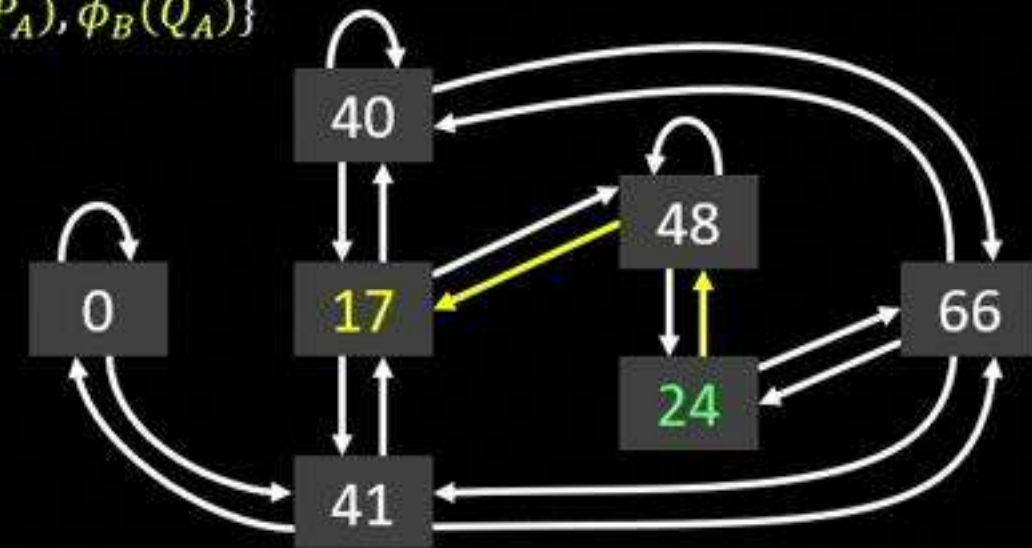
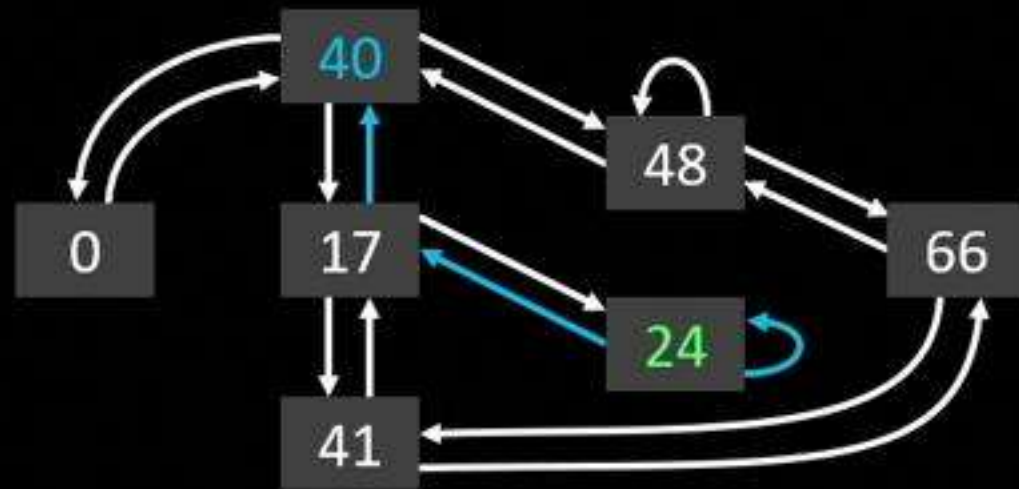
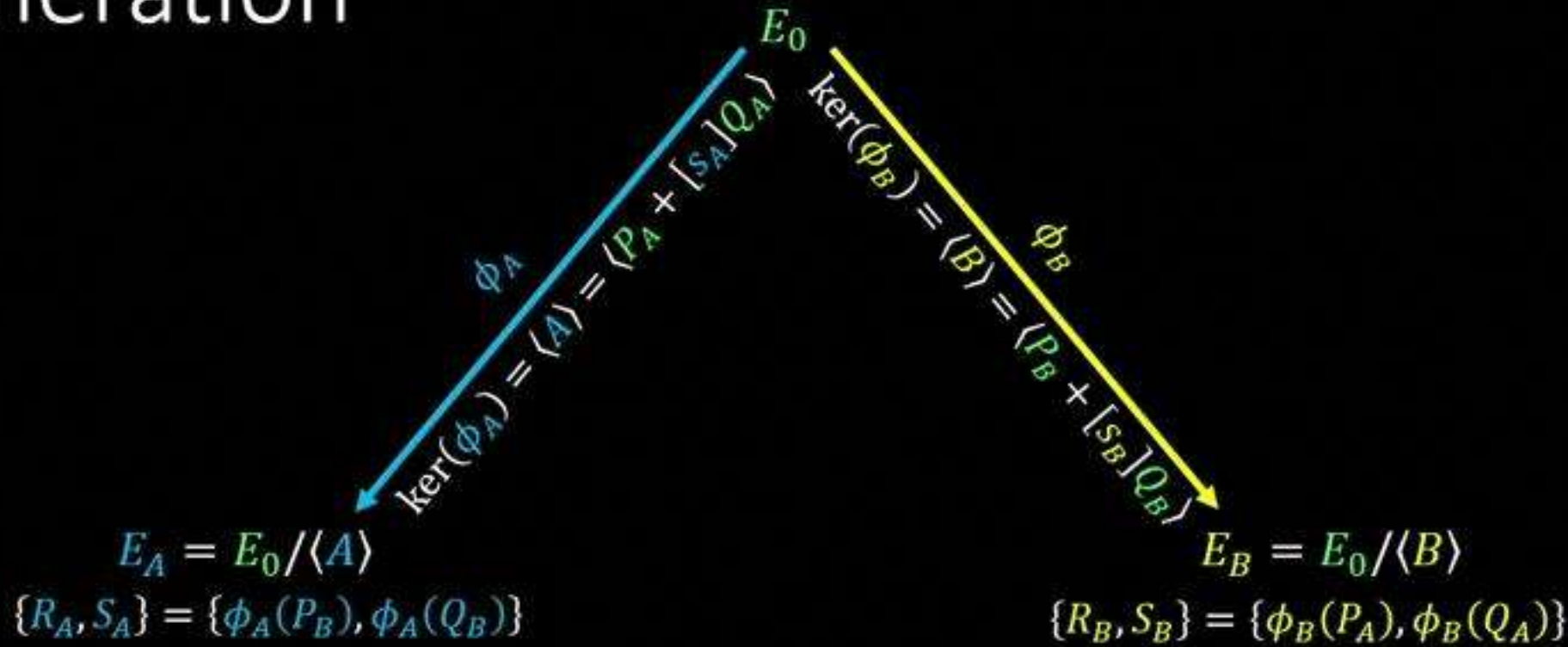
$$E_0: y^2 = x^3 + x$$

$$\phi_B: E_0 \rightarrow E_B$$

# Public Key Generation

Alice

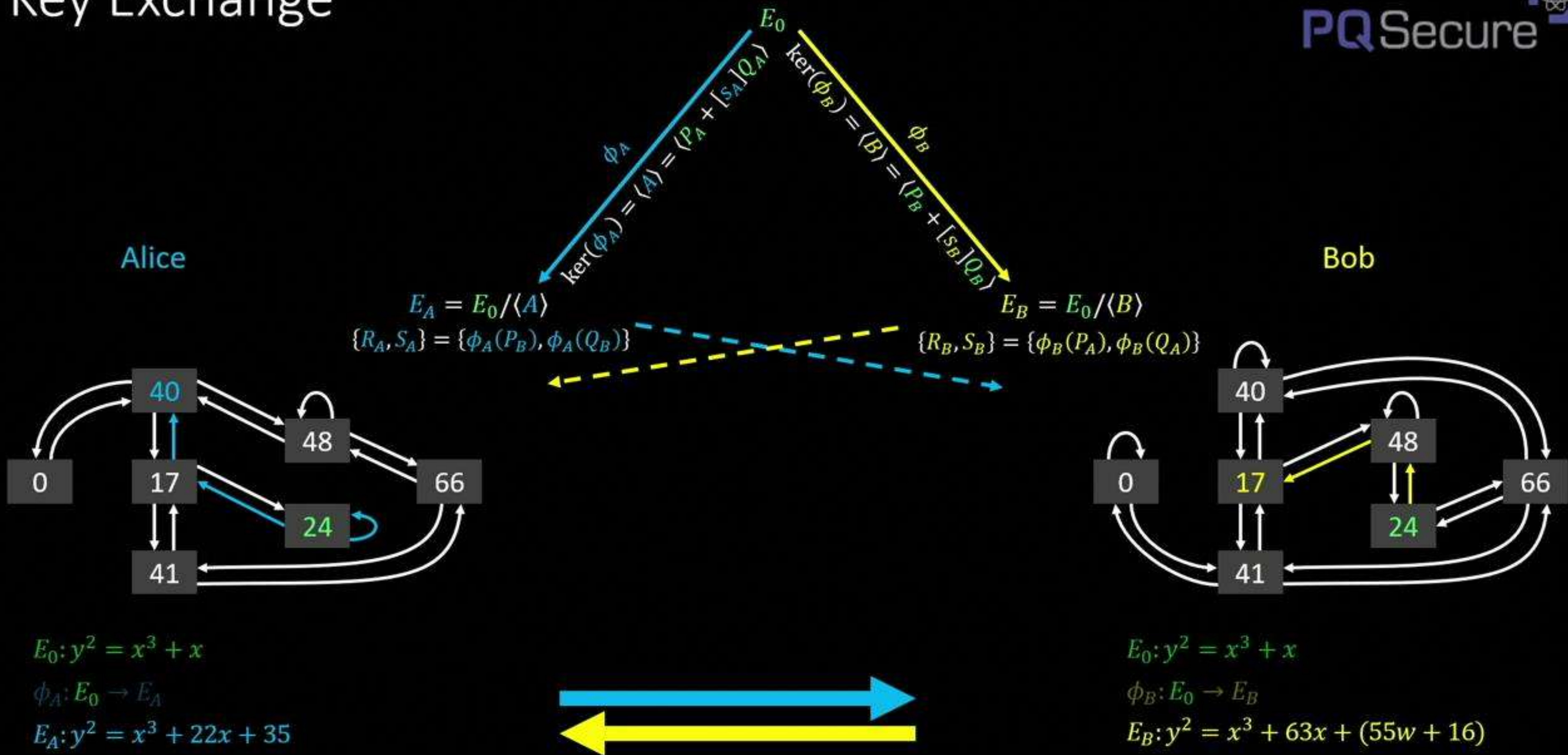
Bob



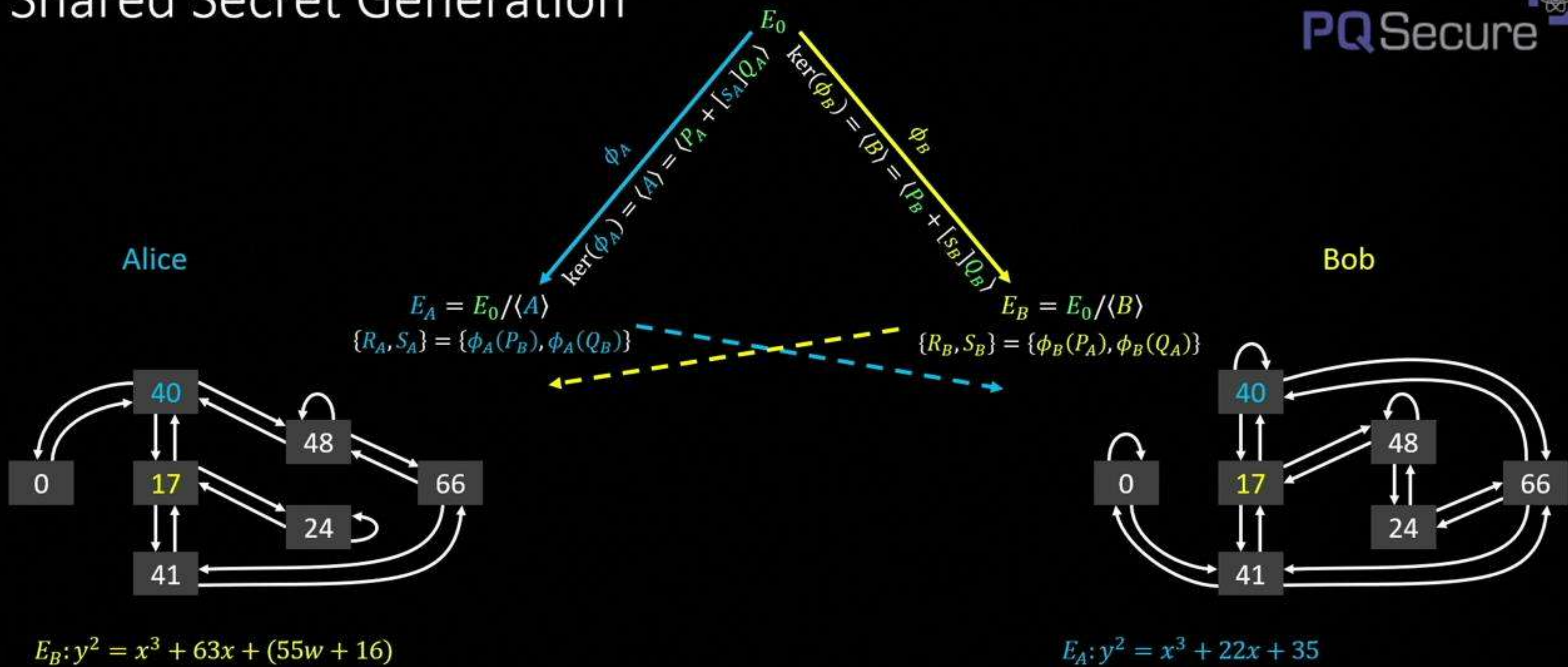
$E_0: y^2 = x^3 + x$   
 $\phi_A: E_0 \rightarrow E_A$   
 $E_A: y^2 = x^3 + 22x + 35$

$E_0: y^2 = x^3 + x$   
 $\phi_B: E_0 \rightarrow E_B$   
 $E_B: y^2 = x^3 + 63x + (55w + 16)$

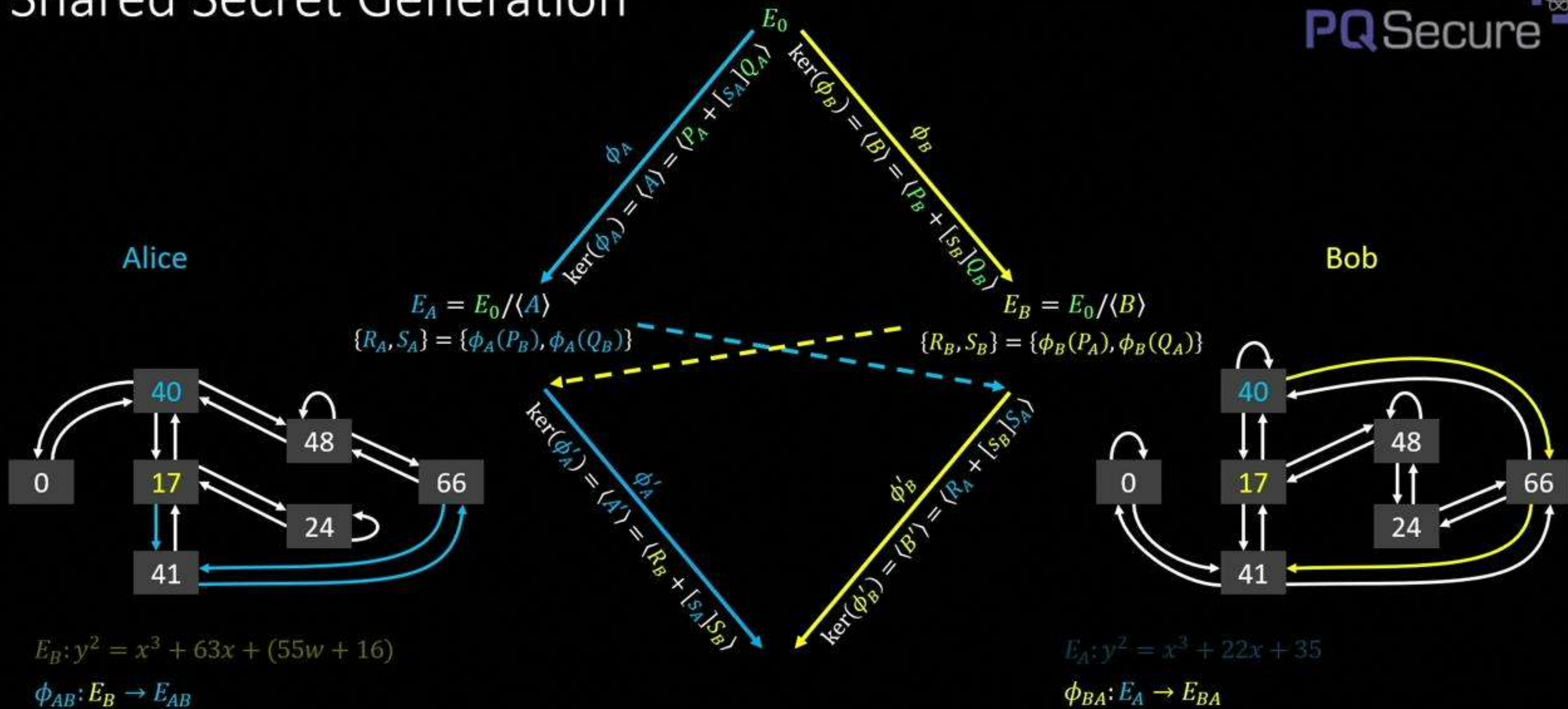
# Key Exchange



# Shared Secret Generation

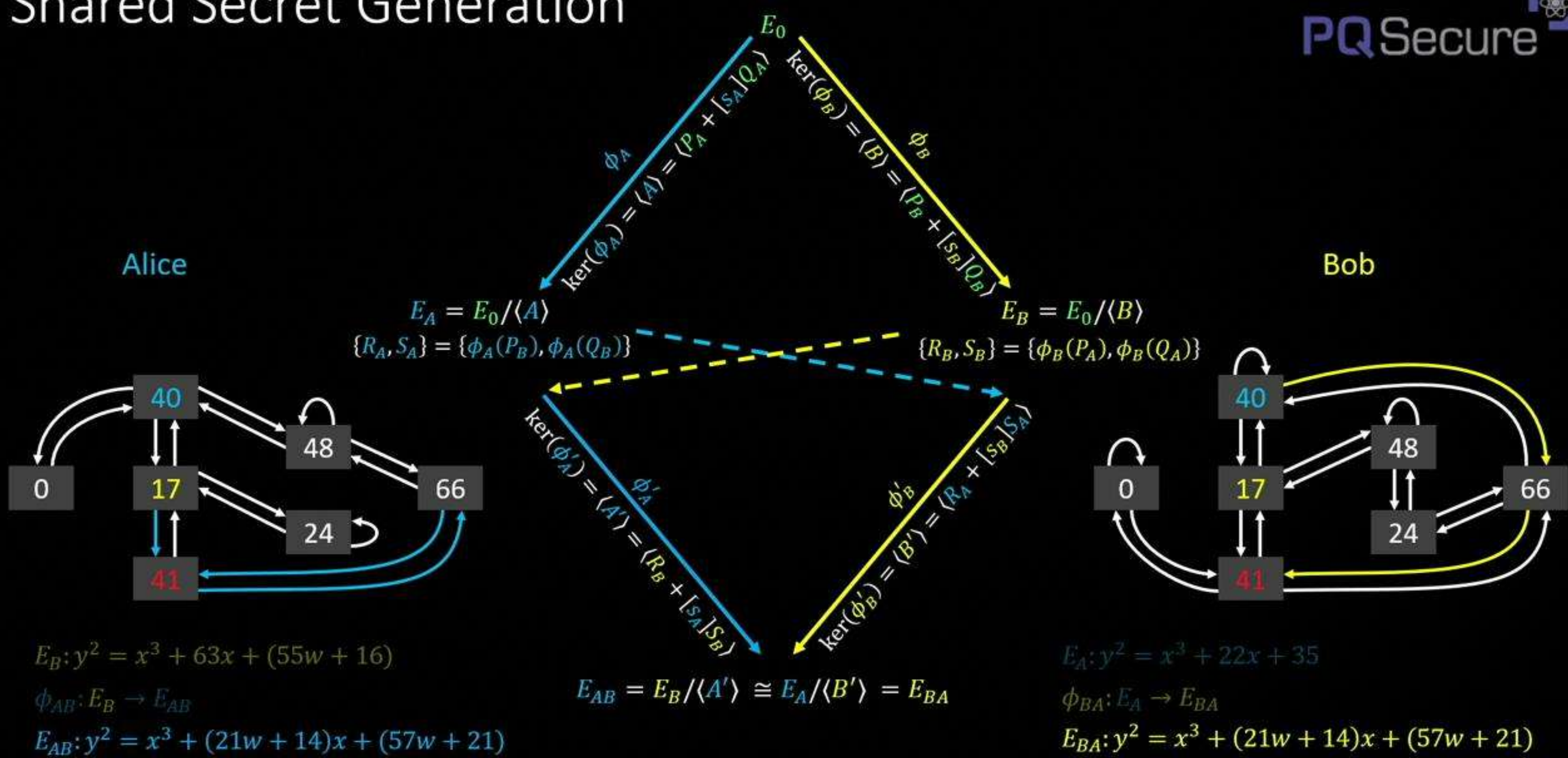


# Shared Secret Generation

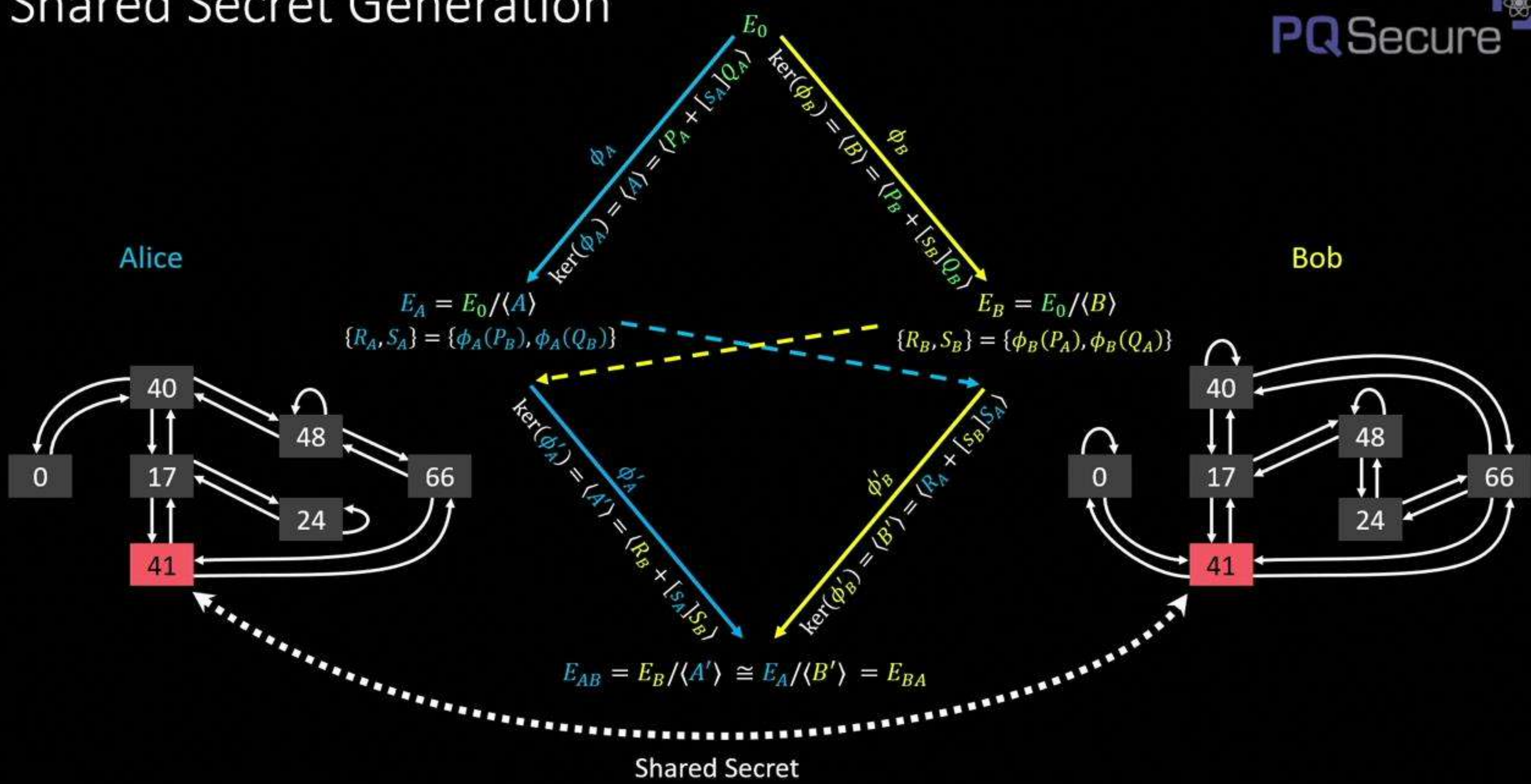




# Shared Secret Generation



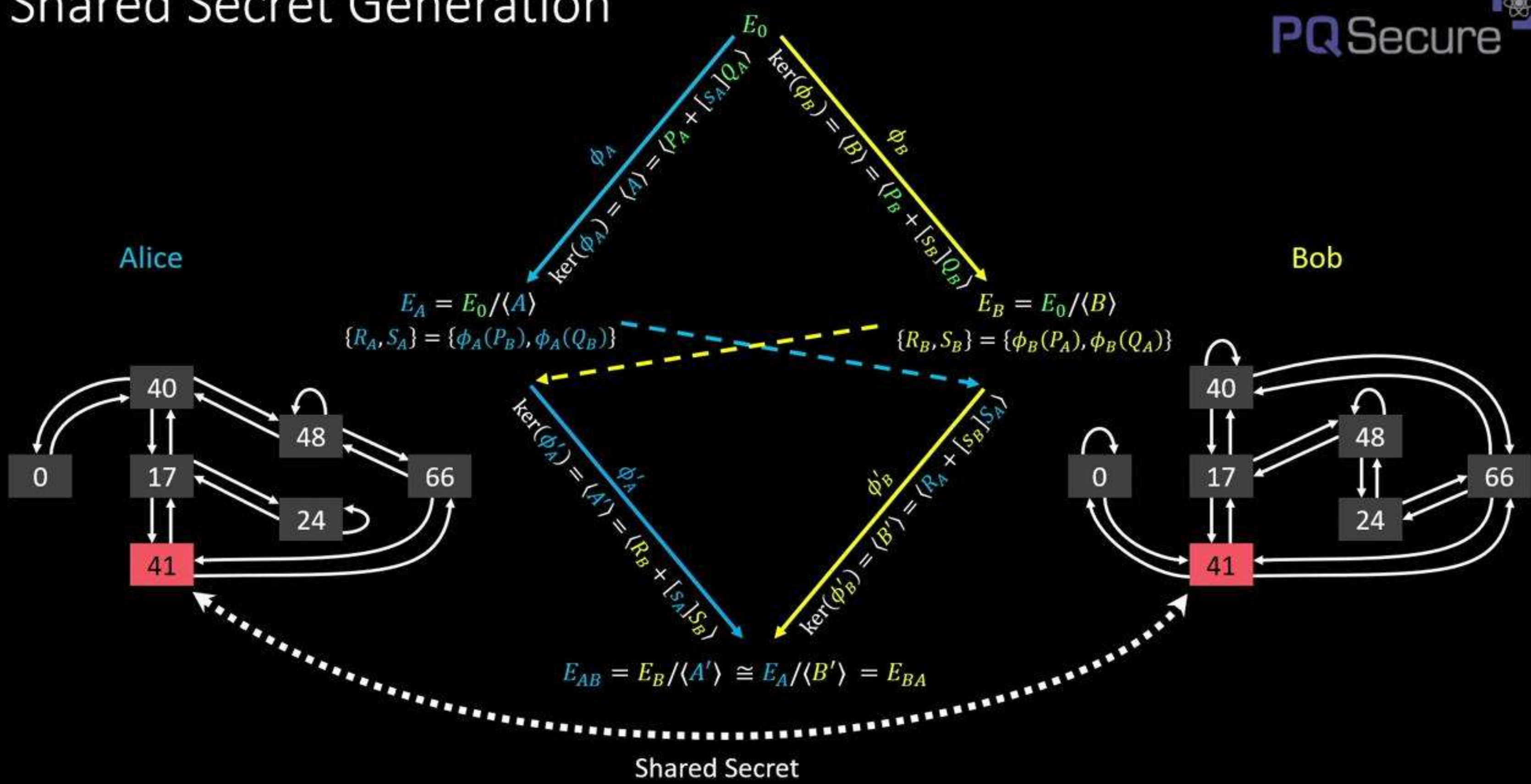
# Shared Secret Generation



# SIKE Key sizes

NIST Level	Prime size (bits)	Prime	Public key size (bytes)	Compressed PK size (bytes)
1	434	$2^{216}3^{137} - 1$	330	196
2	503	$2^{250}3^{159} - 1$	378	224
3	610	$2^{305}3^{192} - 1$	462	273
5	751	$2^{372}3^{239} - 1$	564	331

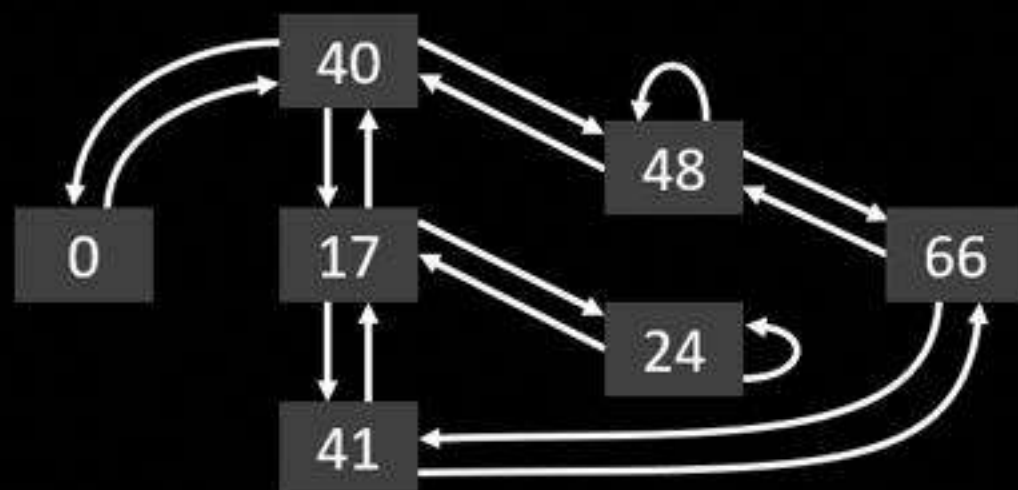
# Shared Secret Generation



# SIKE Key sizes

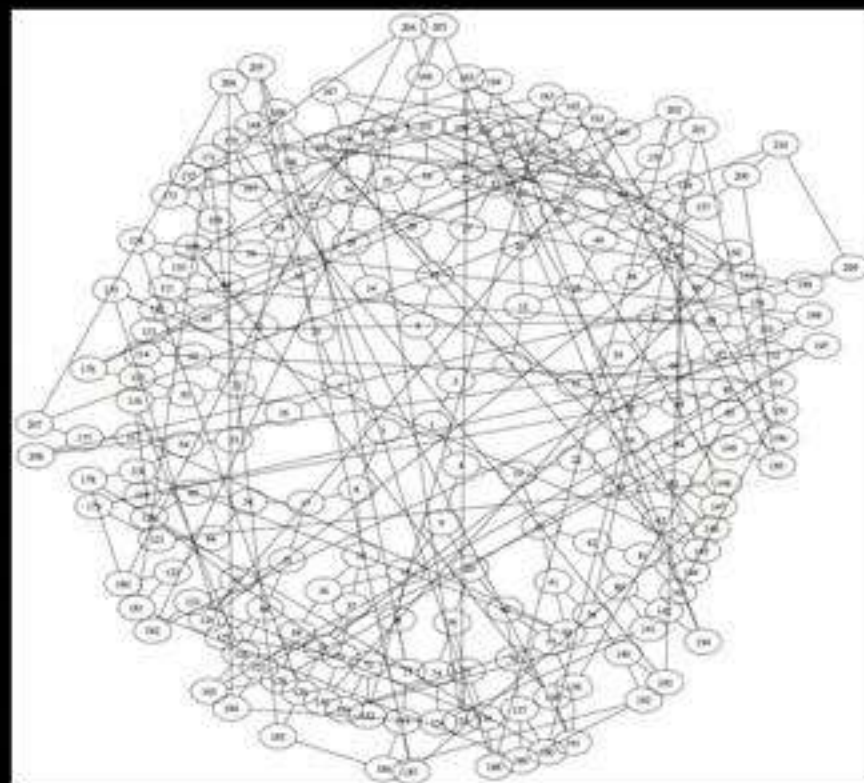
NIST Level	Prime size (bits)	Prime	Public key size (bytes)	Compressed PK size (bytes)
1	434	$2^{216}3^{137} - 1$	330	196
2	503	$2^{250}3^{159} - 1$	378	224
3	610	$2^{305}3^{192} - 1$	462	273
5	751	$2^{372}3^{239} - 1$	564	331

# Isogeny Graphs



$$p = 71 = 2^3 \cdot 3^2 - 1$$

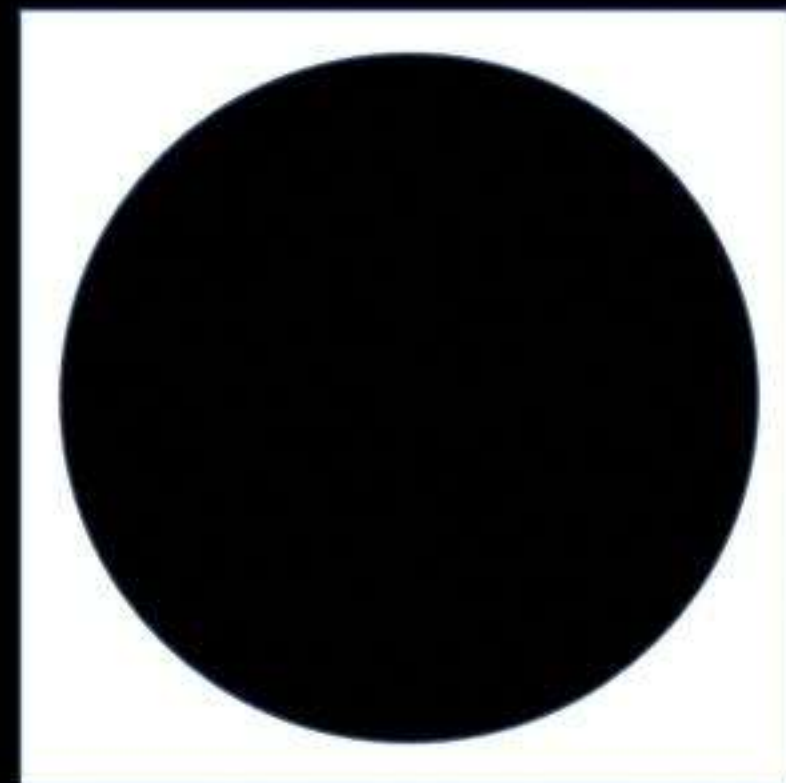
nodes = 7



$$p = 2521$$

nodes = 210

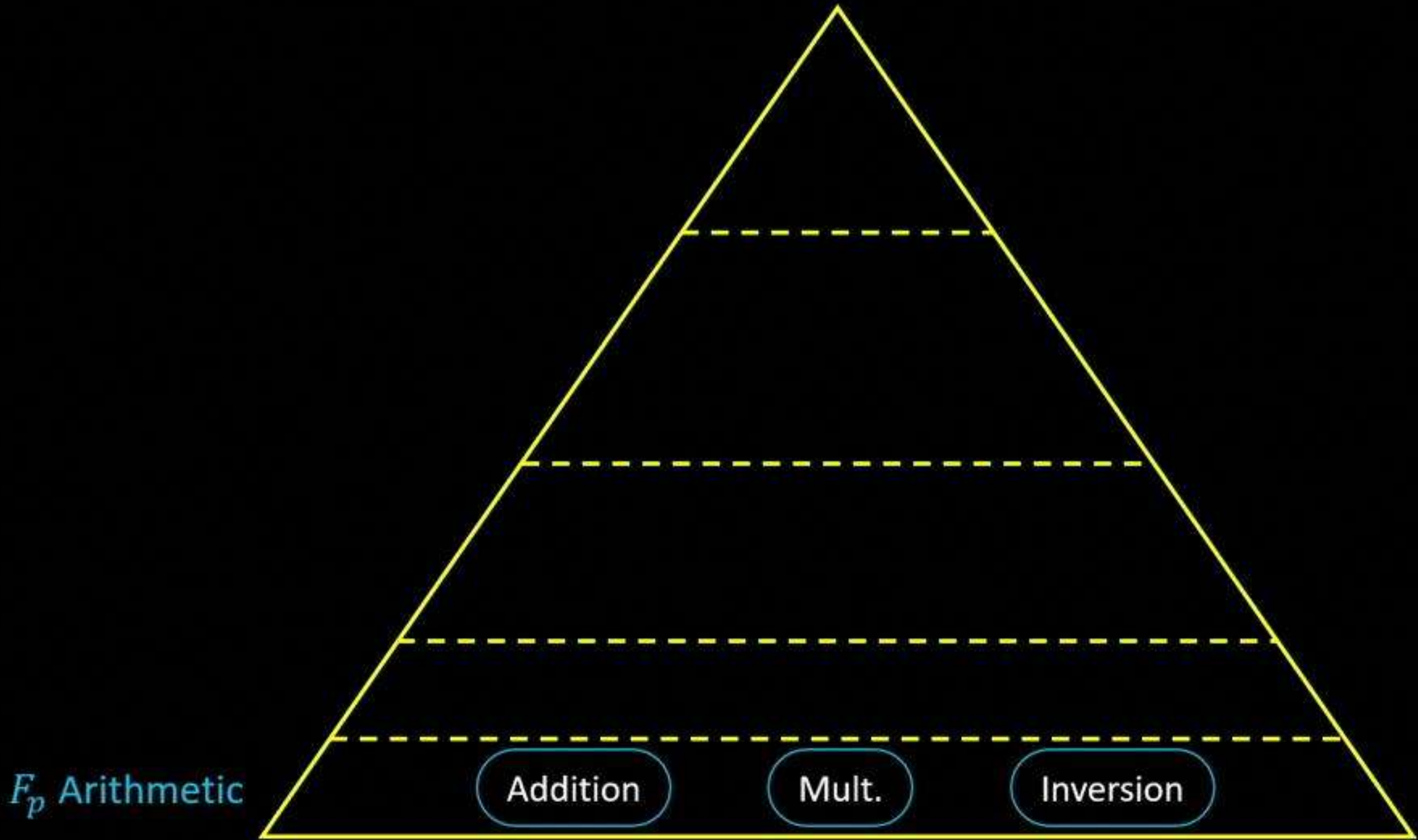
[CGL06]



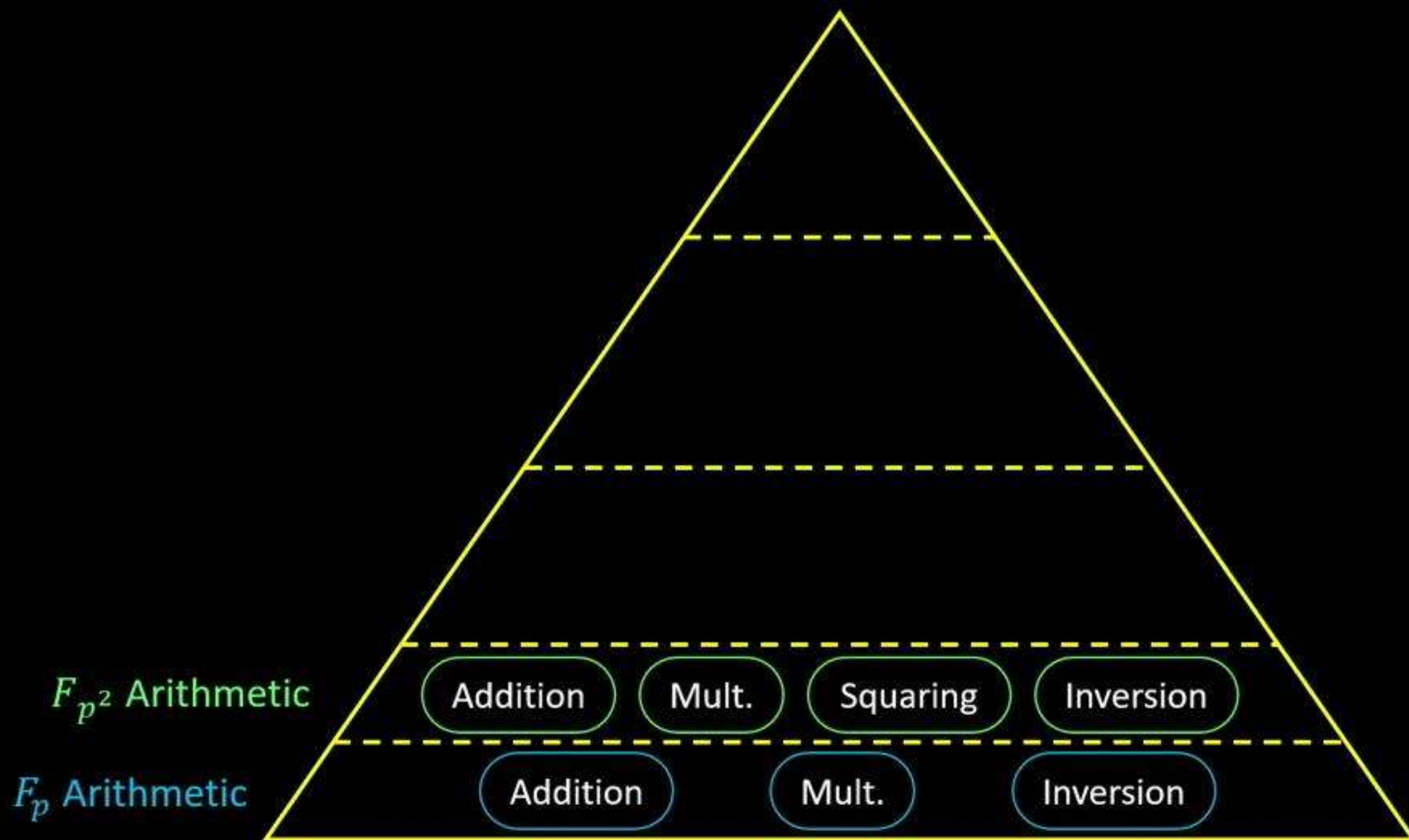
$$\text{SIKEp434} \approx 2^{216} \cdot 3^{137} - 1$$

nodes  $\approx 2^{430}$

# SIDH Computations

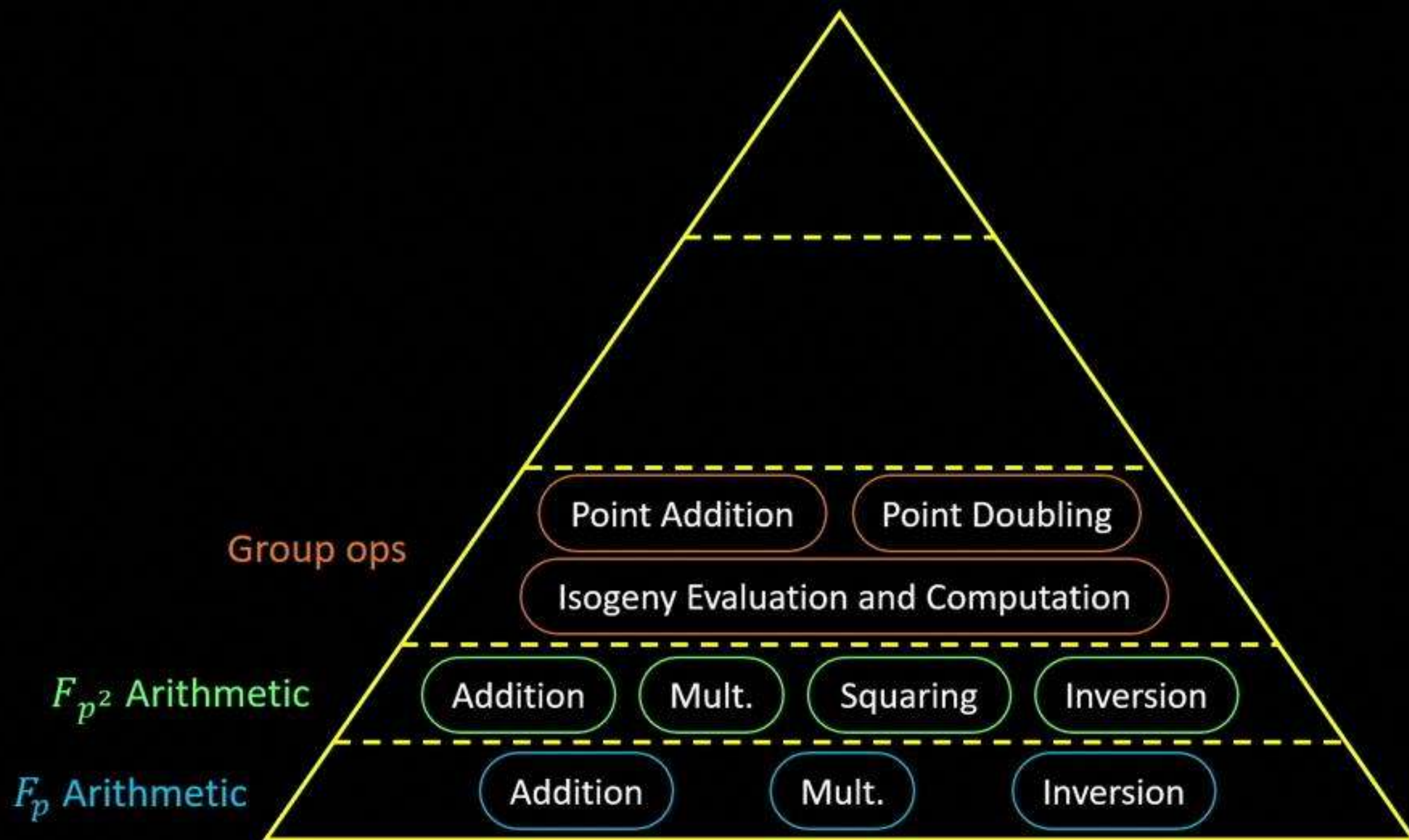


# SIDH Computations

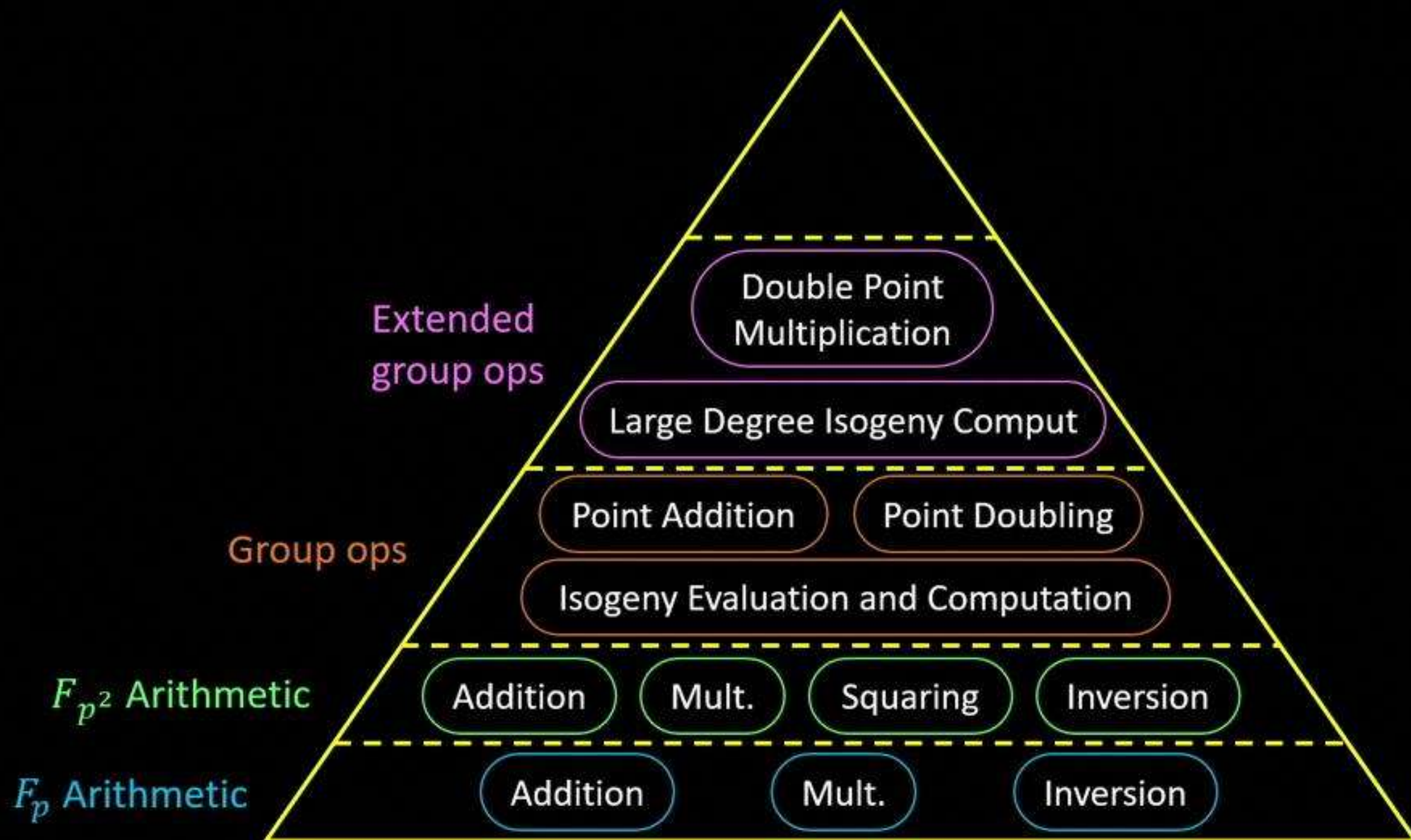




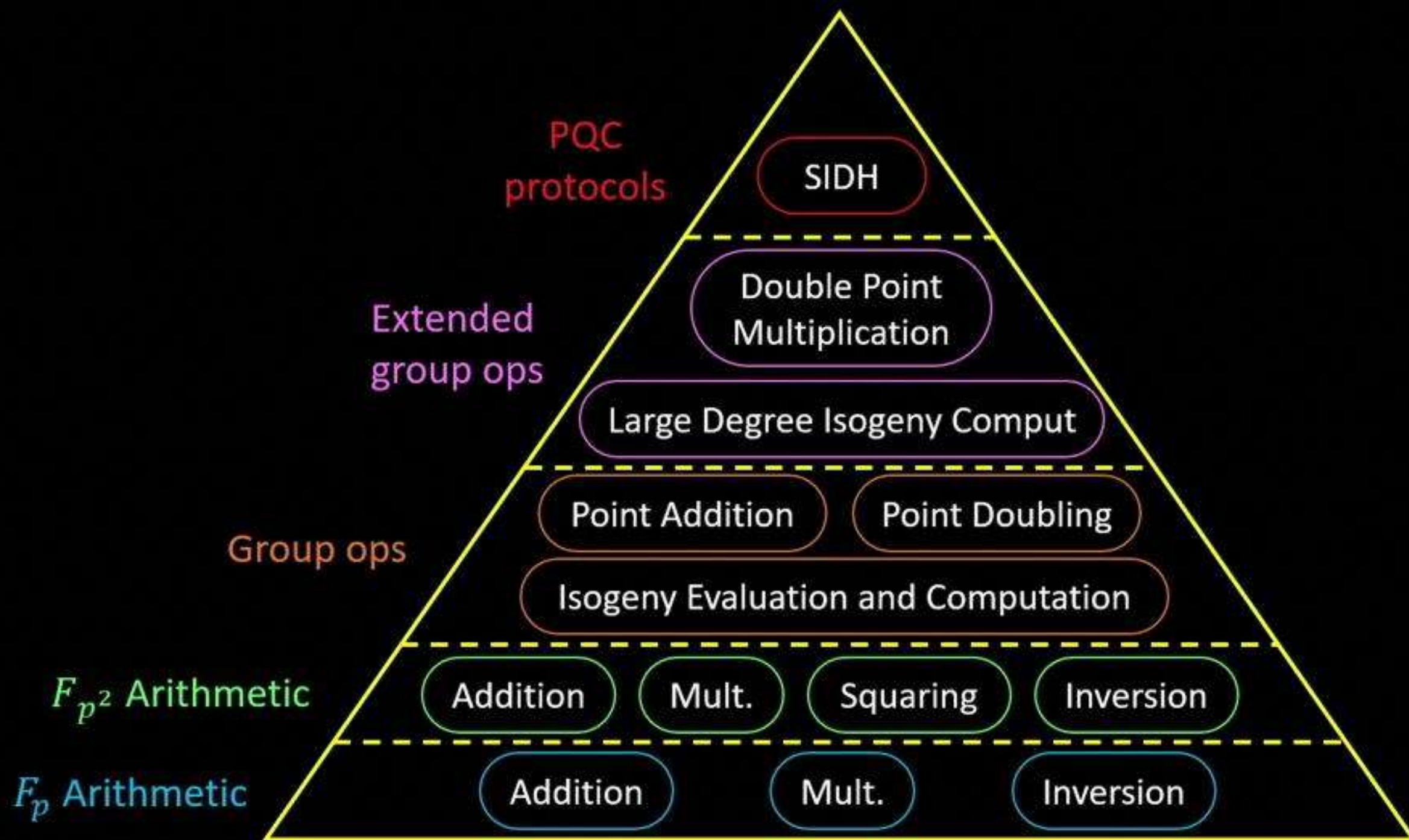
# SIDH Computations



# SIDH Computations



# SIDH Computations

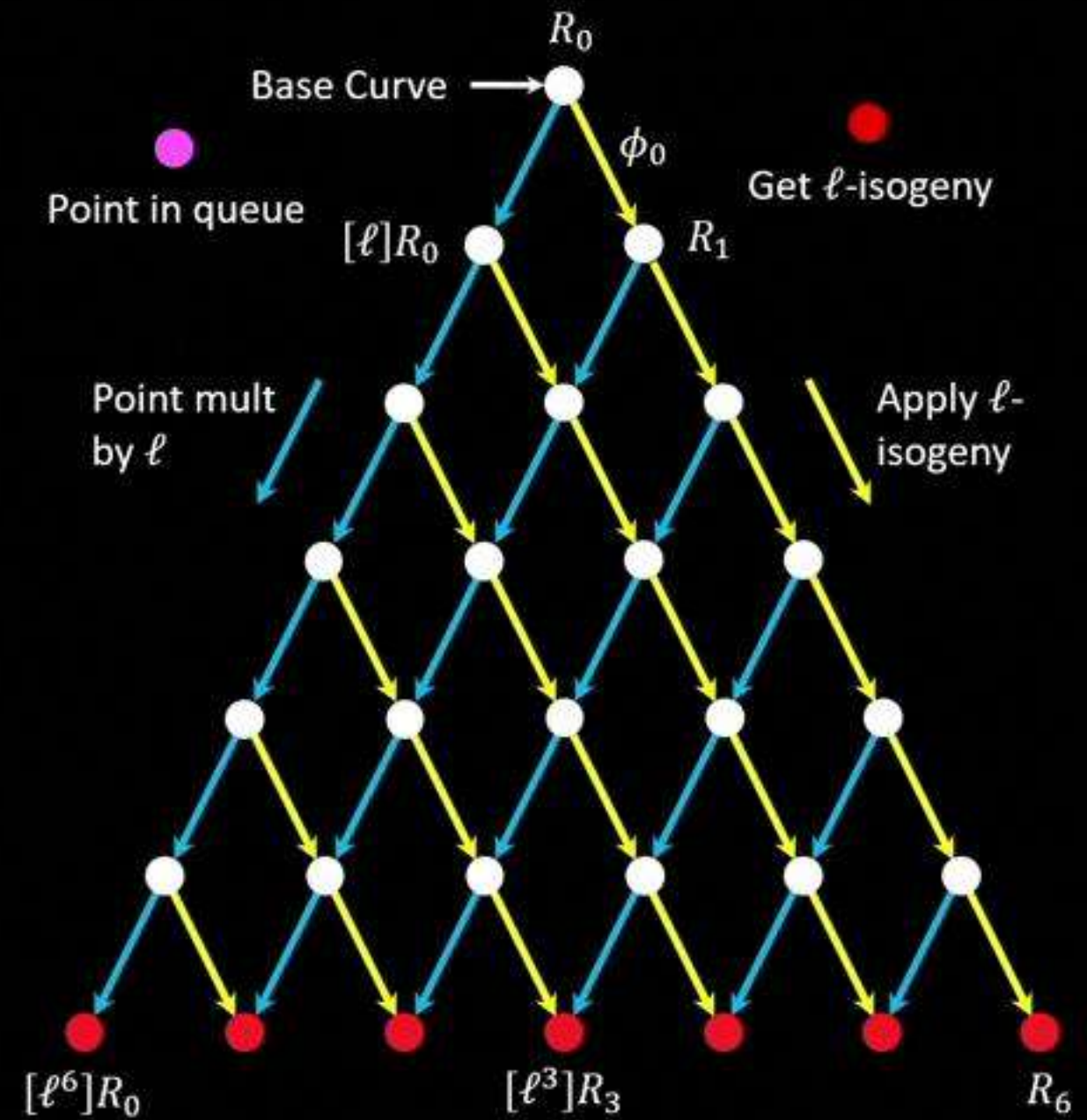


# Large degree isogeny computations

e.g.,  $\phi: E = E_0 / \langle R_0 \rangle$ ,  $\text{ord}(R_0) = \ell^7$

- Get isogeny Kernel  $[\ell^{e-i-1}]R_i$
- Compute Isogenies  $\phi_i := E_i / \langle [\ell^{e-i-1}]R_i \rangle$
- Compute  $E_{i+1} = \phi_i(E_i)$
- Push points to new curve  $R_{i+1} = \phi_i(R_i)$

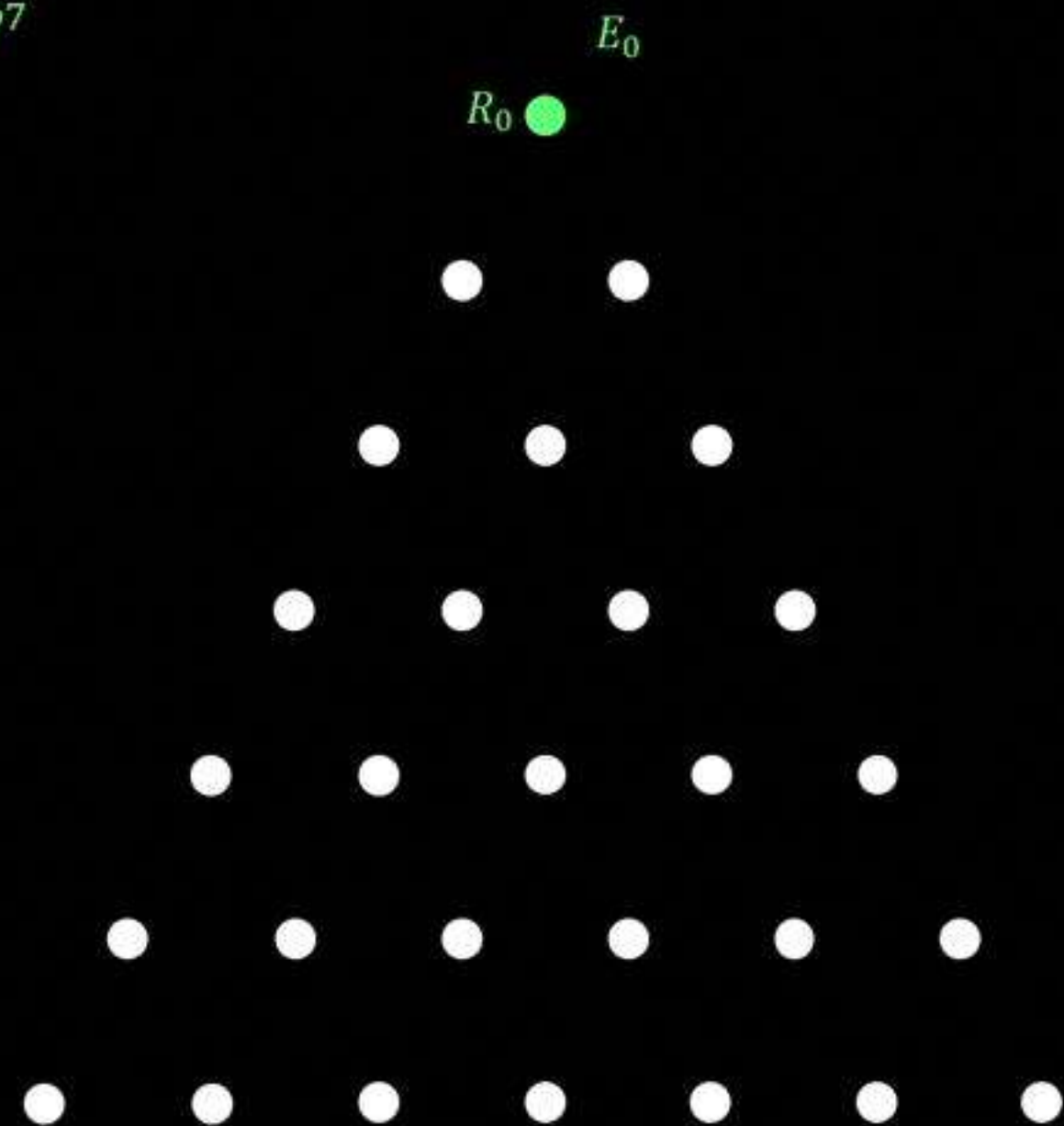
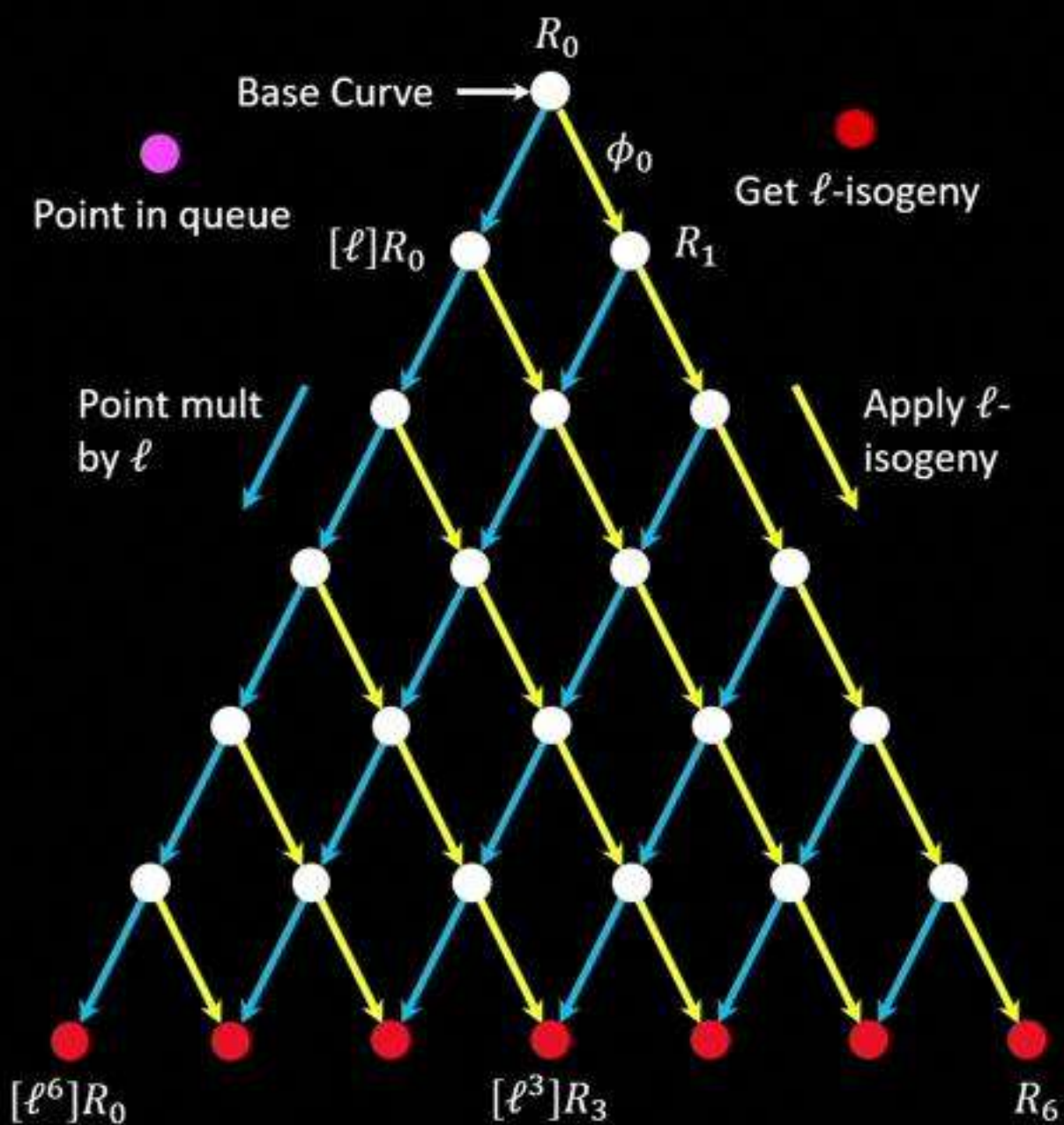
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$



# Large degree isogeny computations

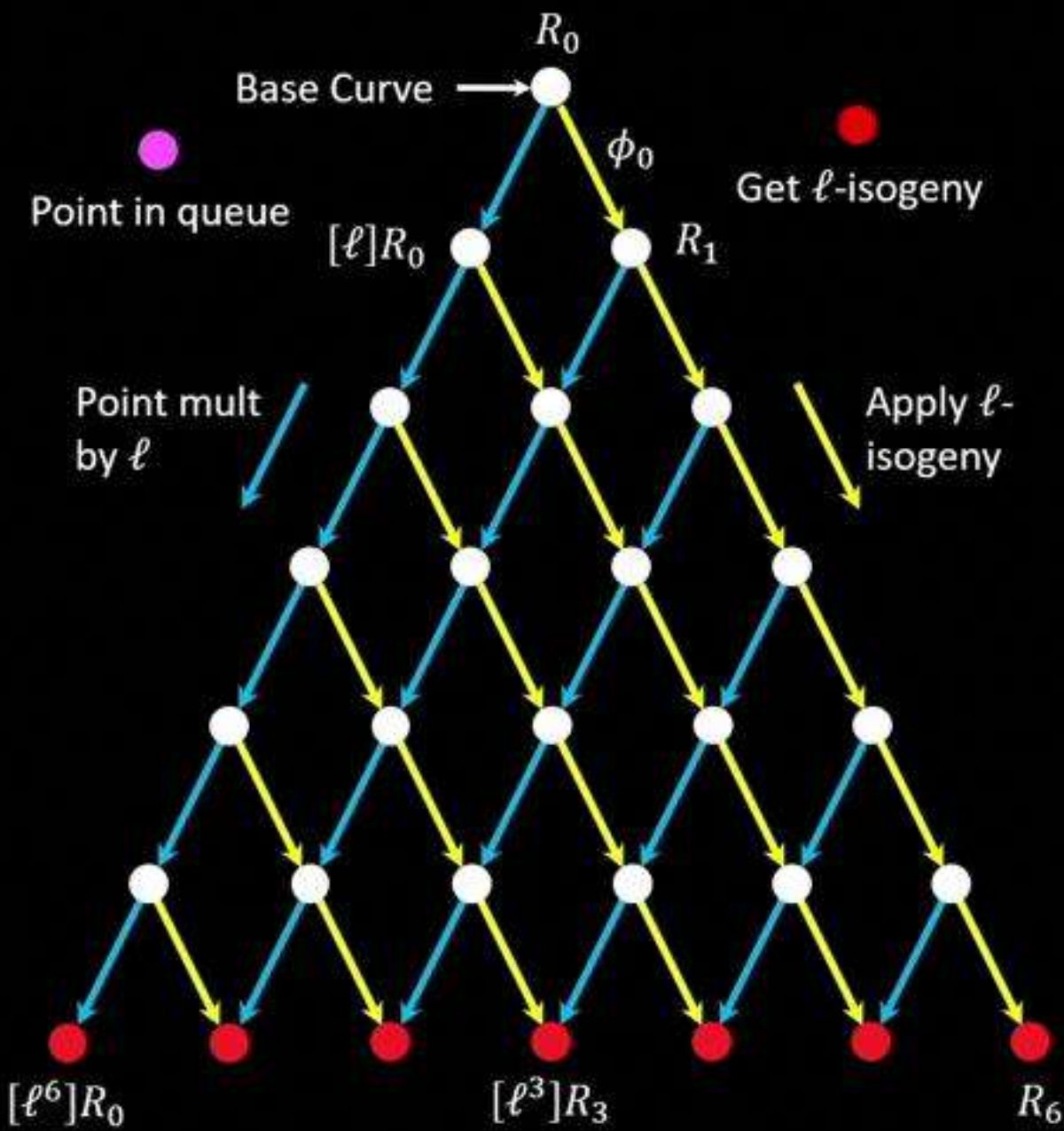
$$e = 7$$

$$\text{e.g., } \phi: E = E_0 / \langle R_0 \rangle, \text{ord}(R_0) = \ell^7$$

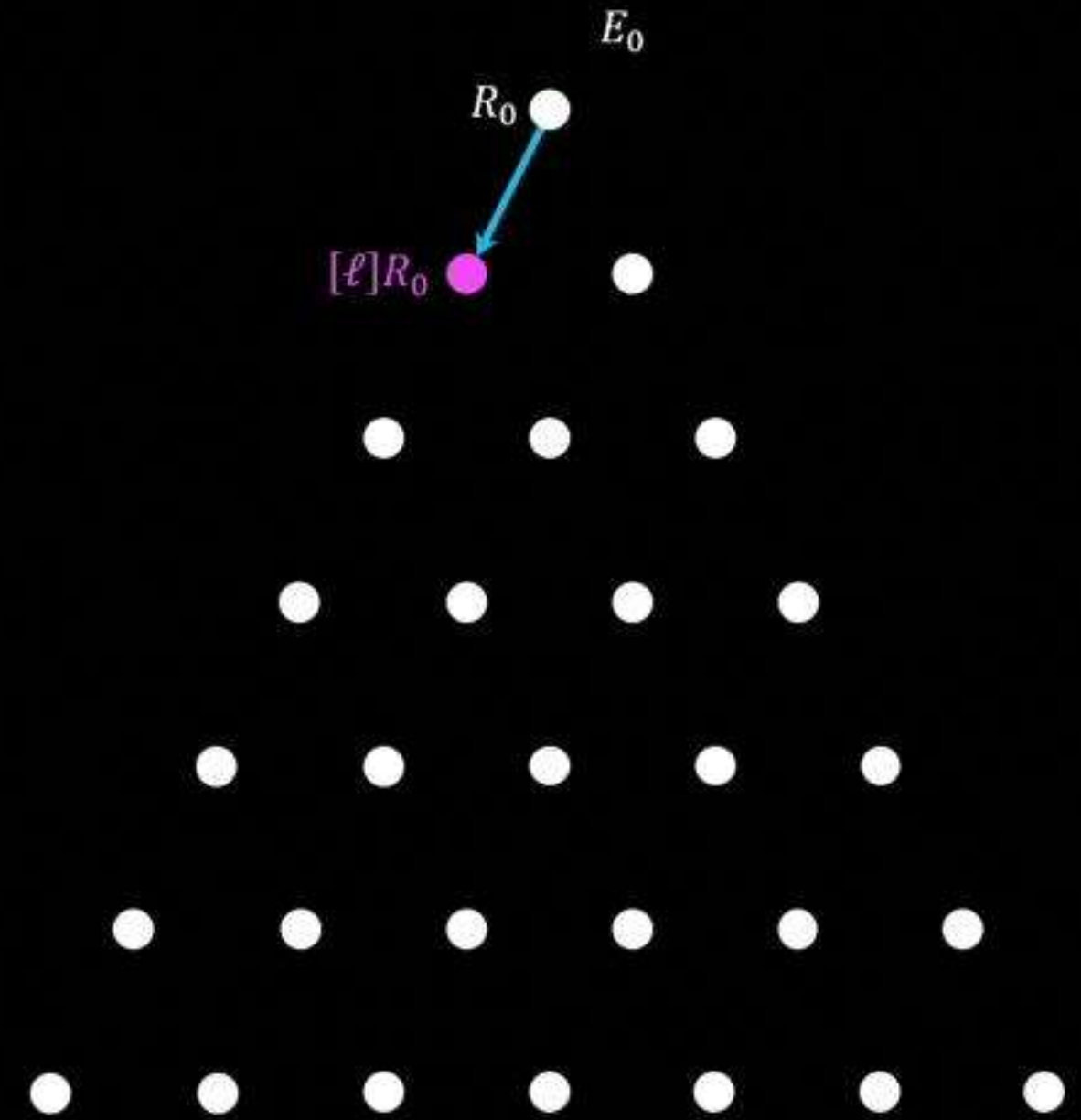


# Large degree isogeny computations

$e = 7$



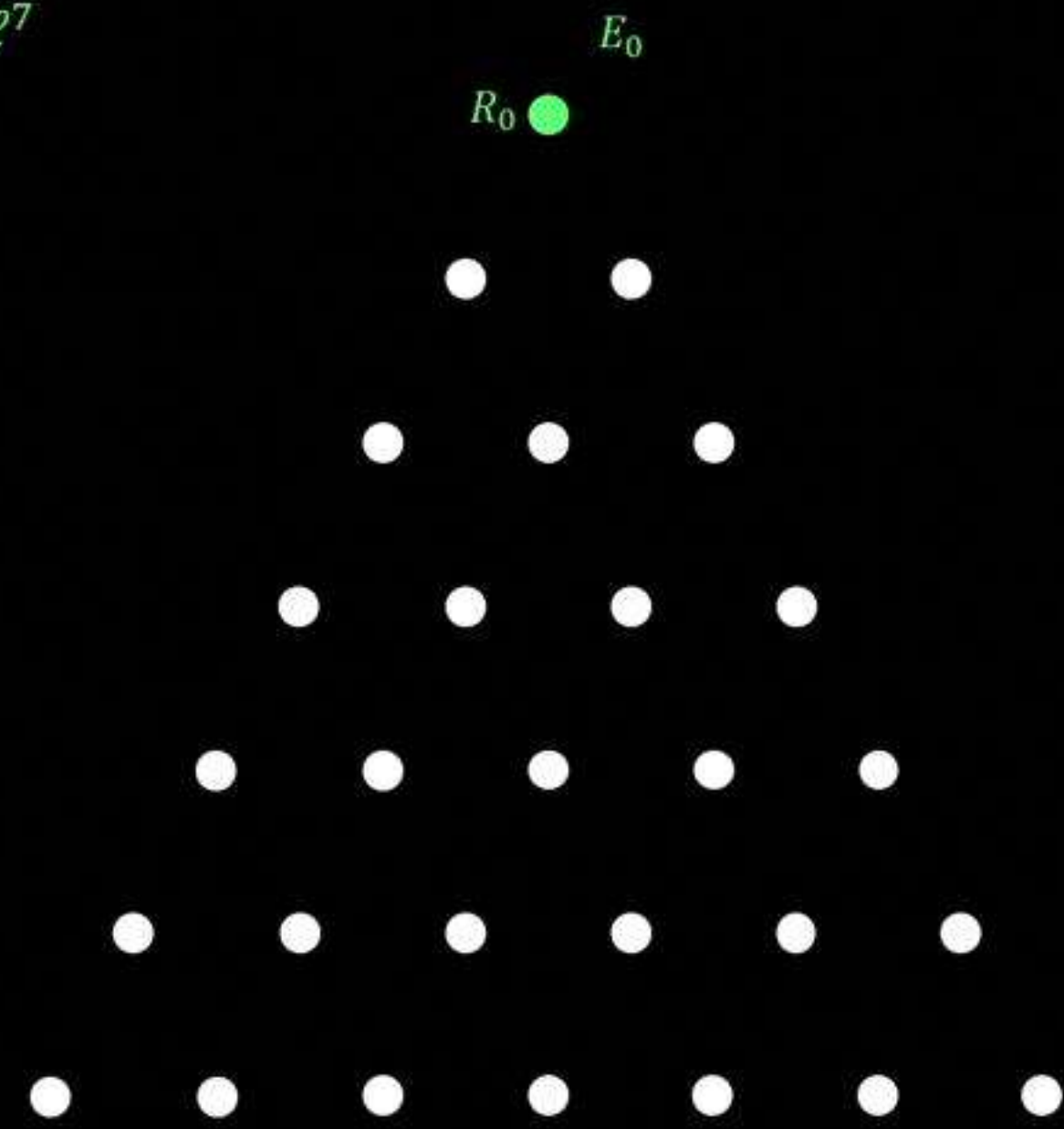
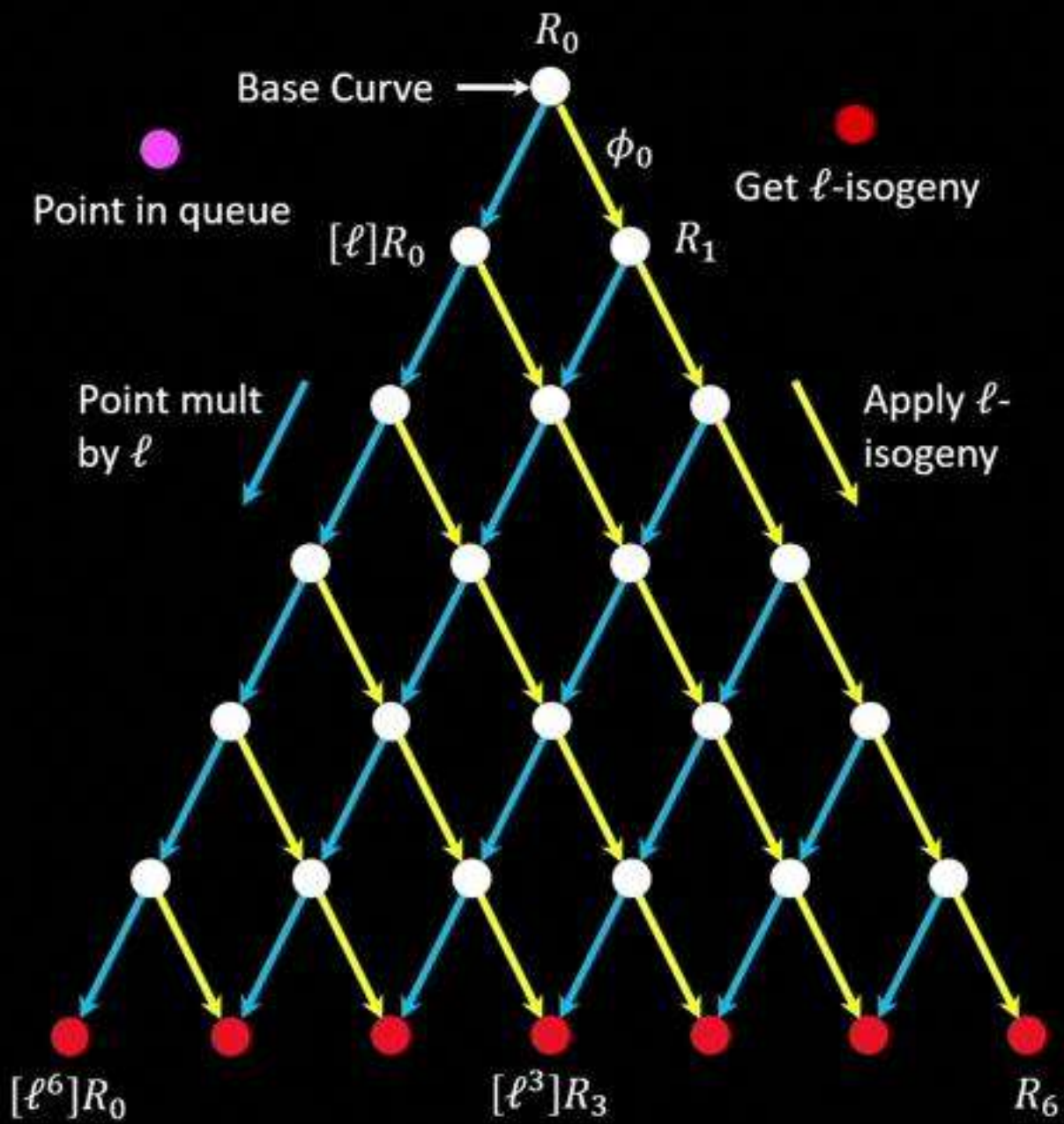
Order of  $[l]R_0$  is  $l^6$



# Large degree isogeny computations

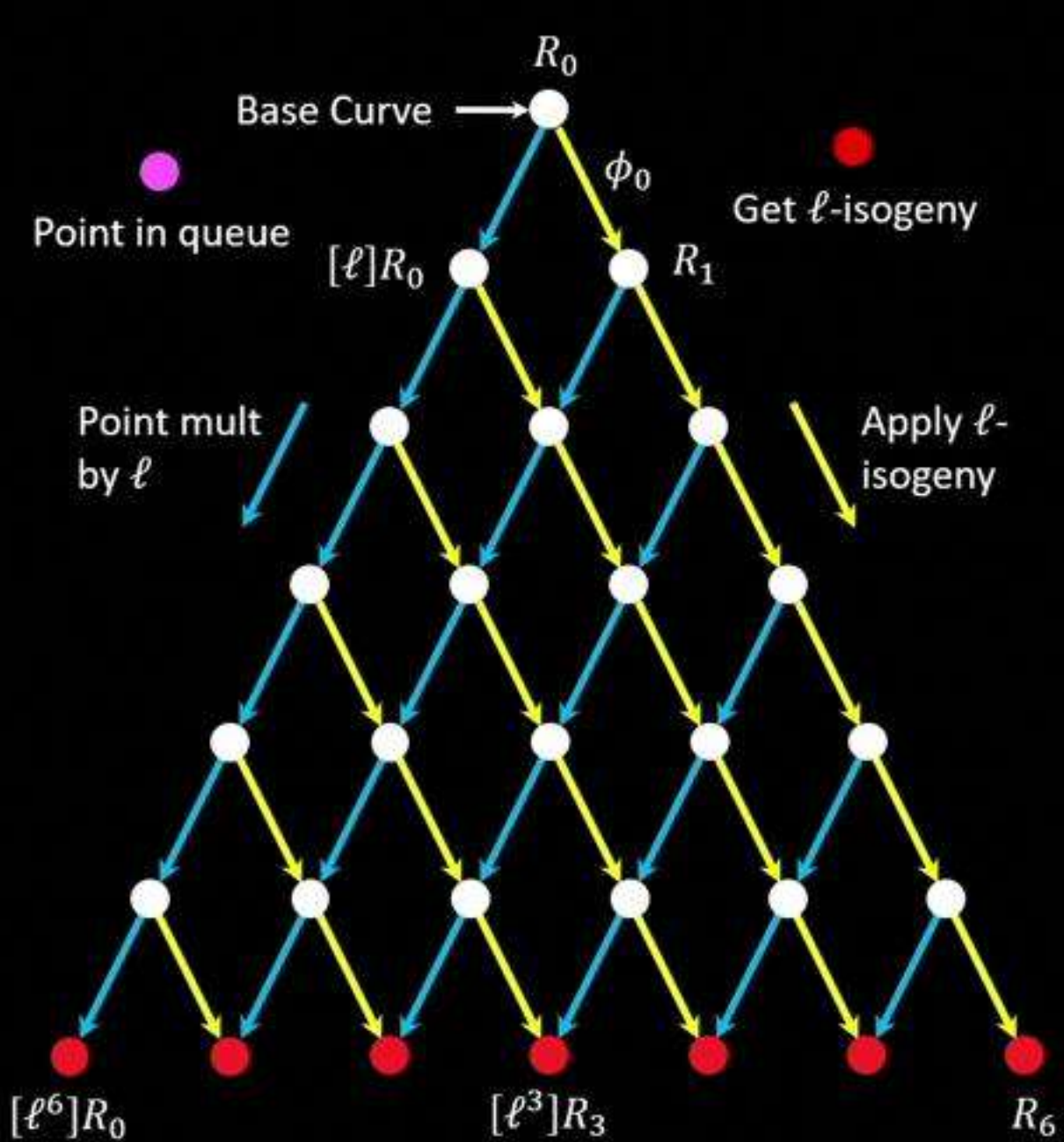
$$e = 7$$

e.g.,  $\phi: E = E_0 / \langle R_0 \rangle$ ,  $\text{ord}(R_0) = \ell^7$

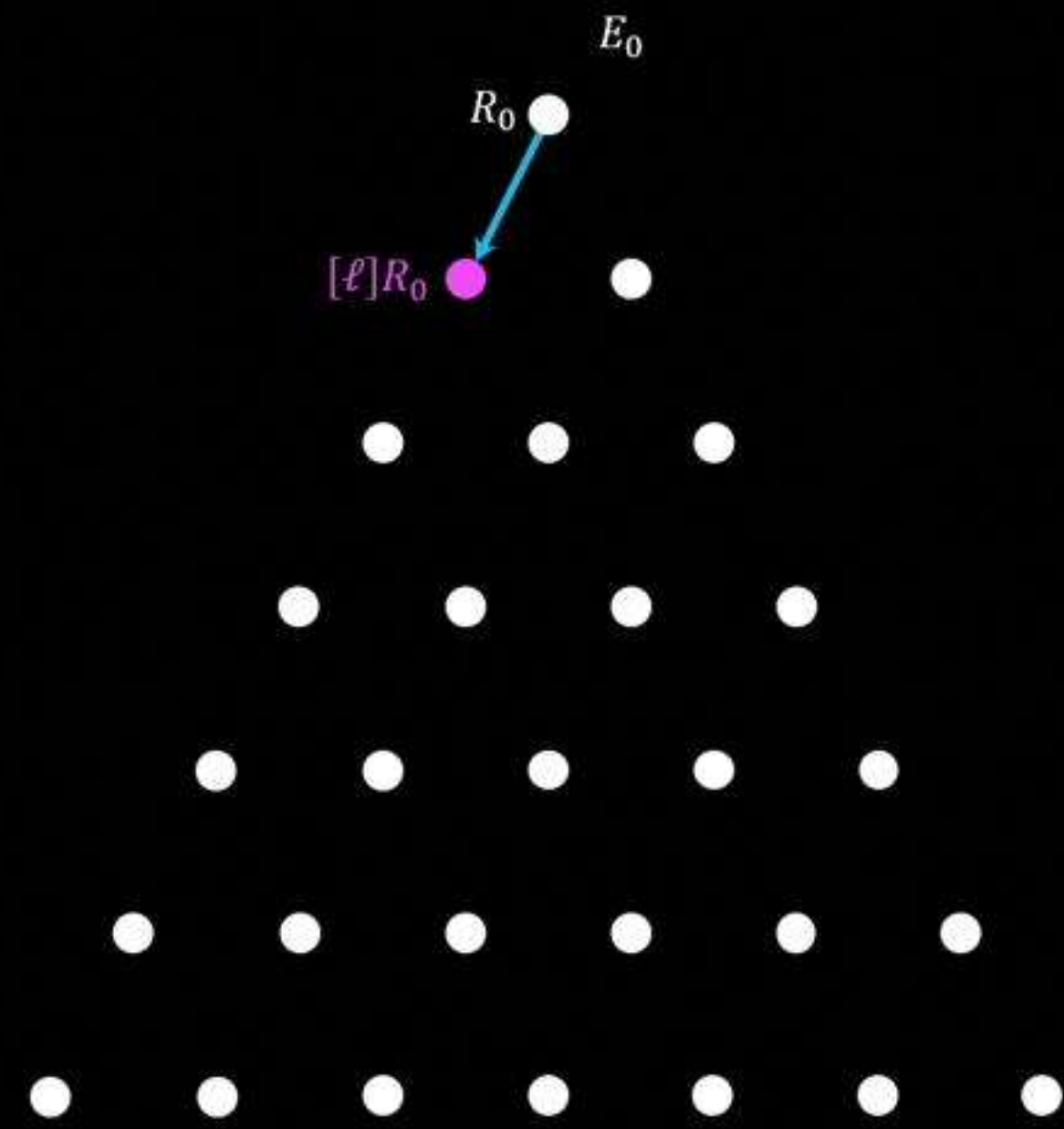


# Large degree isogeny computations

$e = 7$



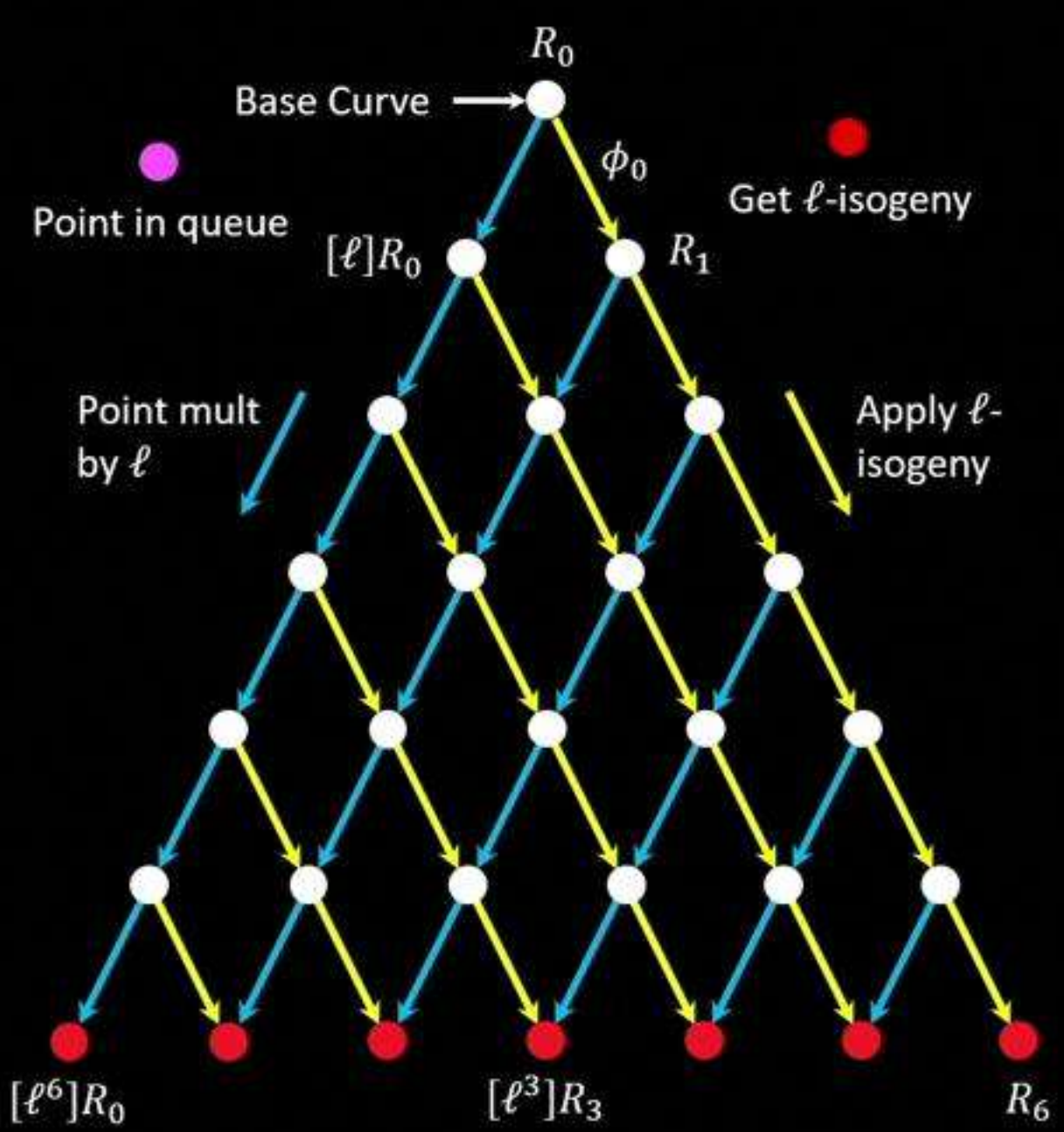
Order of  $[\ell]R_0$  is  $\ell^6$



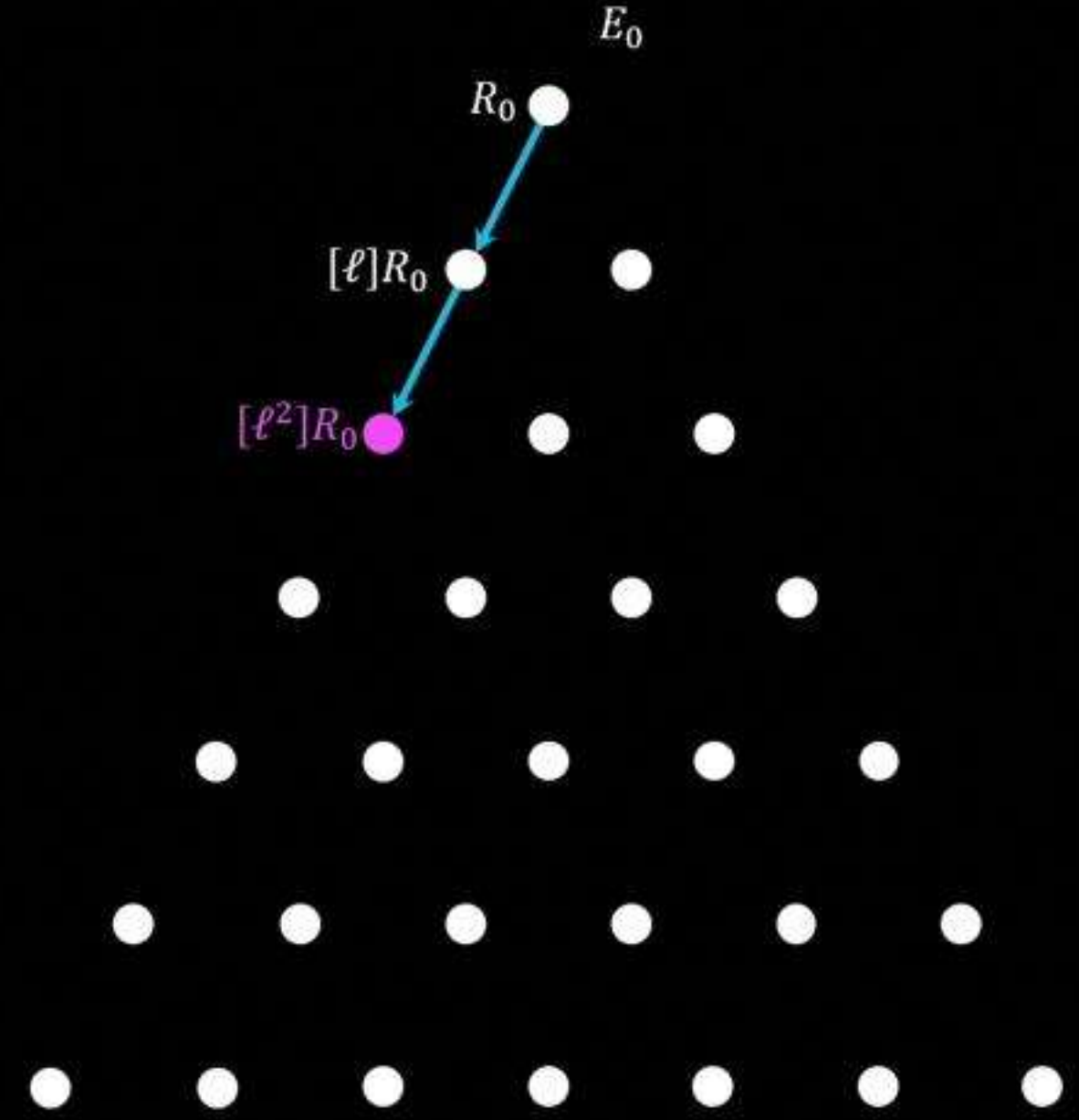


# Large degree isogeny computations

$e = 7$

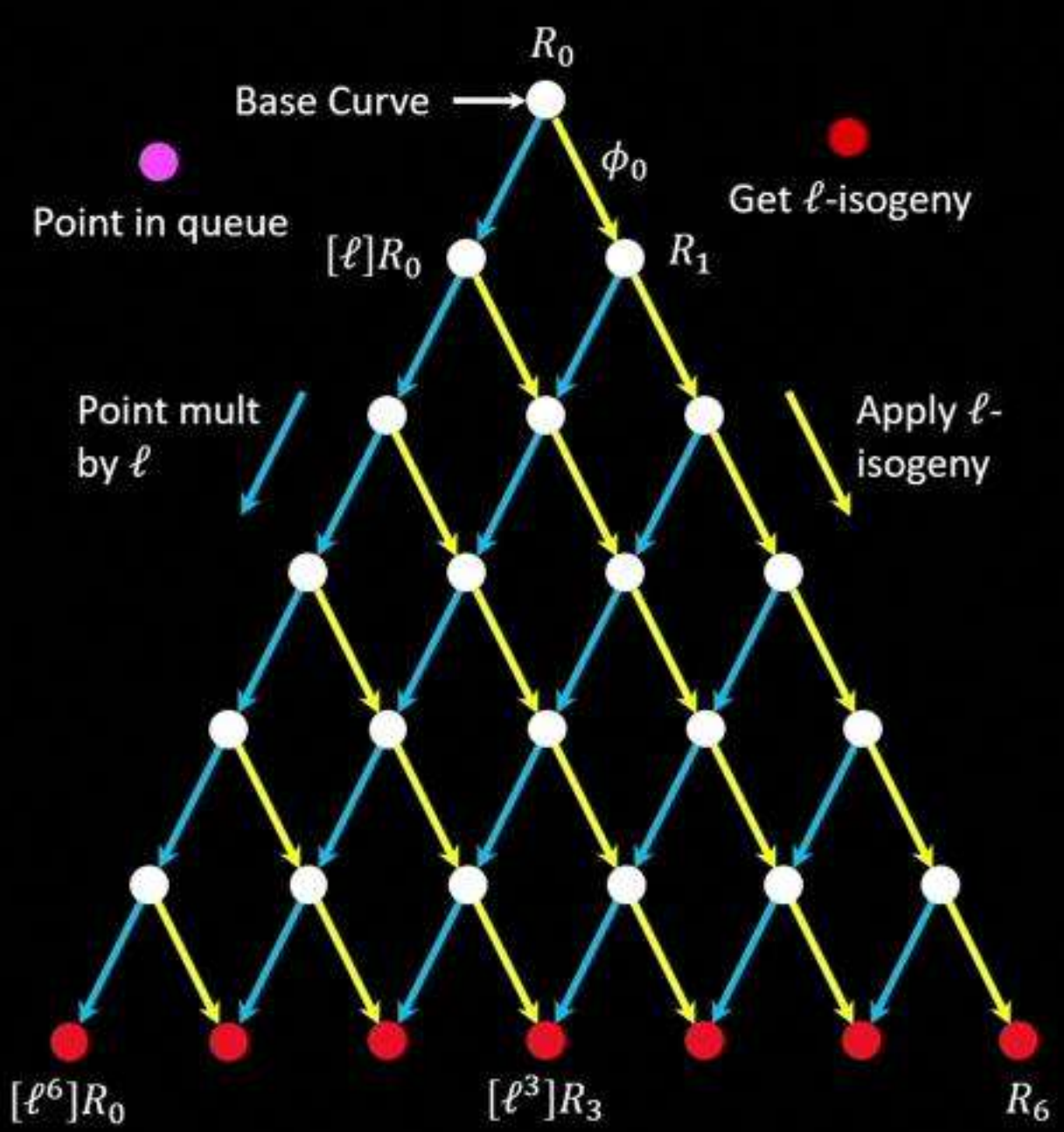


Order of  $[l^2]R_0$  is  $\ell^5$

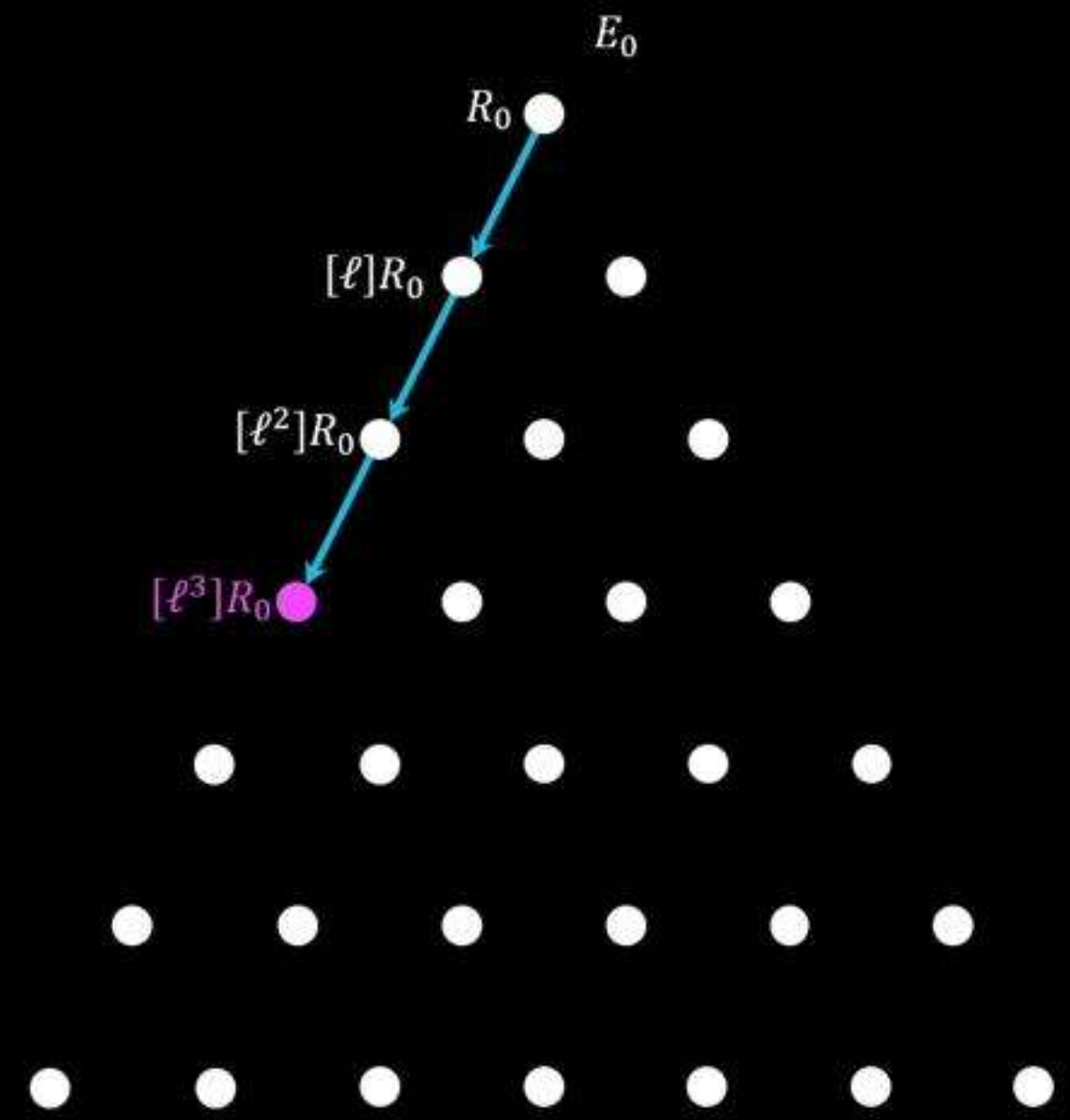


# Large degree isogeny computations

$e = 7$

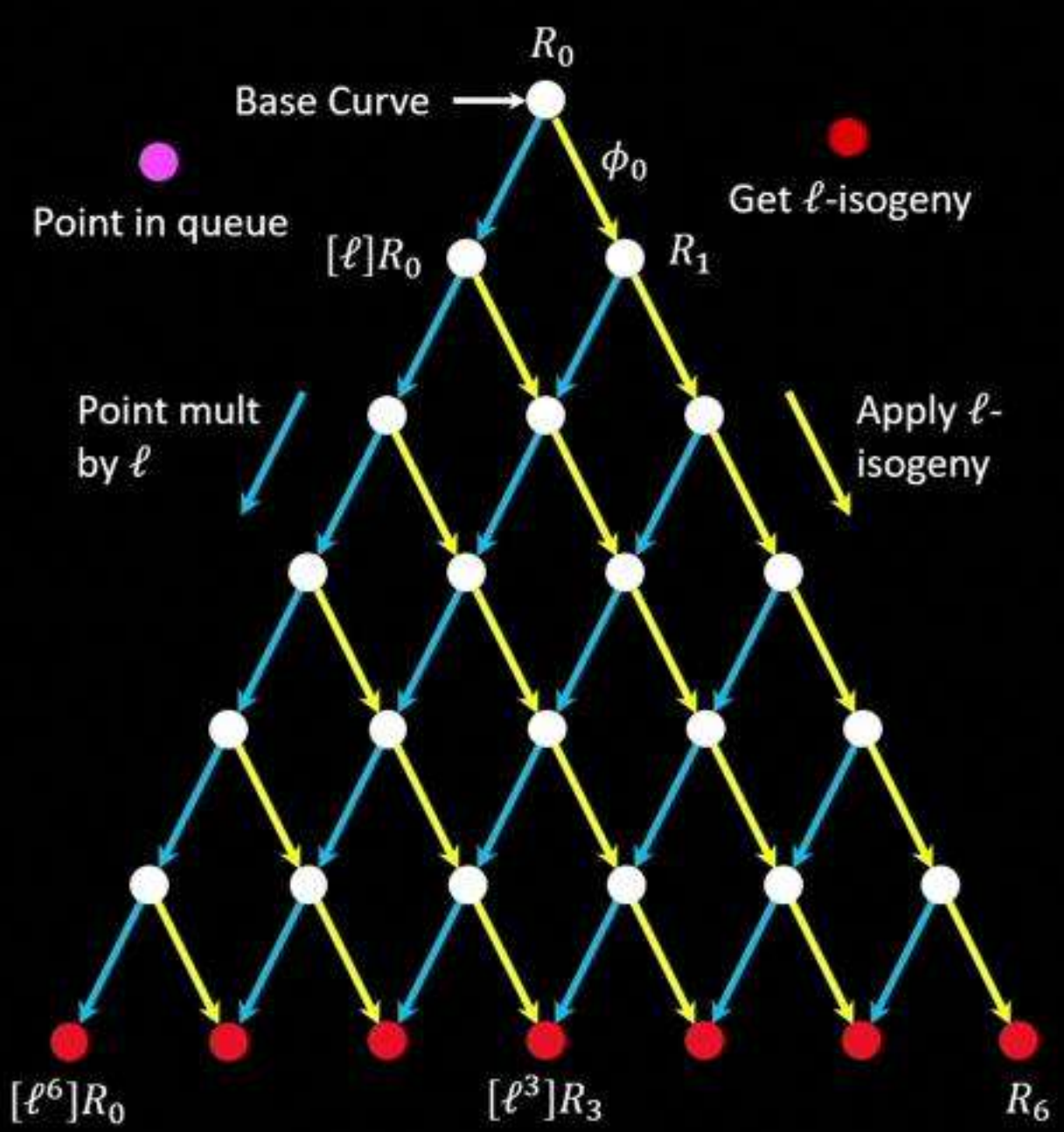


Order of  $[l^3]R_0$  is  $l^4$

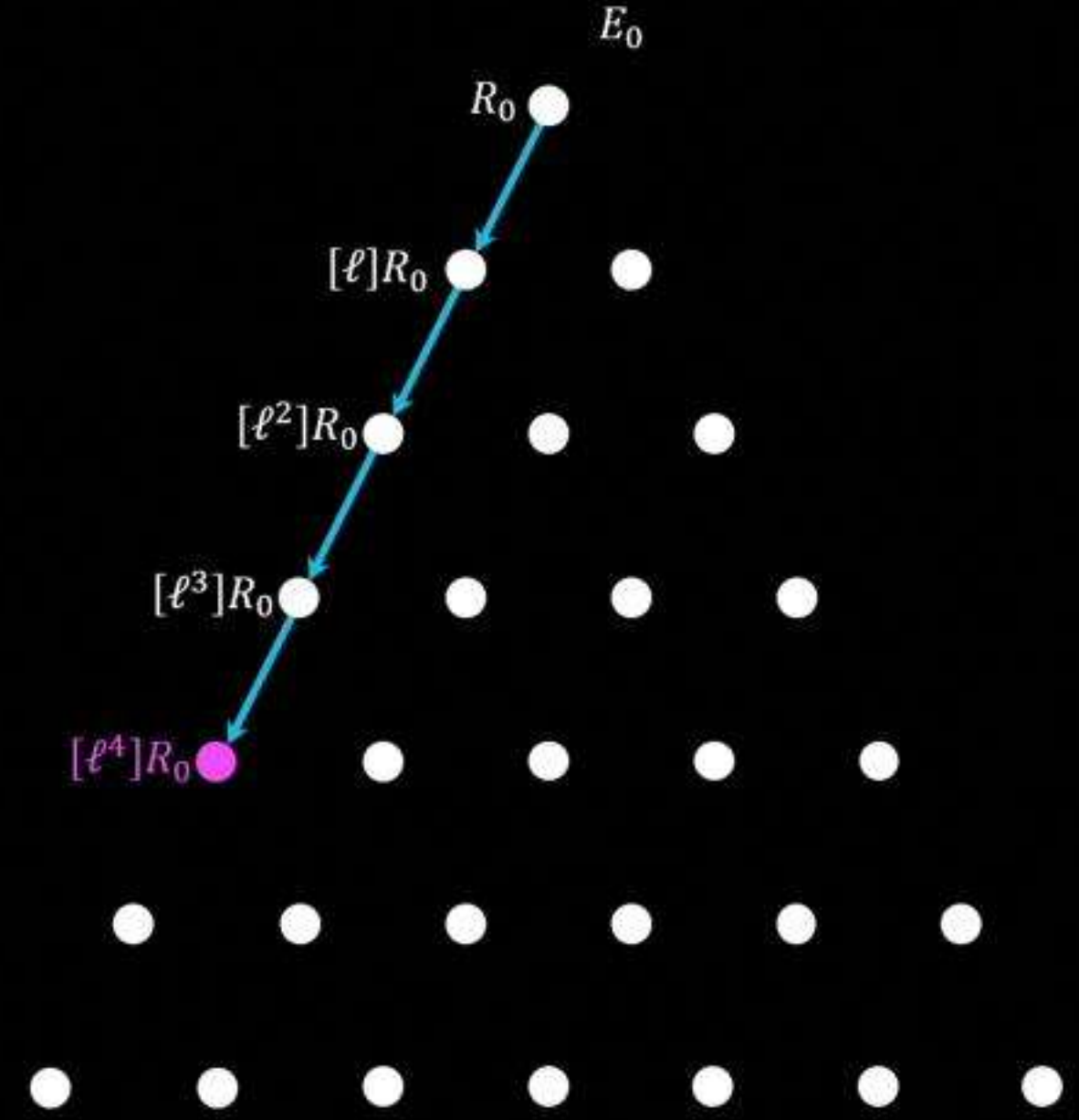


# Large degree isogeny computations

$e = 7$

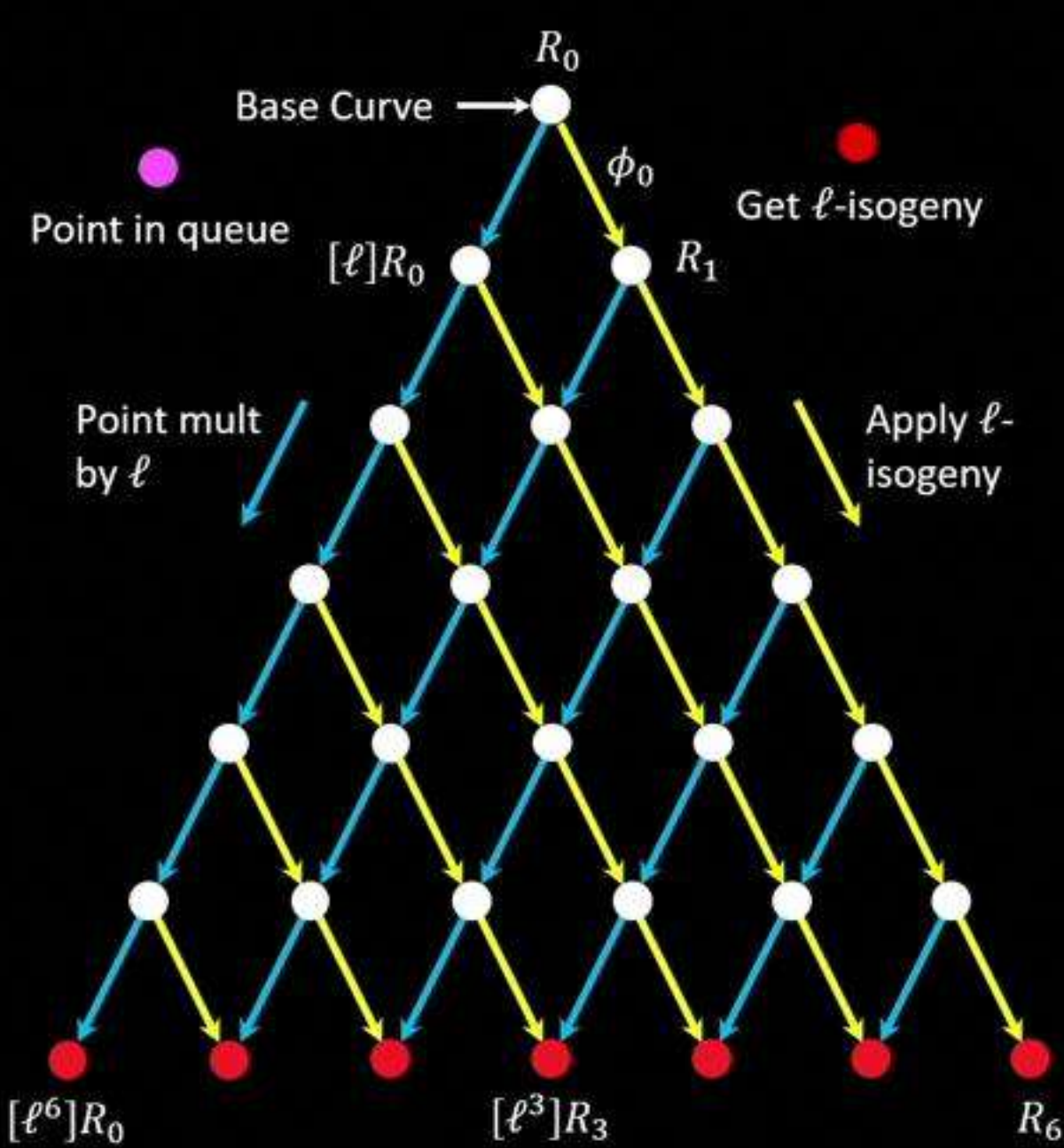


Order of  $[l^4]R_0$  is  $l^3$

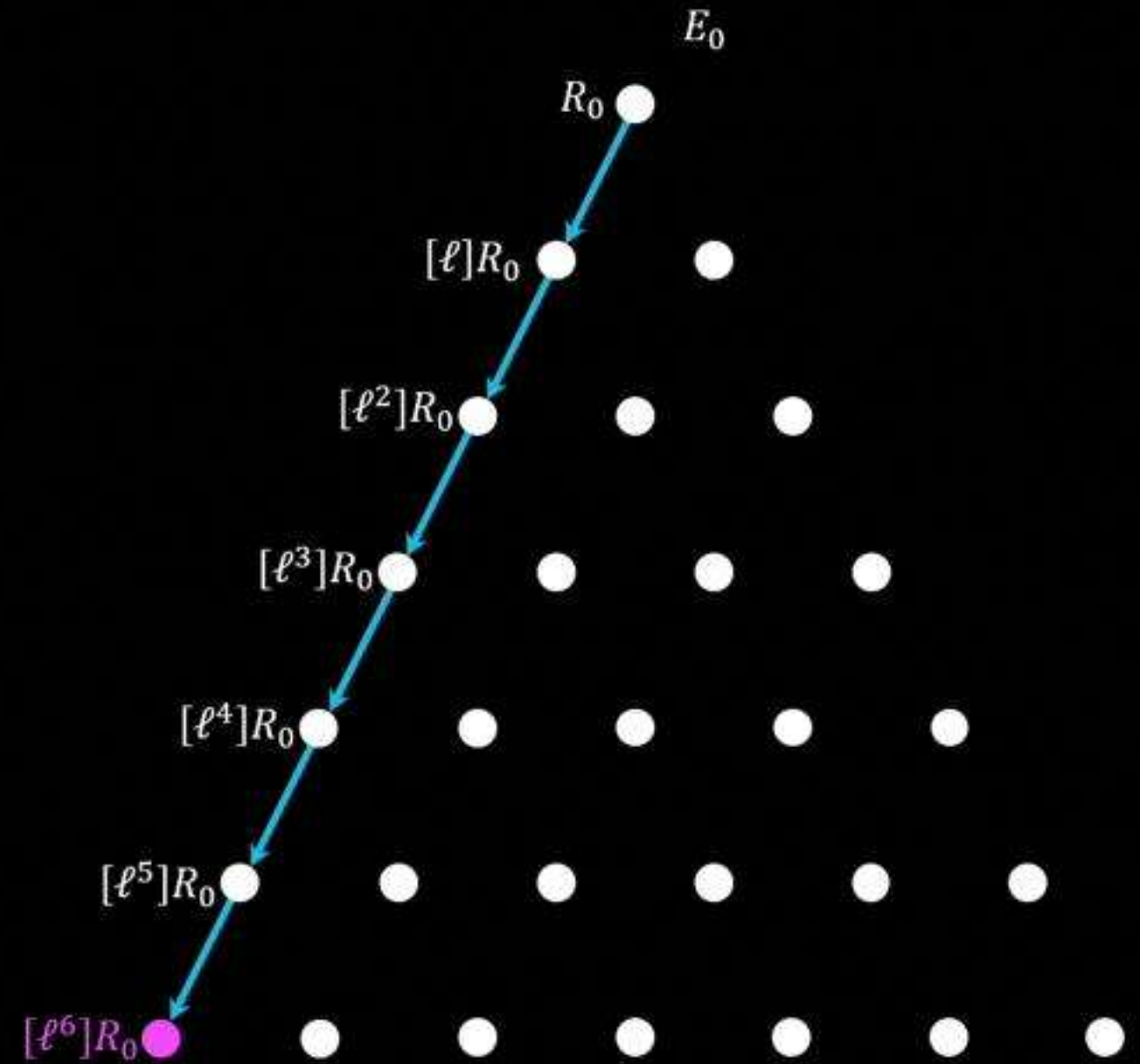


# Large degree isogeny computations

$e = 7$

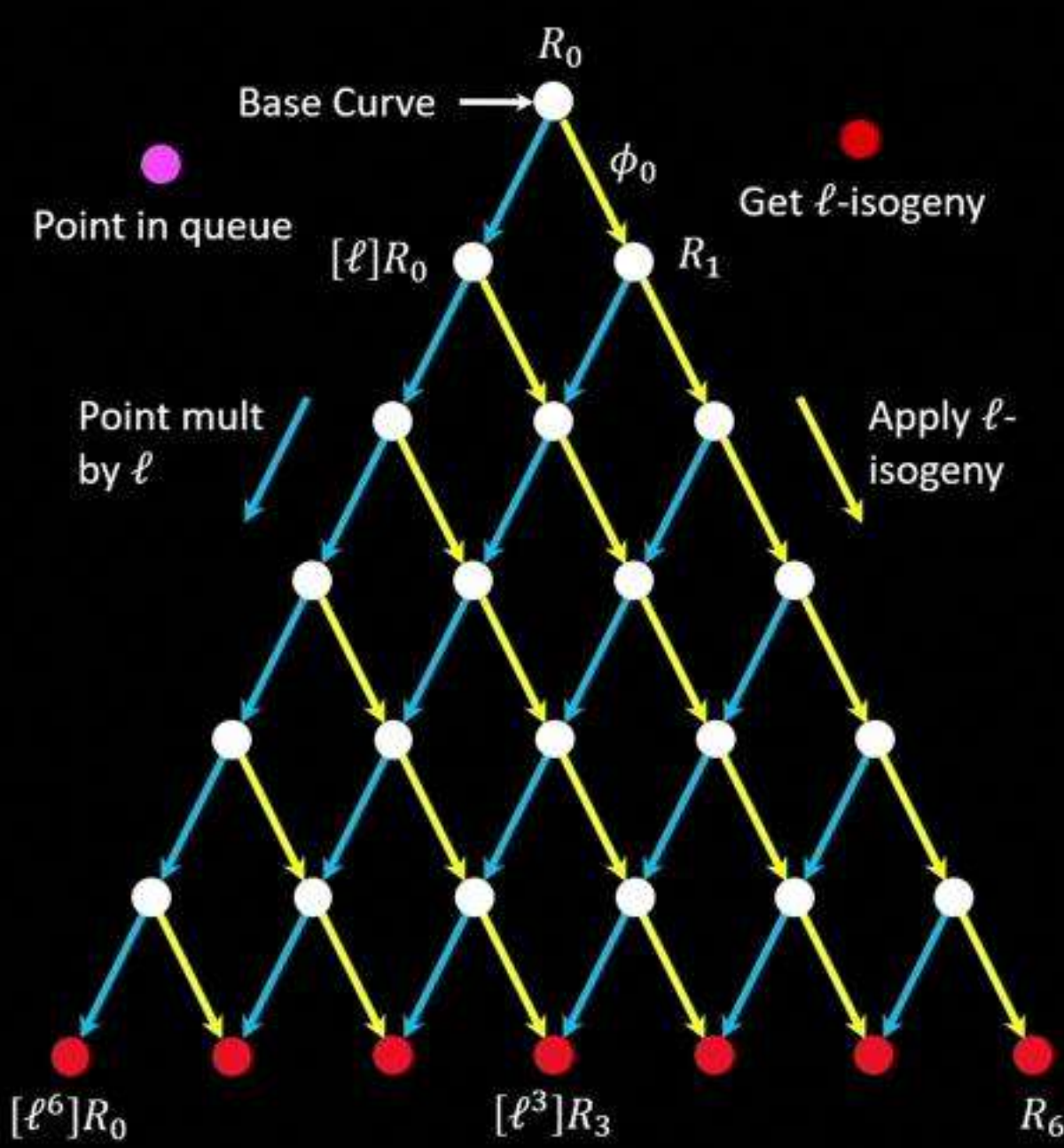


Order of  $[l^6]R_0$  is  $\ell$

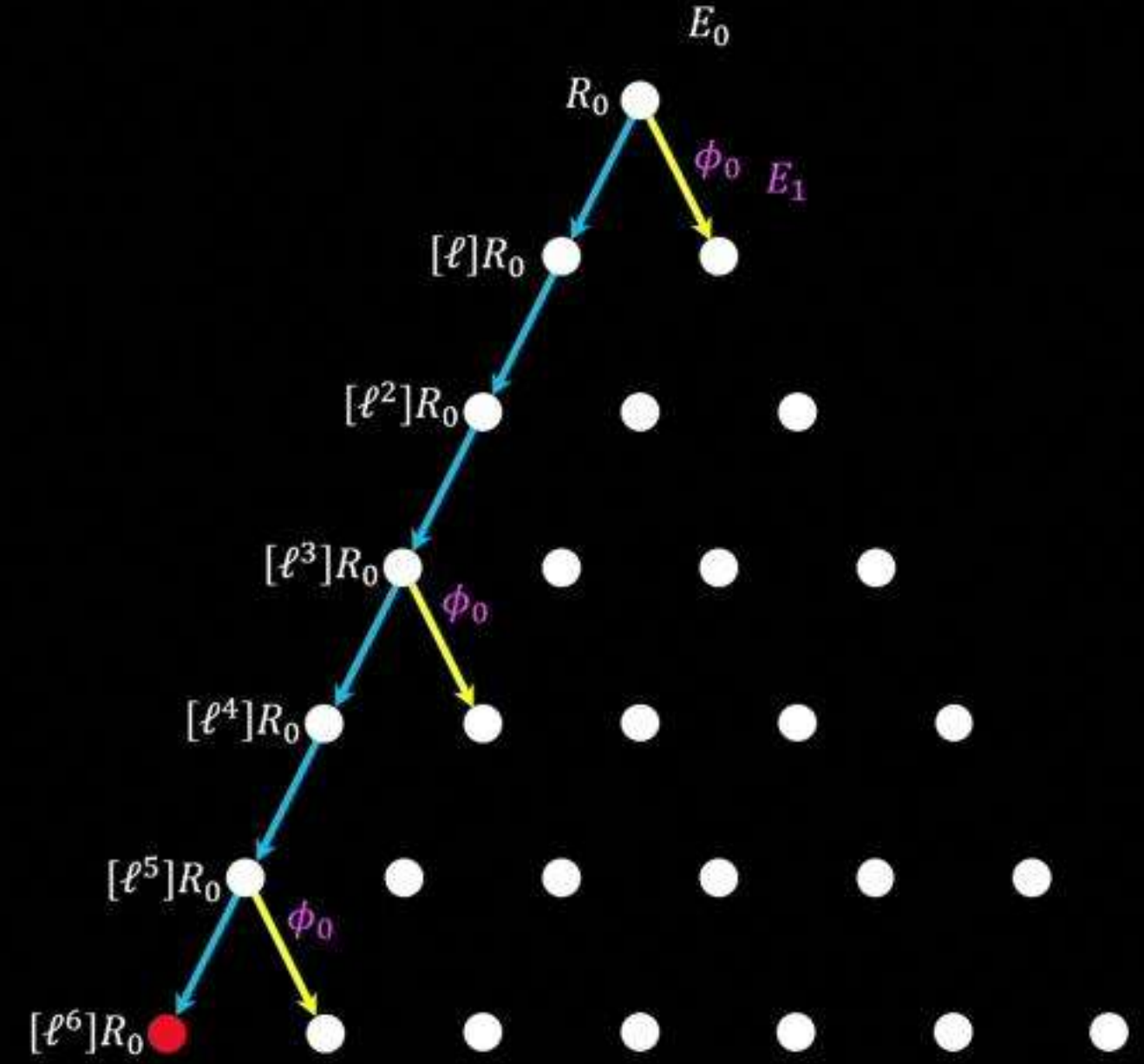


# Large degree isogeny computations

$e = 7$

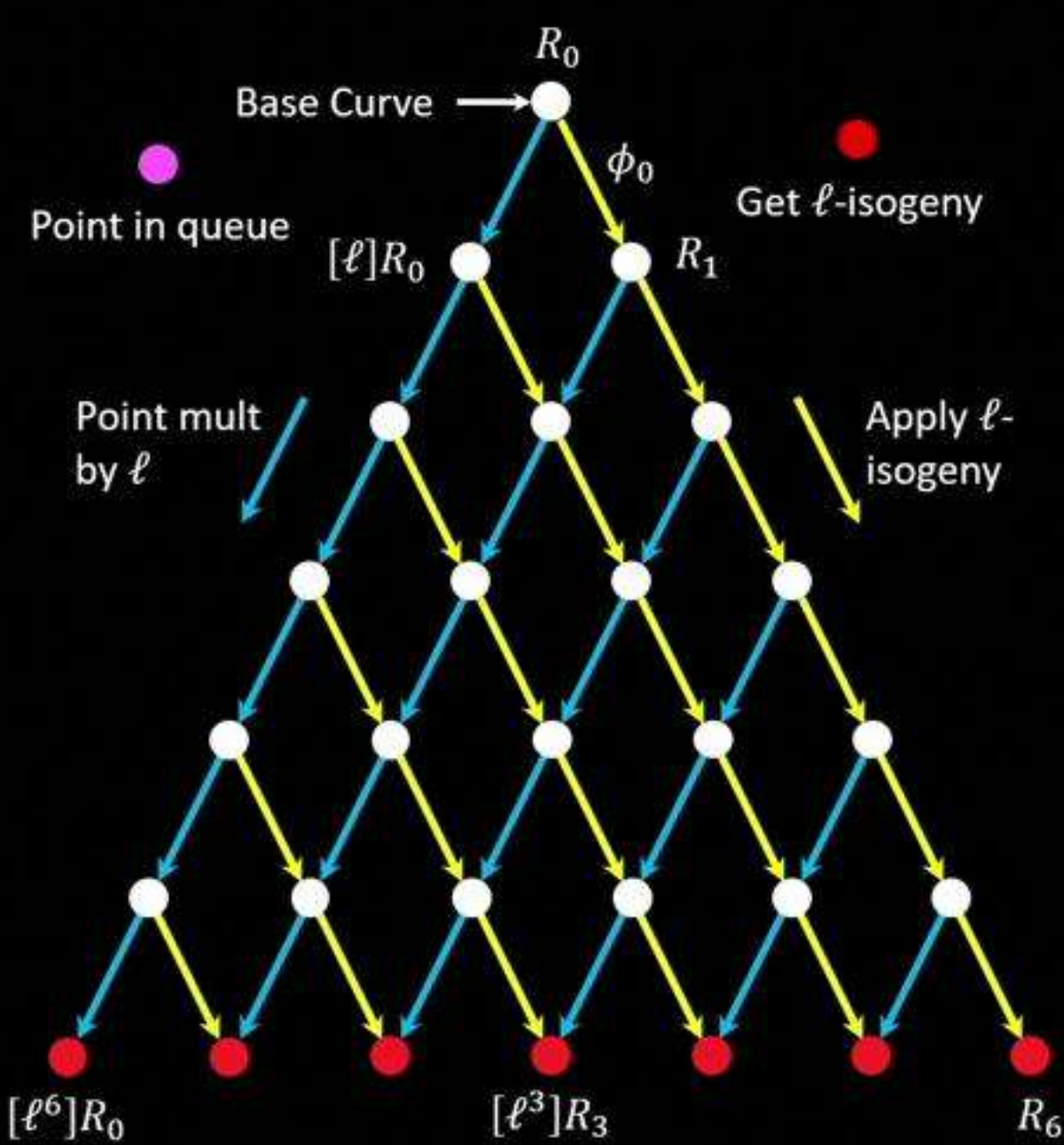


$$\phi_0 := E_0 / \langle [l^6]R_0 \rangle$$
$$E_1 = \phi_0(E_0)$$

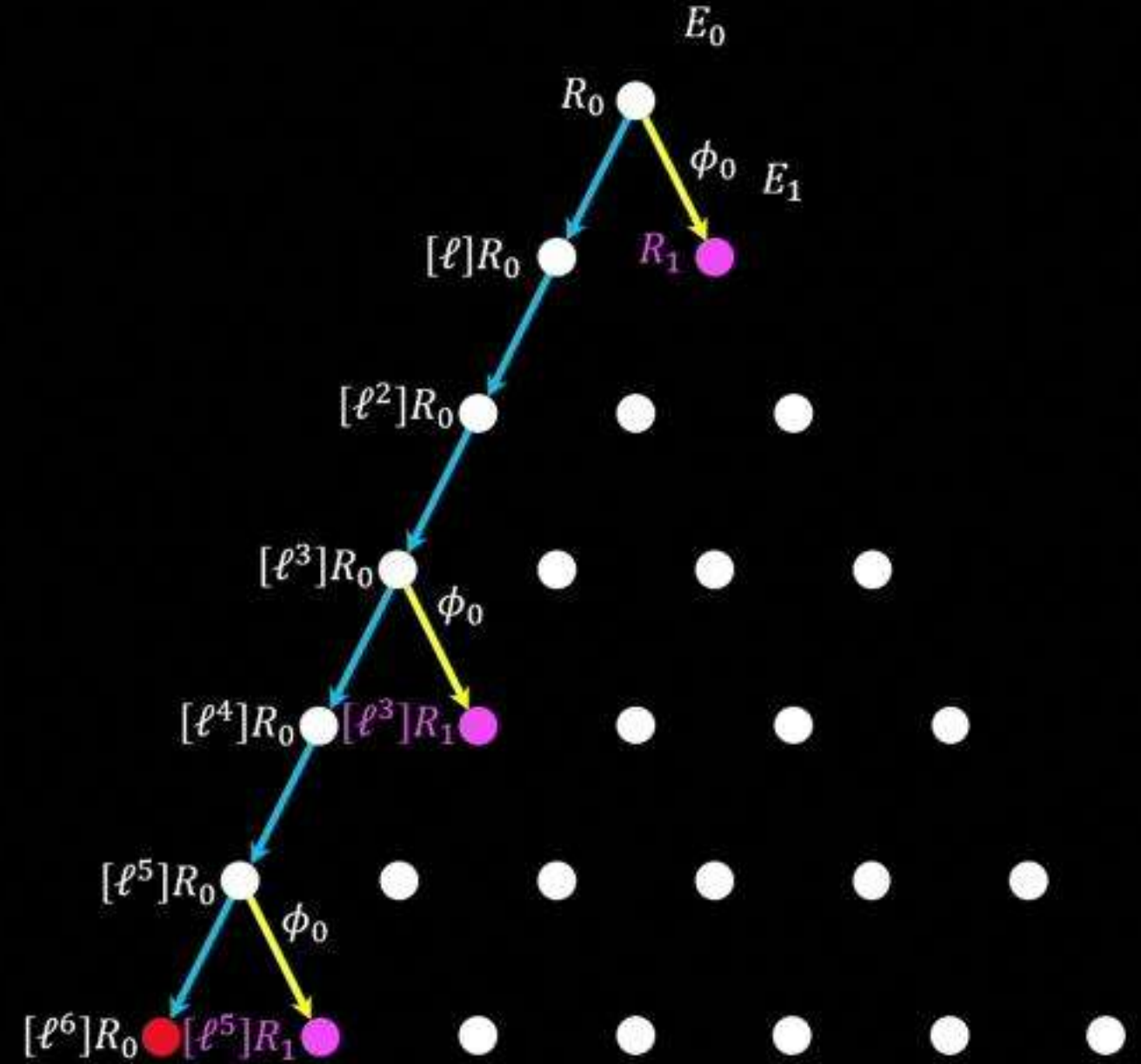


# Large degree isogeny computations

$e = 7$

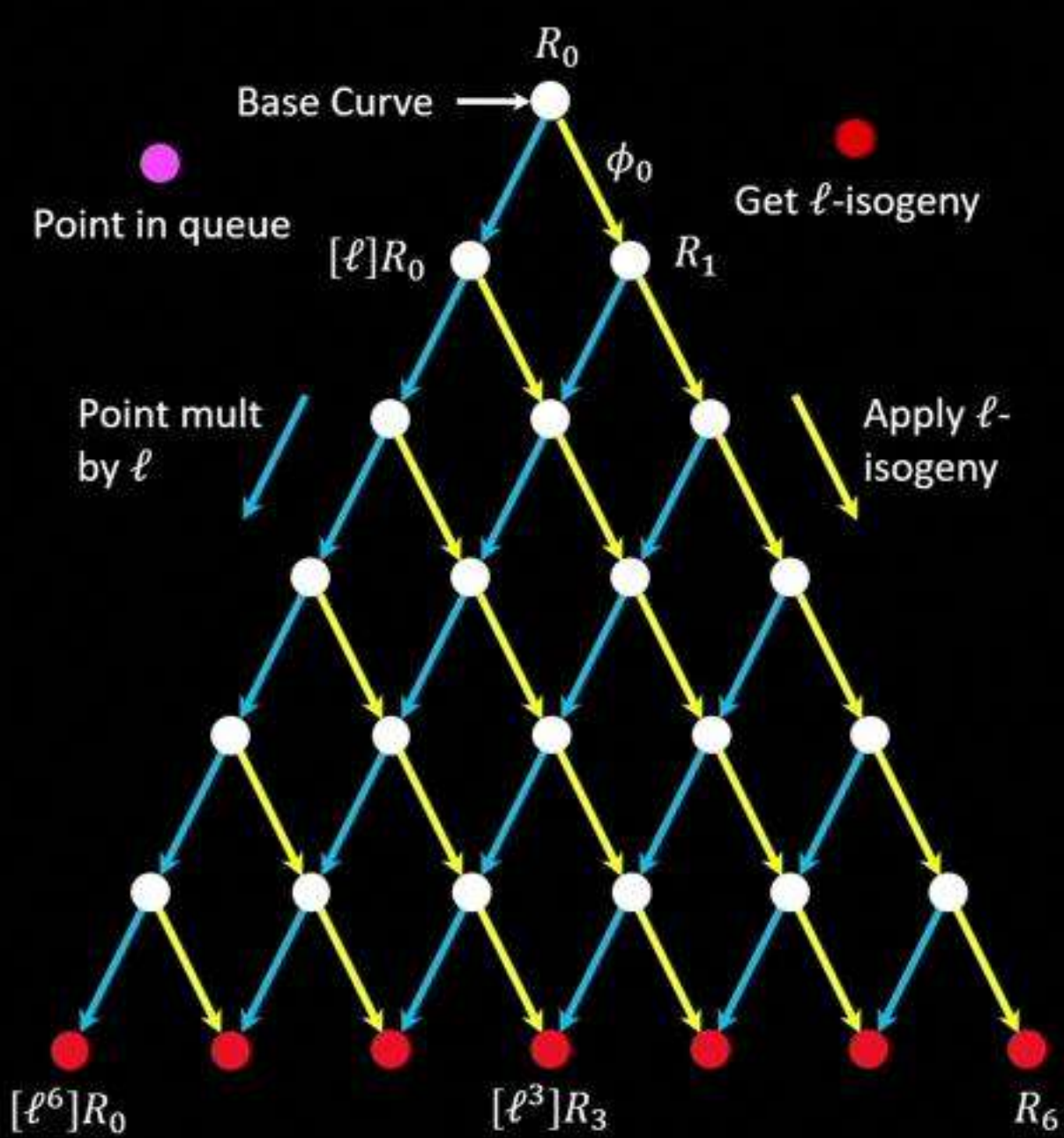


$R_1 = \phi_0(R_0)$   
Order of  $[l^5]R_1$  is  $\ell$



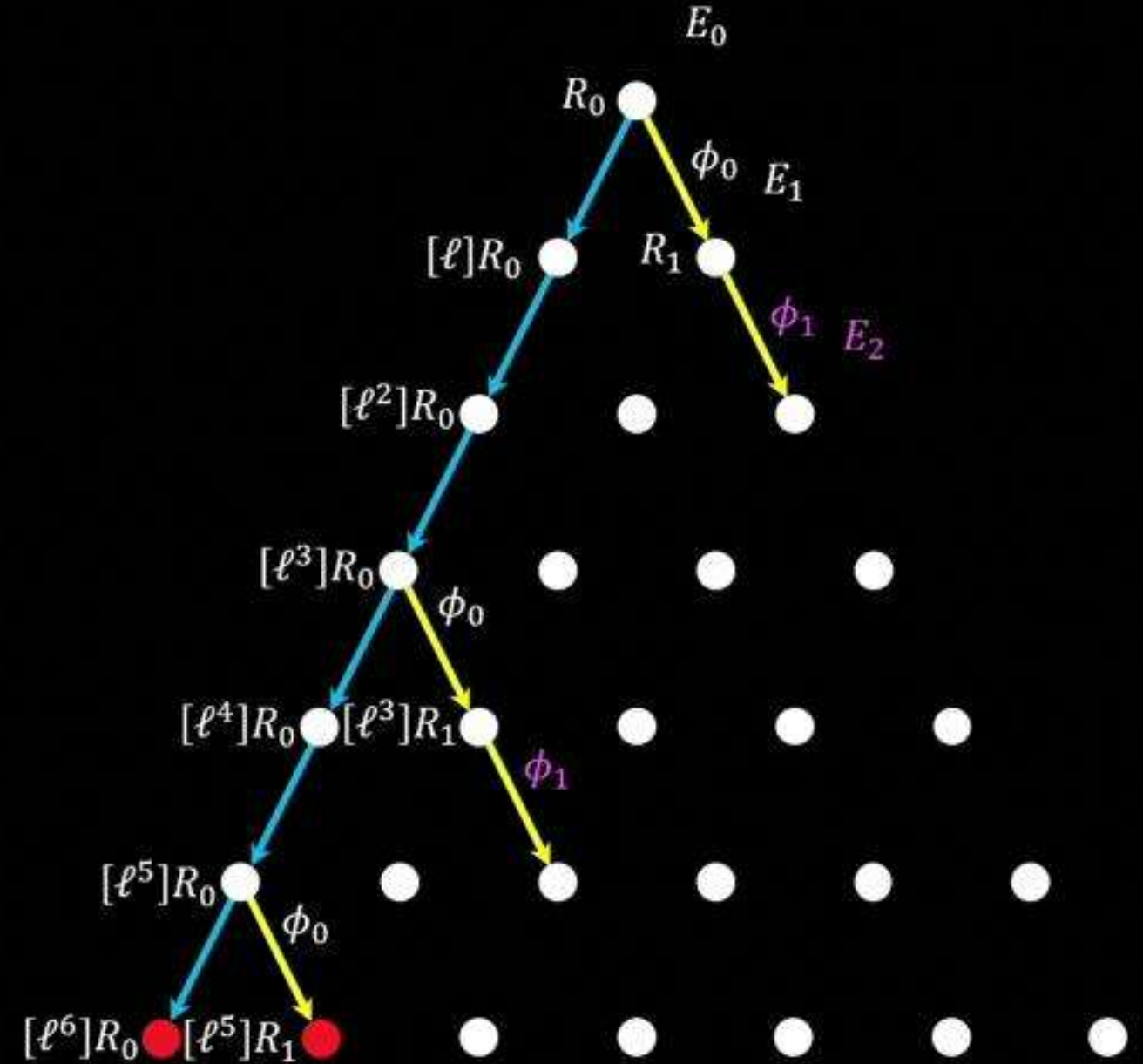
# Large degree isogeny computations

$e = 7$



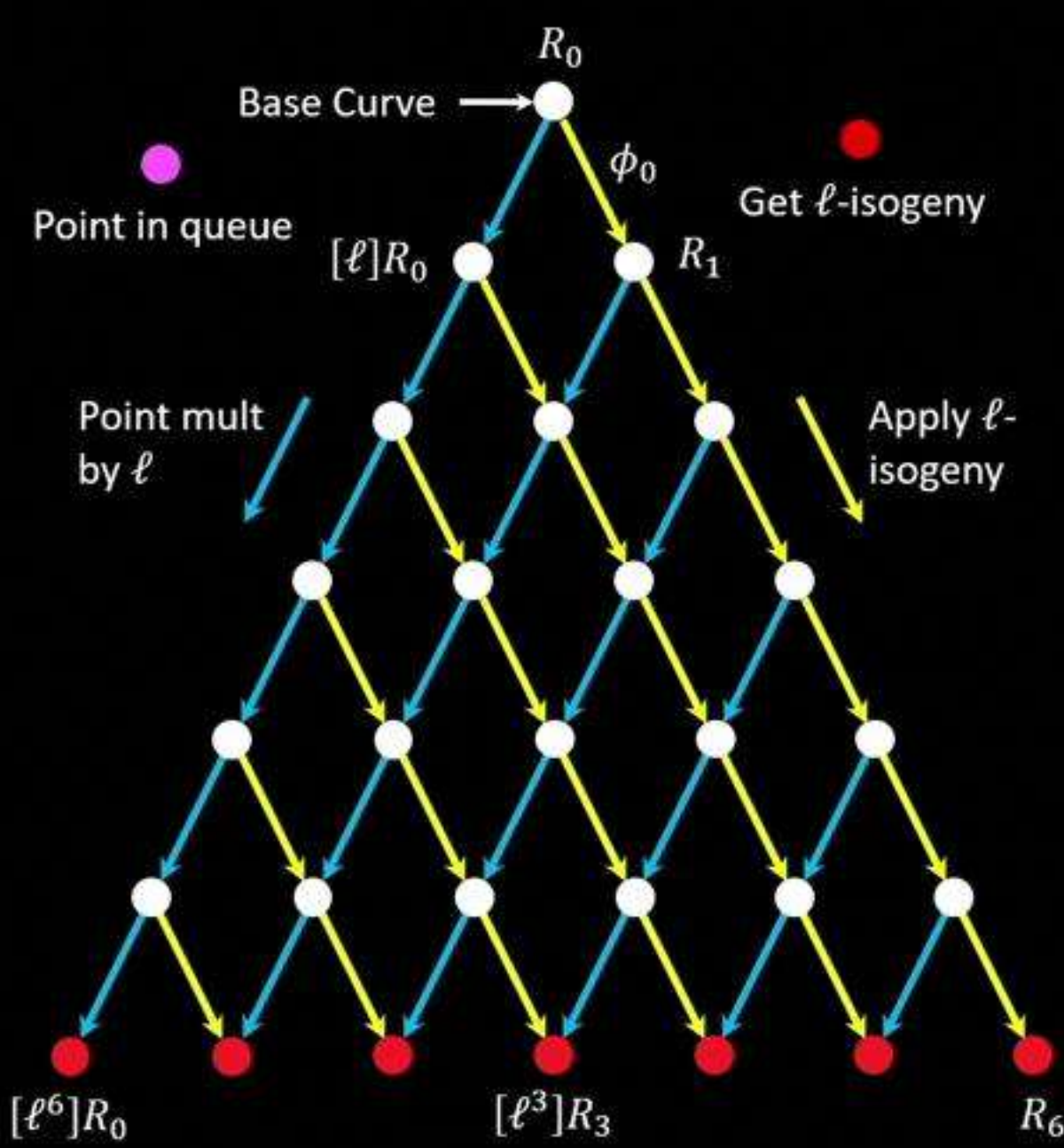
$$\phi_1 := E_1 / \langle [l^5]R_1 \rangle$$

$$E_2 = \phi_1(E_1)$$

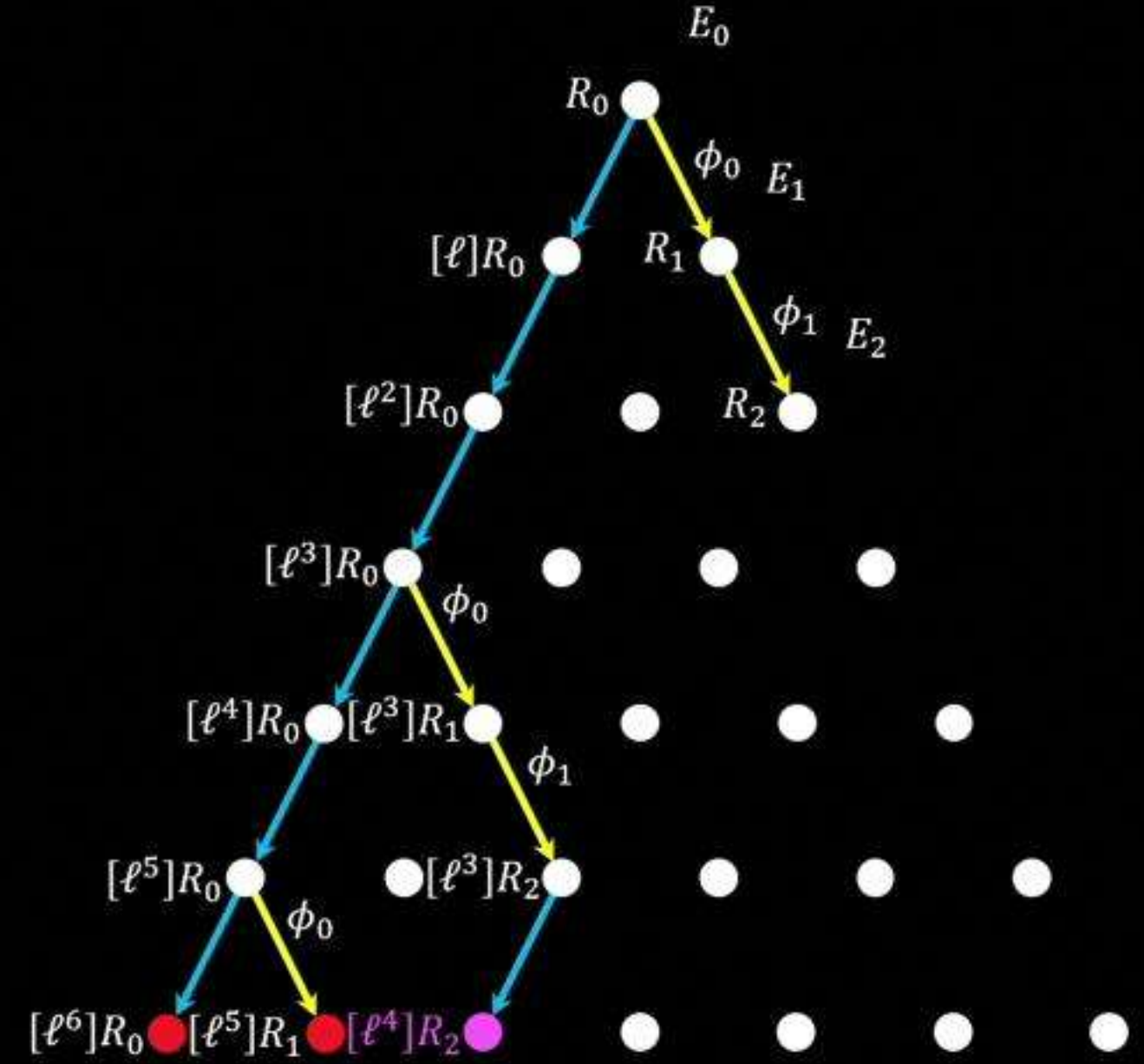


# Large degree isogeny computations

$e = 7$



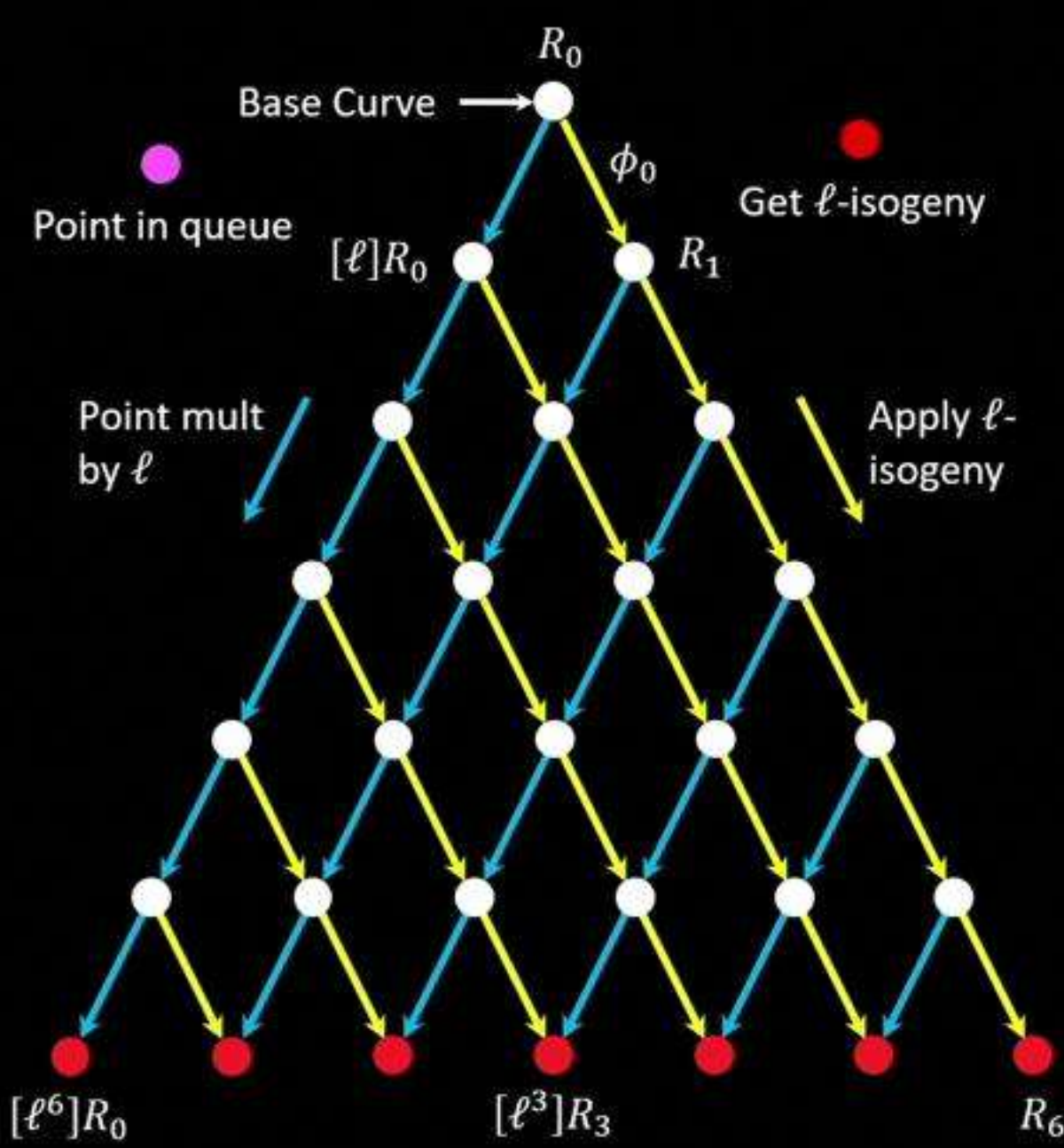
Order of  $[l^4]R_2$  is  $\ell$





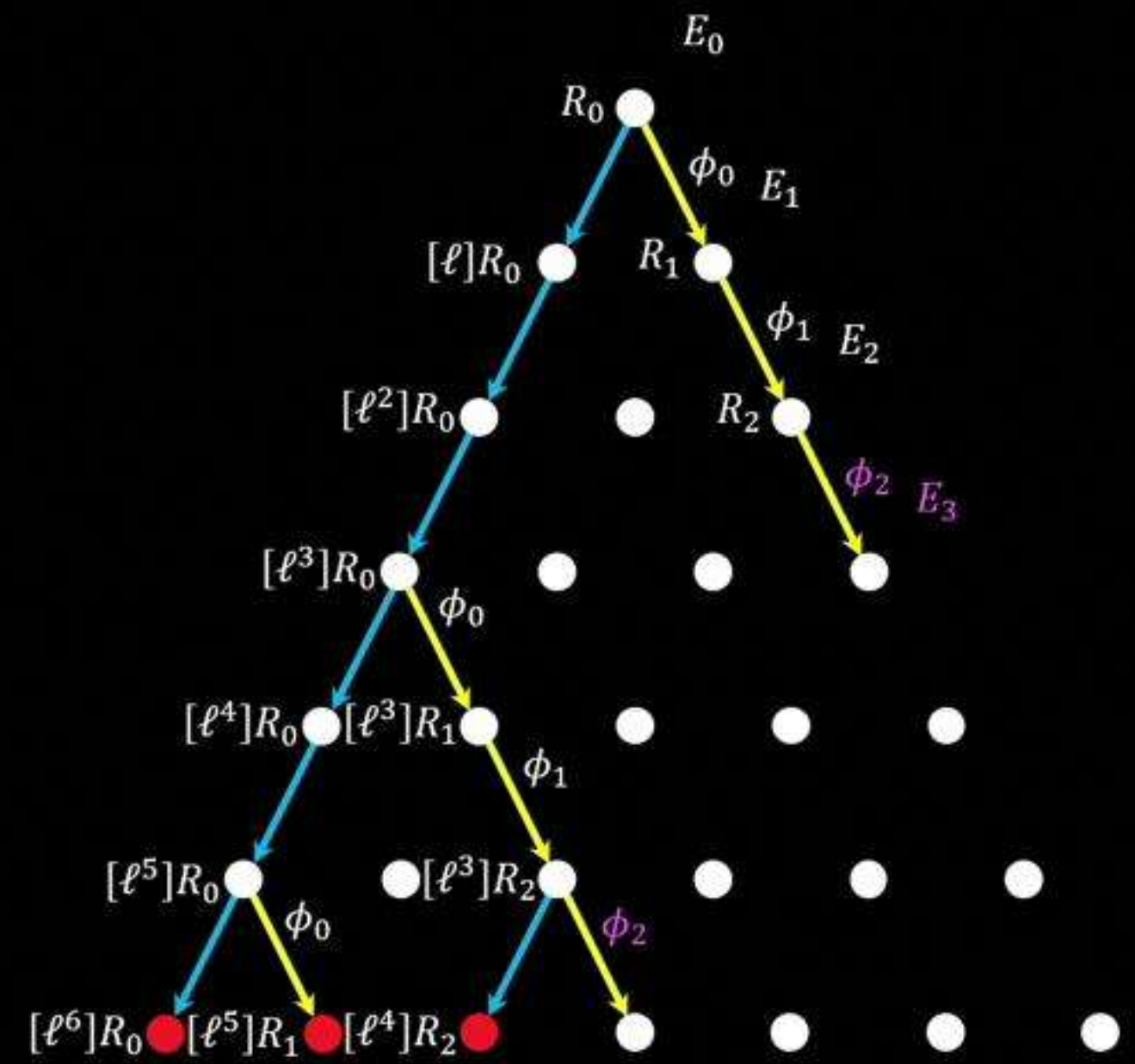
# Large degree isogeny computations

$e = 7$



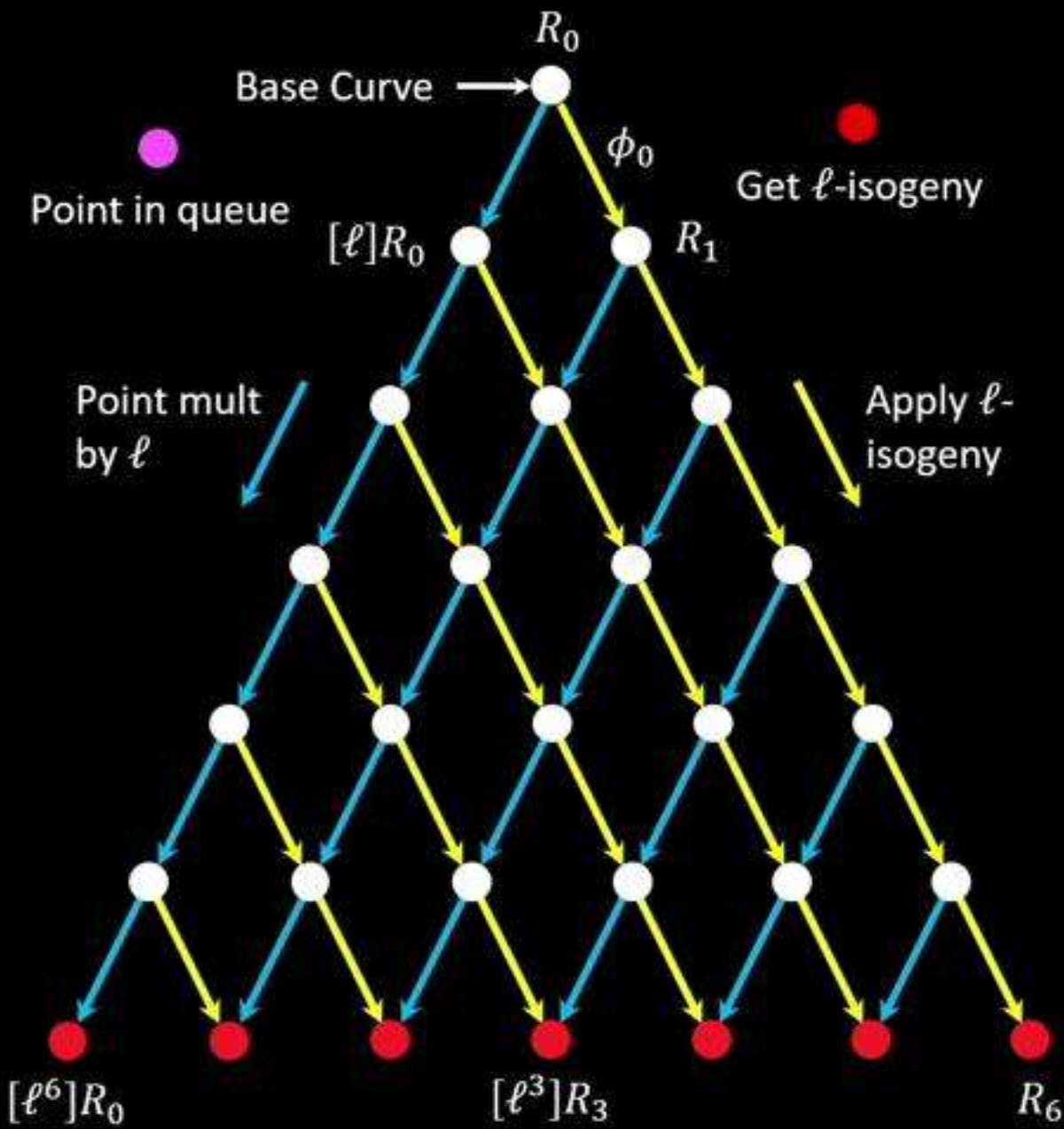
$$\phi_2 := E_2 / \langle [l^4]R_2 \rangle$$

$$E_3 = \phi_2(E_2)$$

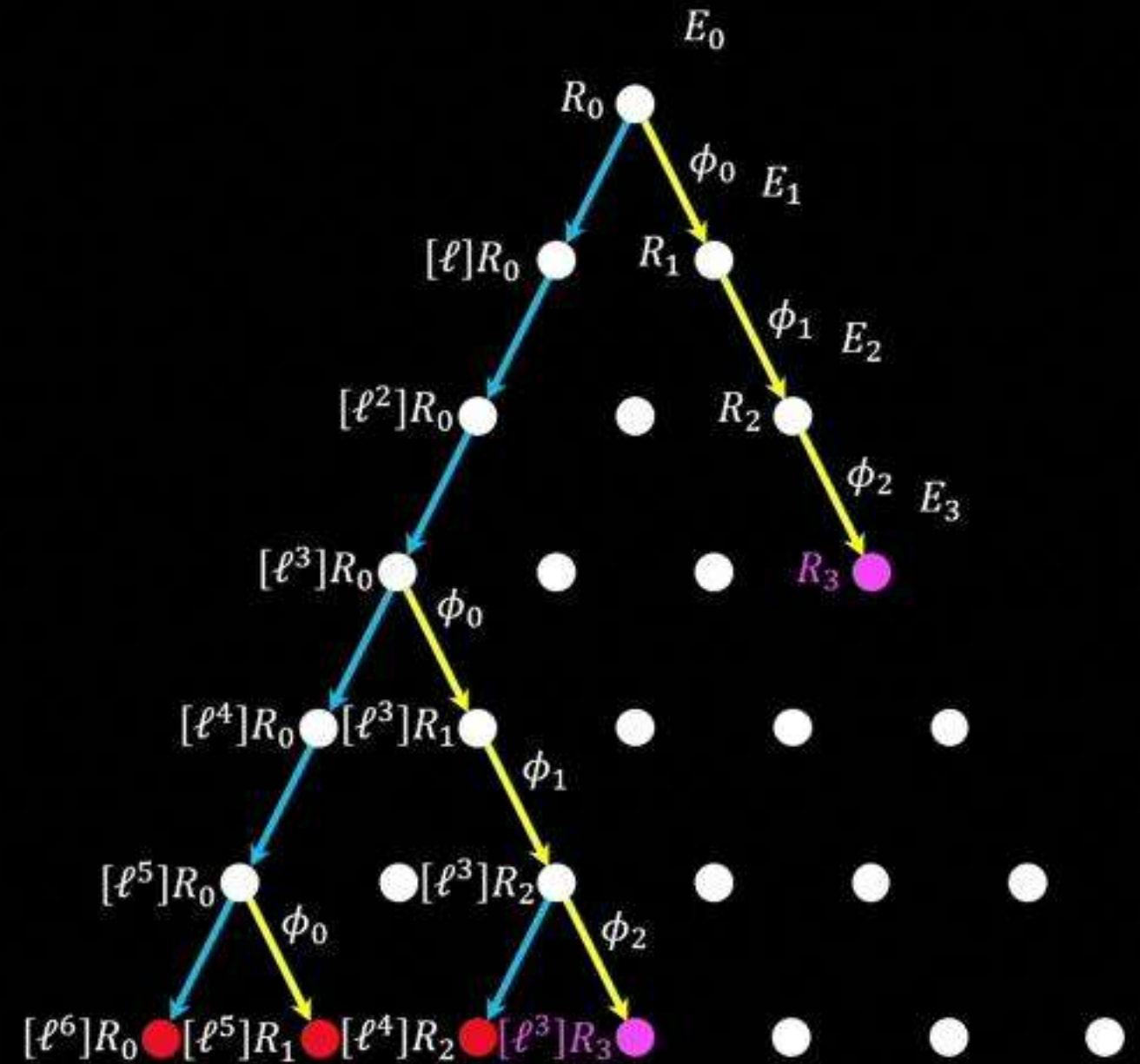


# Large degree isogeny computations

$e = 7$

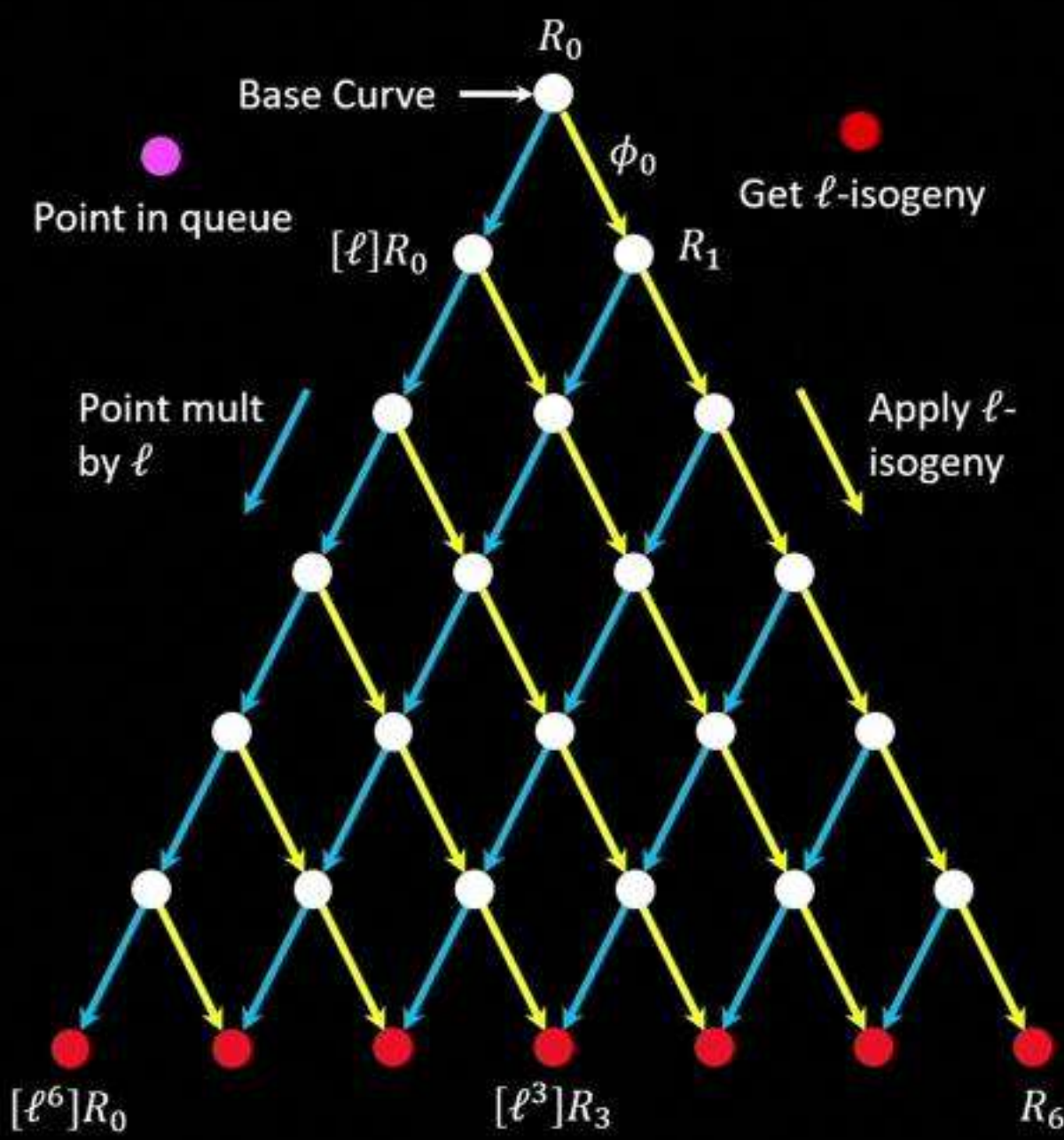


$R_3 = \phi_2(R_2)$   
 Order of  $[l^3]R_3$  is  $l$



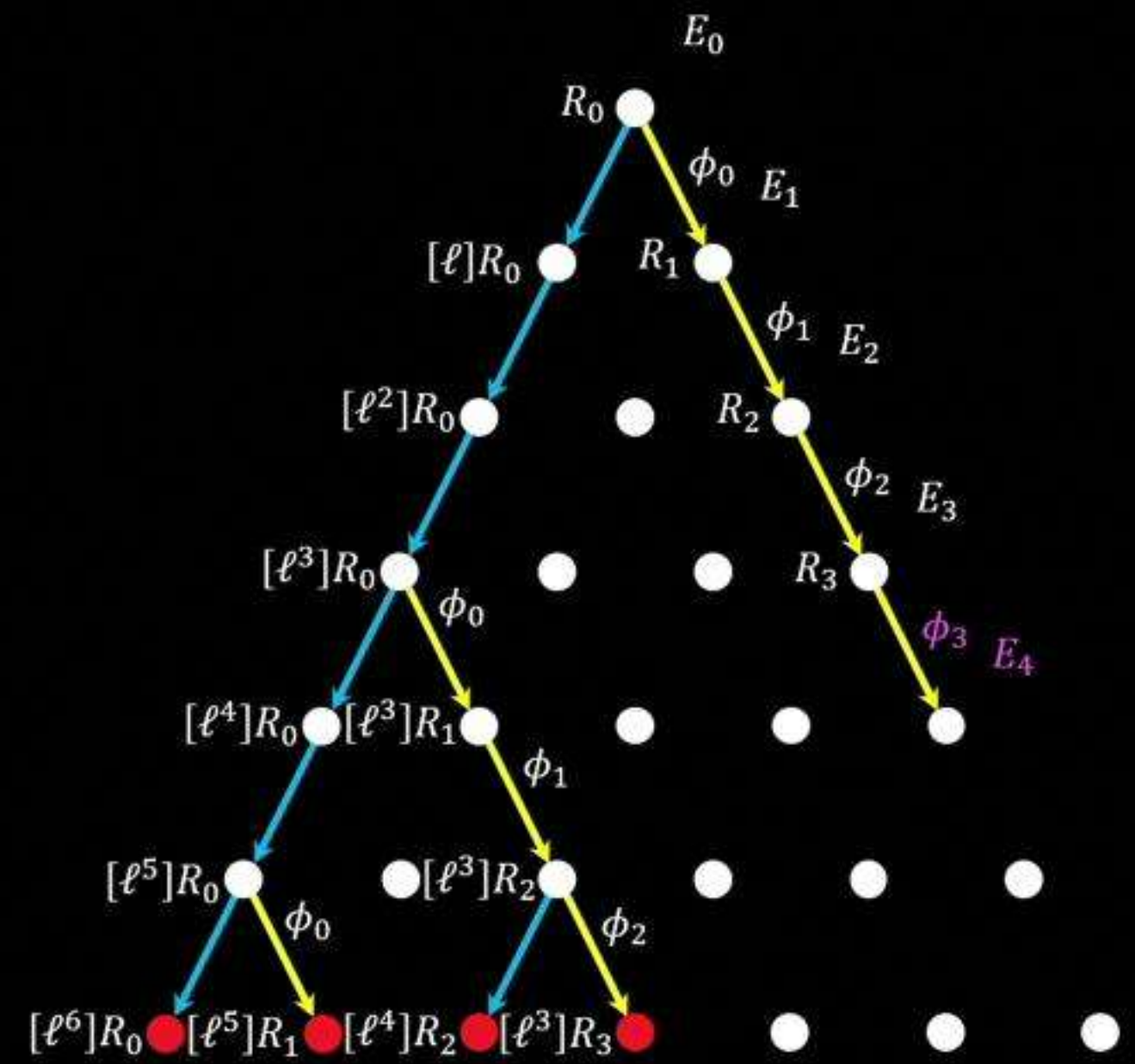
# Large degree isogeny computations

$e = 7$



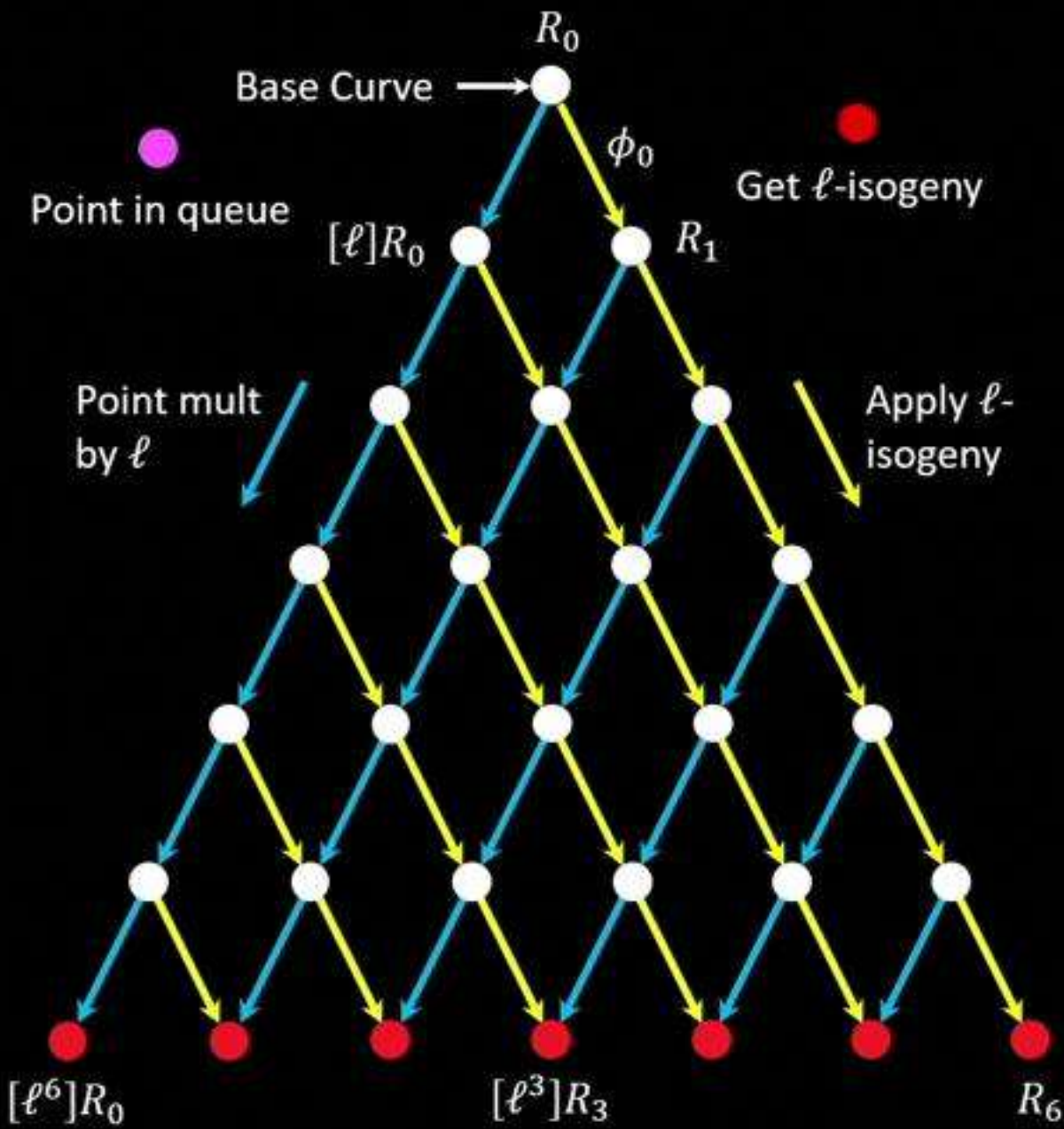
$$\phi_3 := E_3 / \langle [\ell^3]R_3 \rangle$$

$$E_4 = \phi_3(E_3)$$

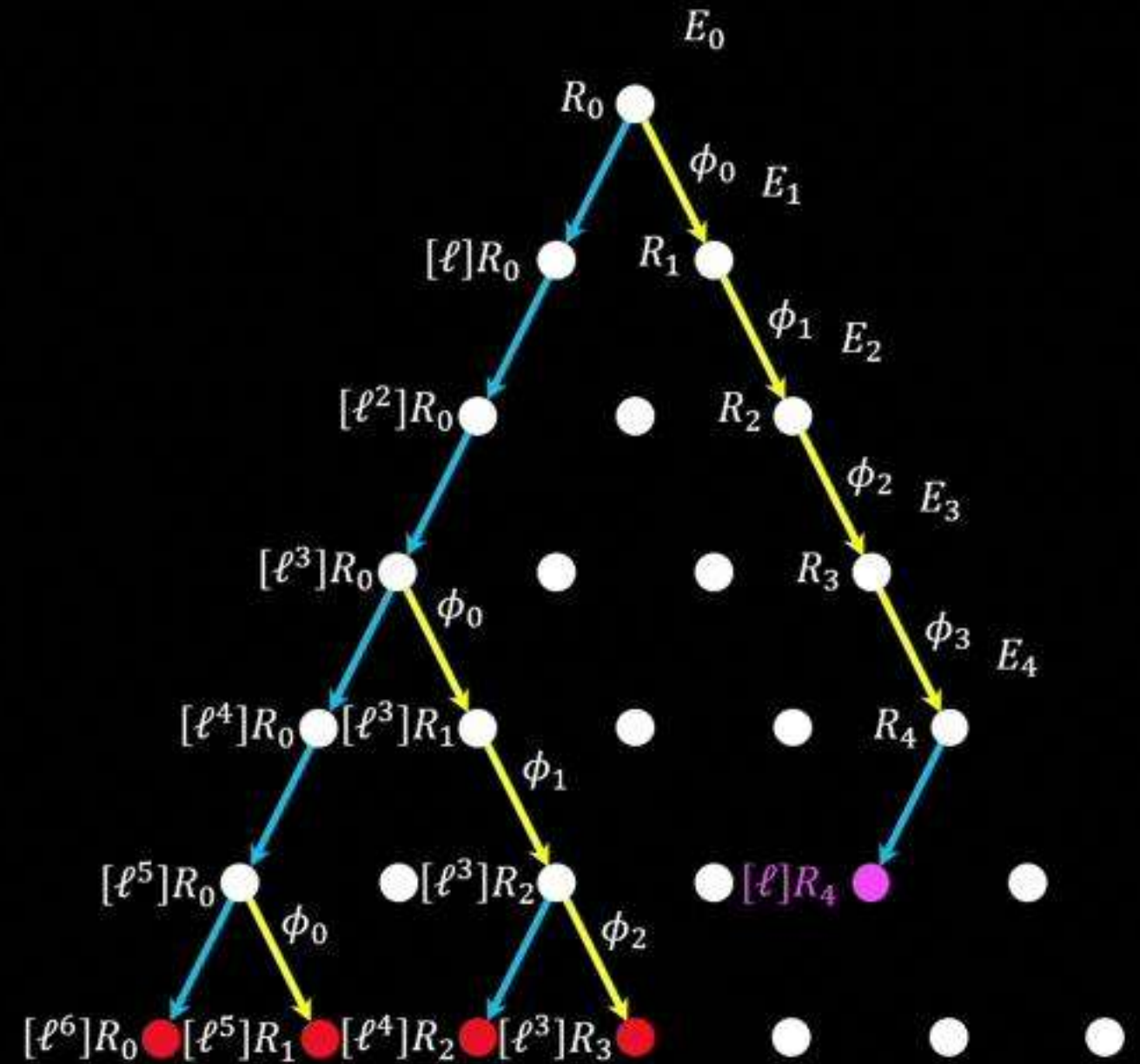


# Large degree isogeny computations

$e = 7$

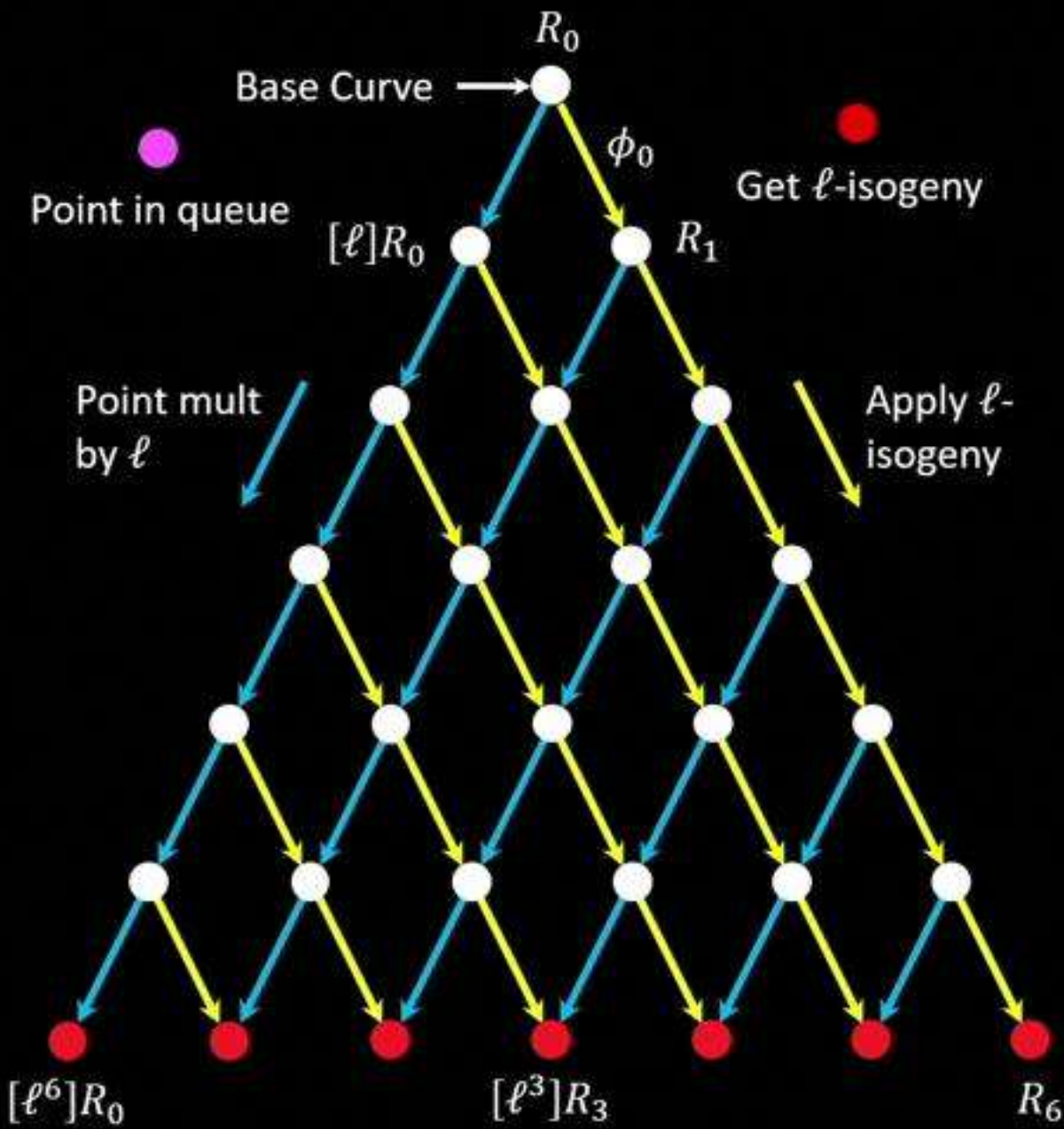


Order of  $[l]R_4$  is  $\ell^2$

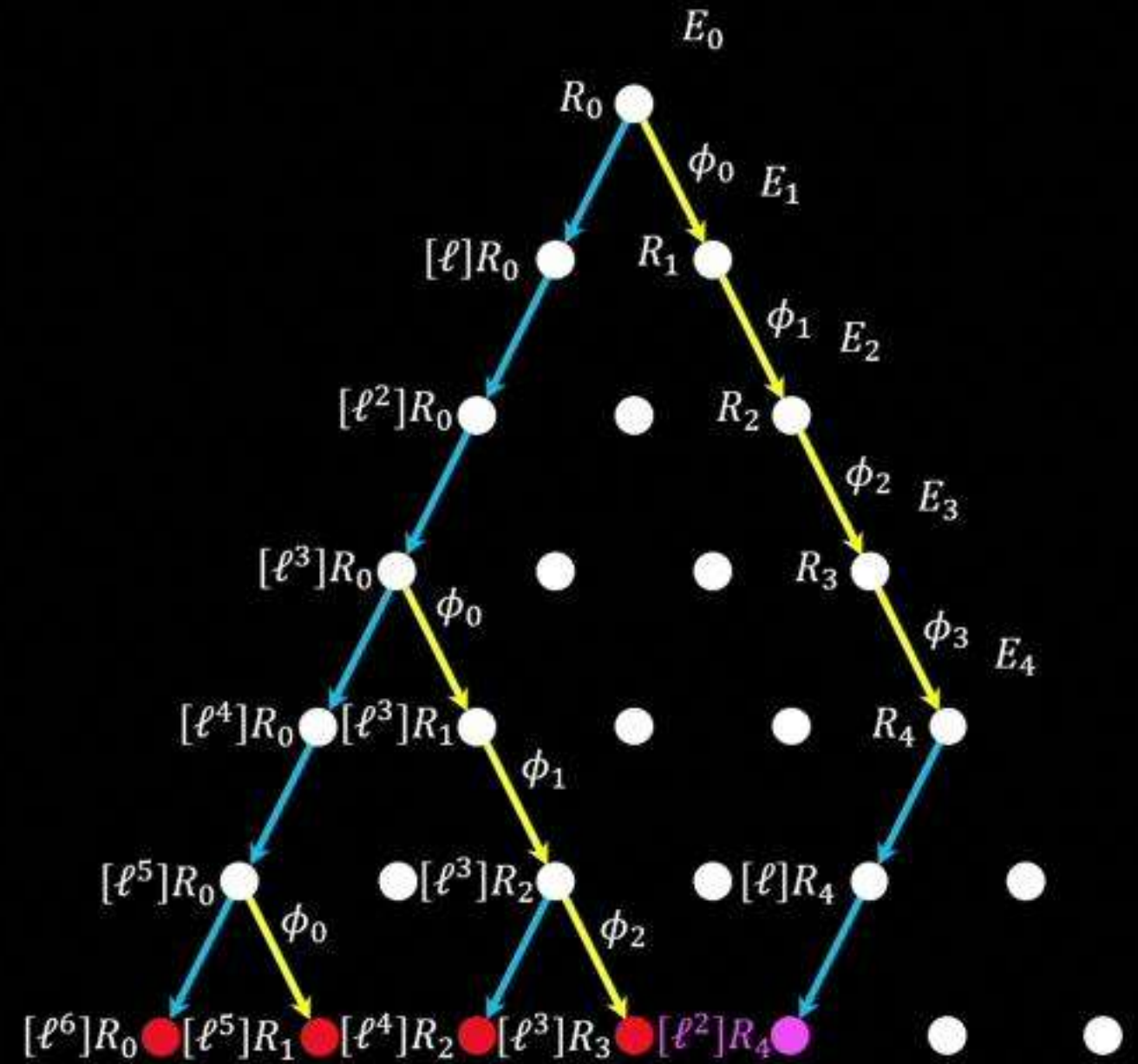


# Large degree isogeny computations

$e = 7$

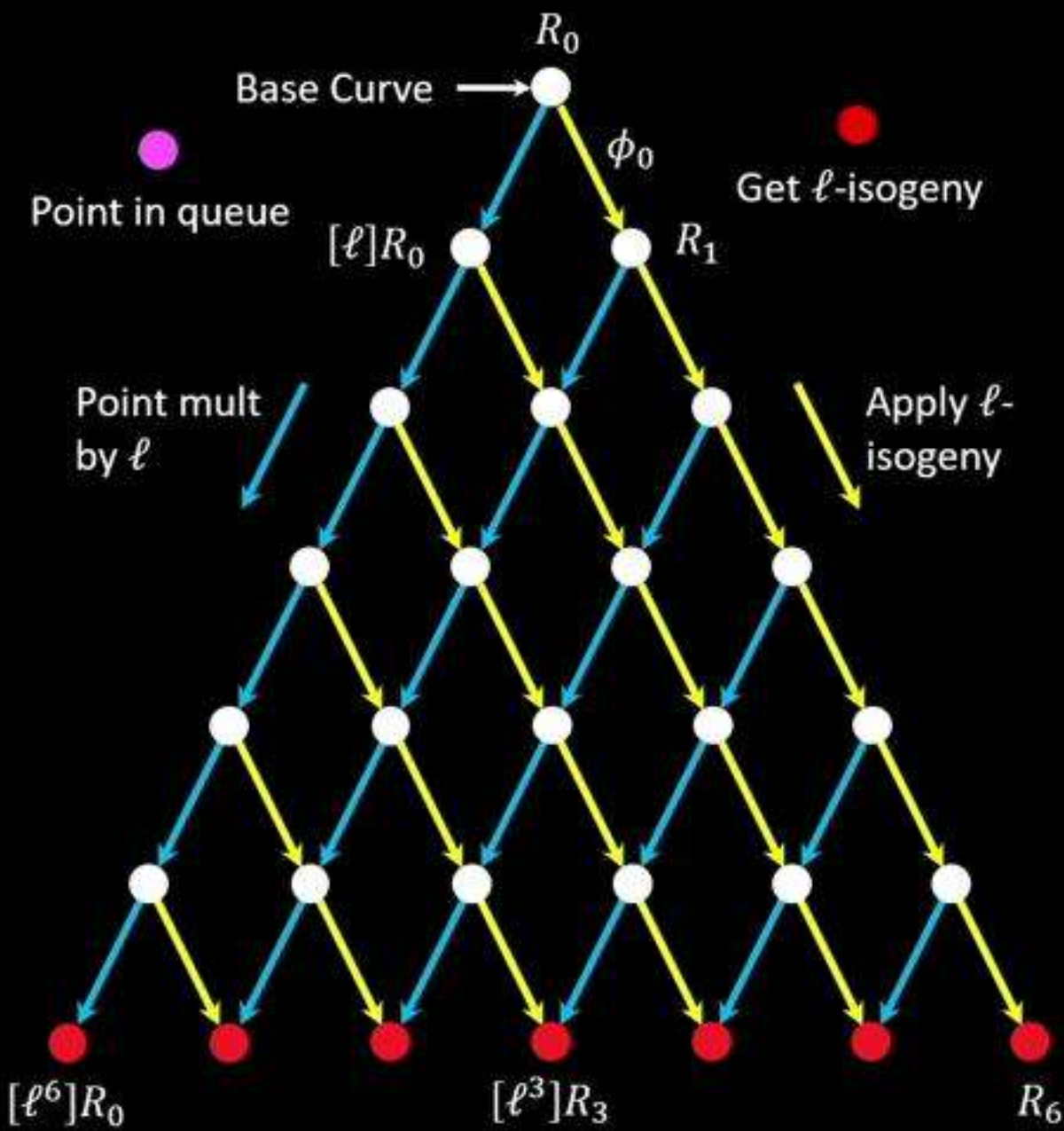


Order of  $[l^2]R_4$  is  $\ell$



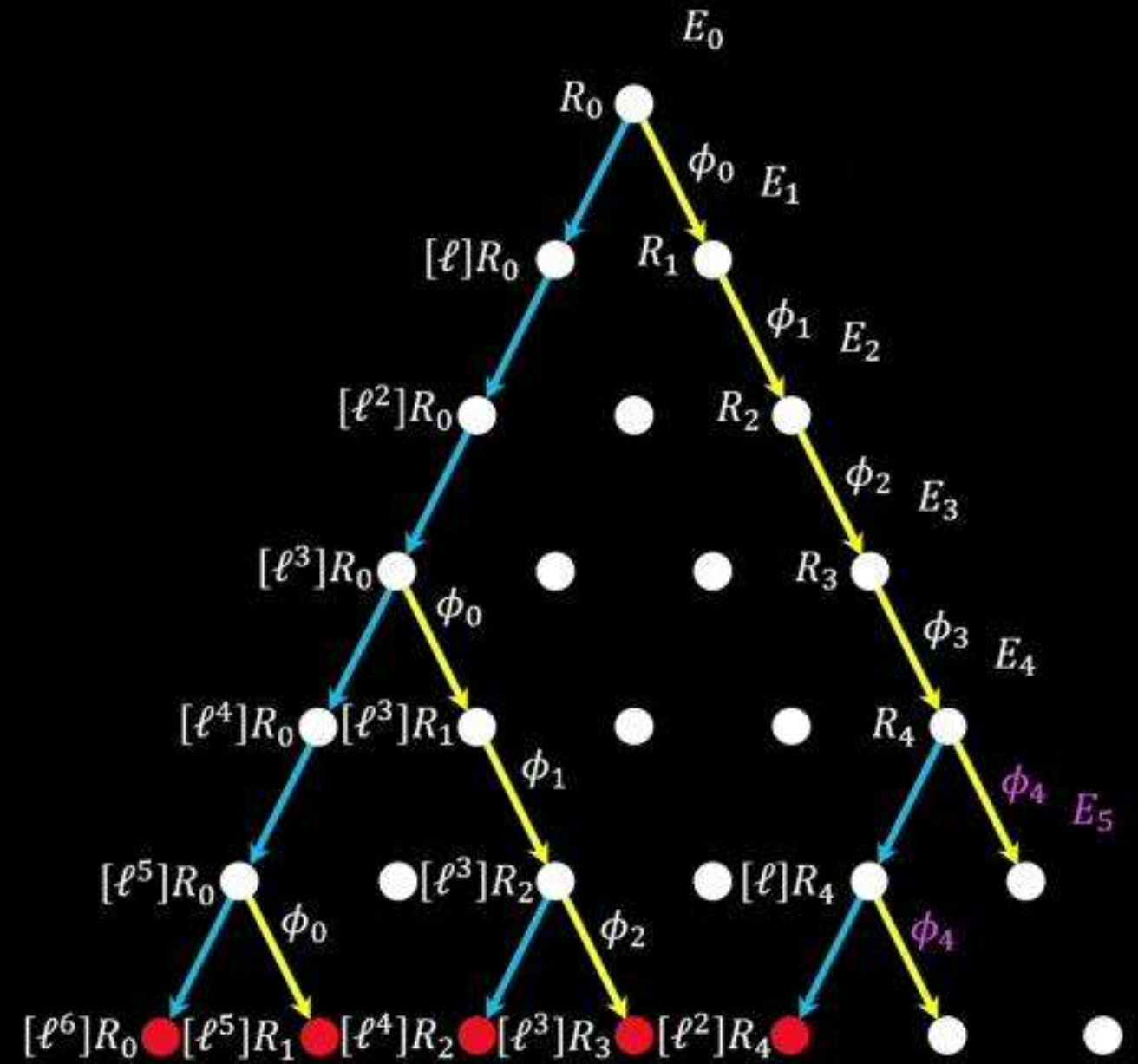
# Large degree isogeny computations

$e = 7$



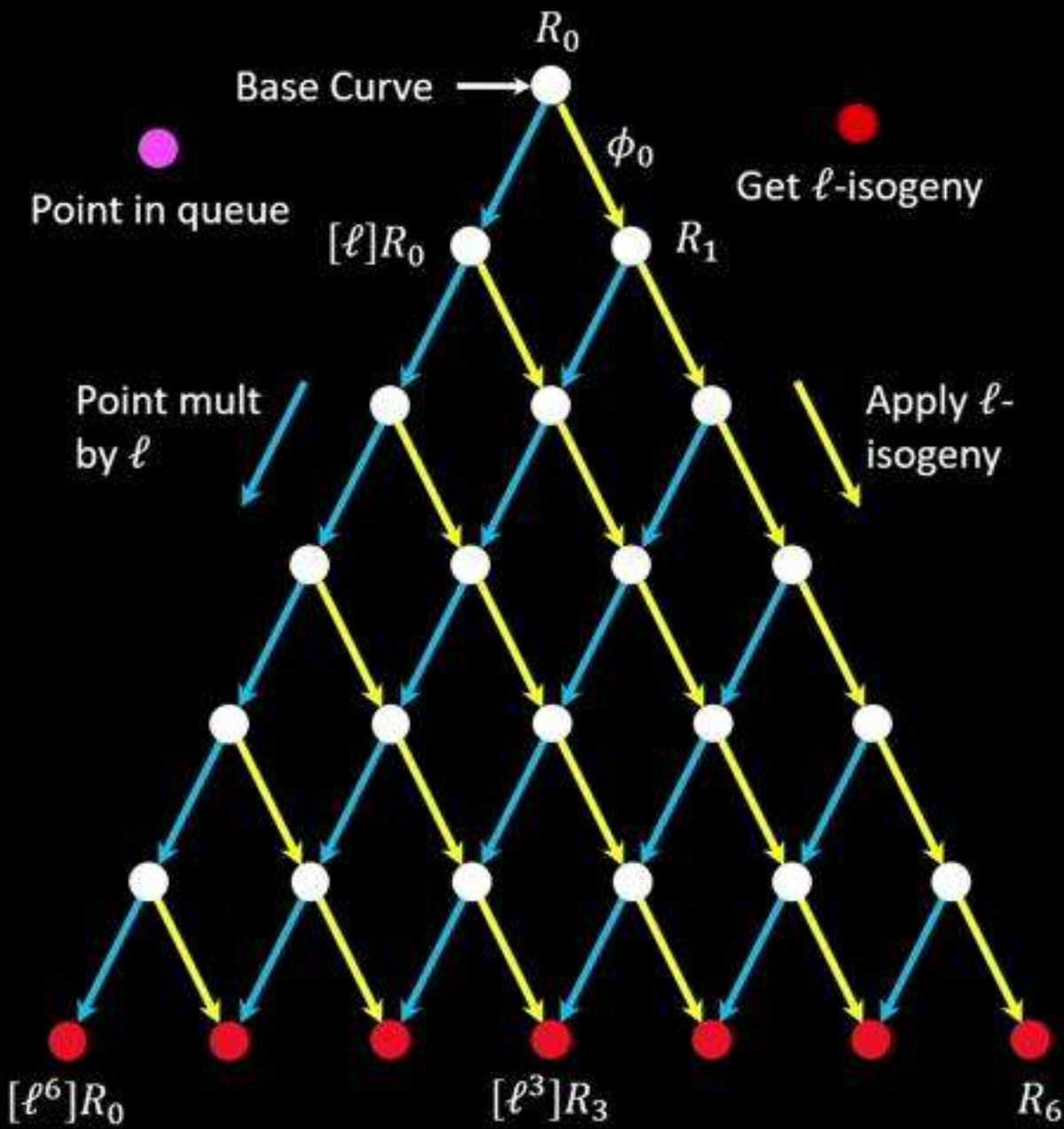
$$\phi_4 := E_4 / \langle [l^2]R_2 \rangle$$

$$E_5 = \phi_4(E_4)$$

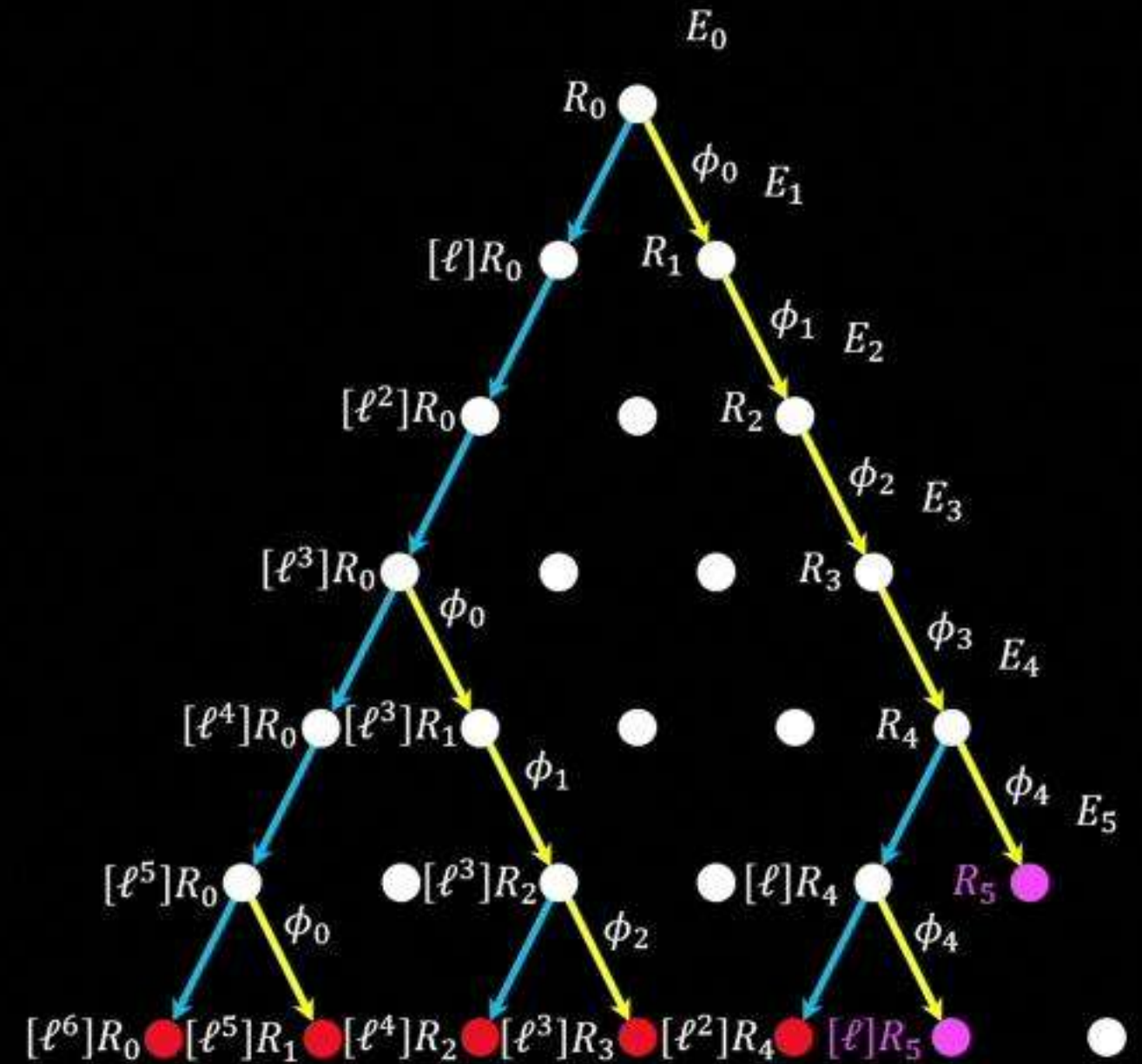


# Large degree isogeny computations

$e = 7$

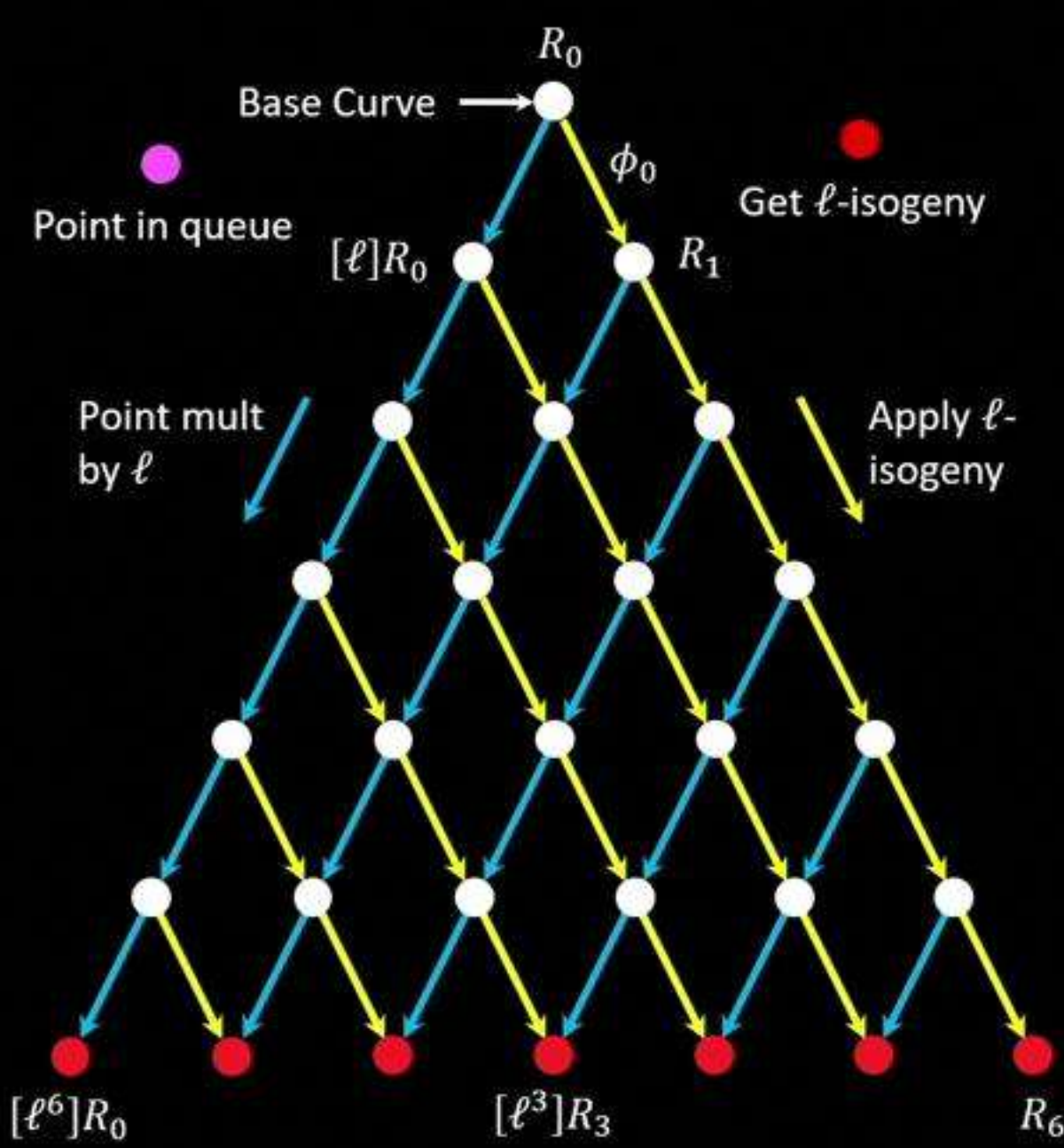


$R_5 = \phi_4(R_4)$   
 Order of  $[l]R_5$  is  $\ell$



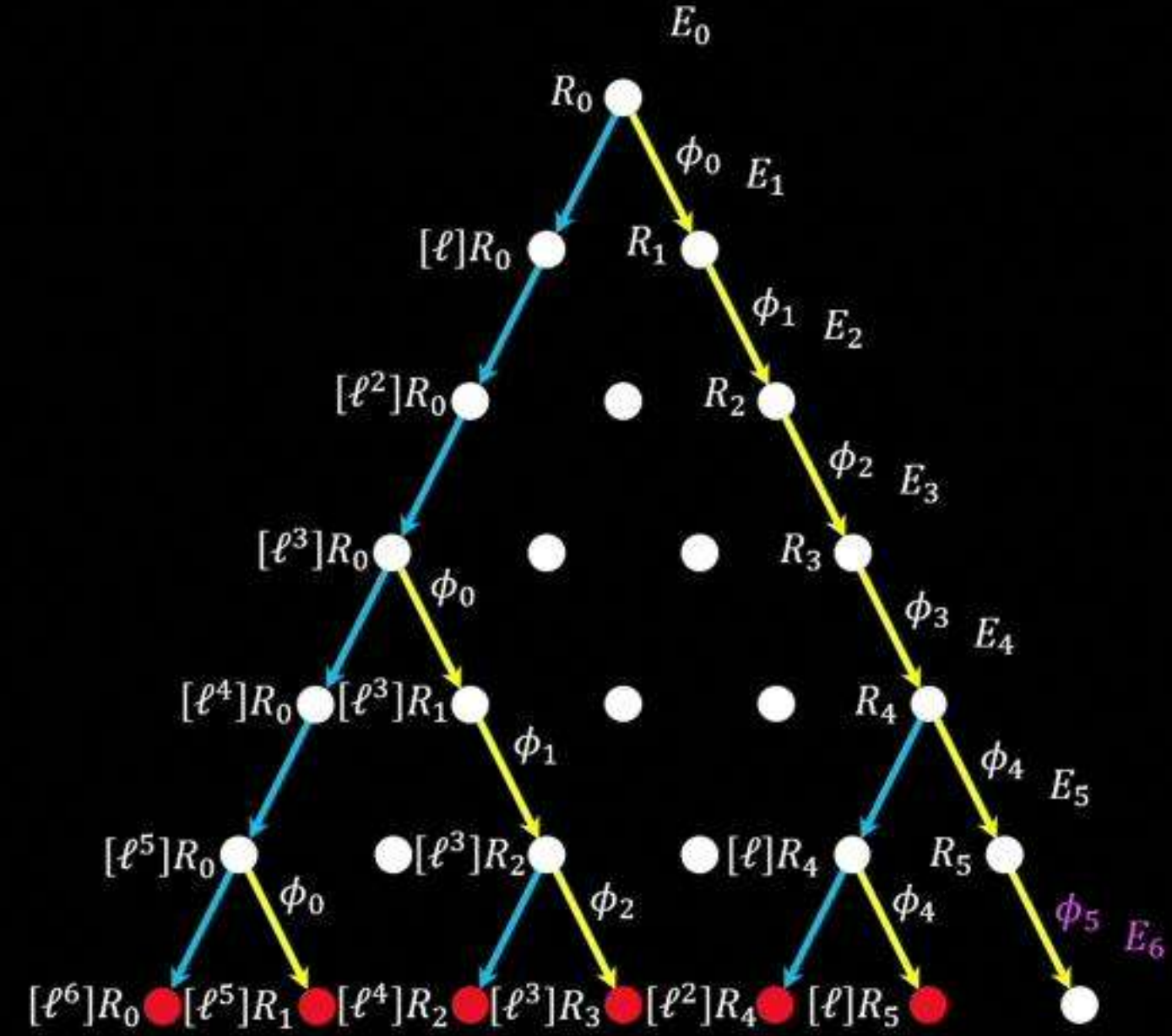
# Large degree isogeny computations

$e = 7$



$$\phi_5 := E_5 / \langle [l]R_5 \rangle$$

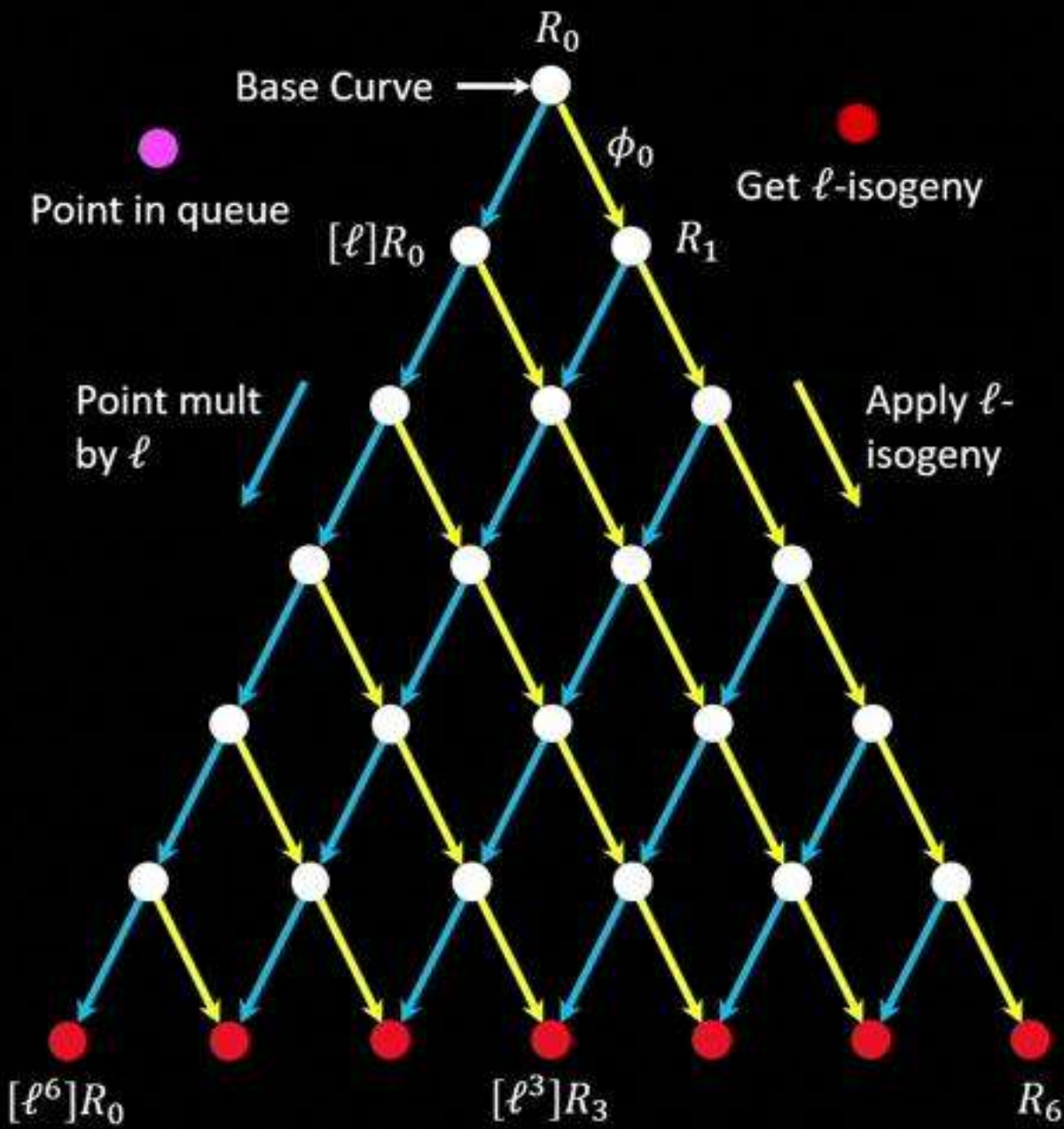
$$E_6 = \phi_5(E_5)$$



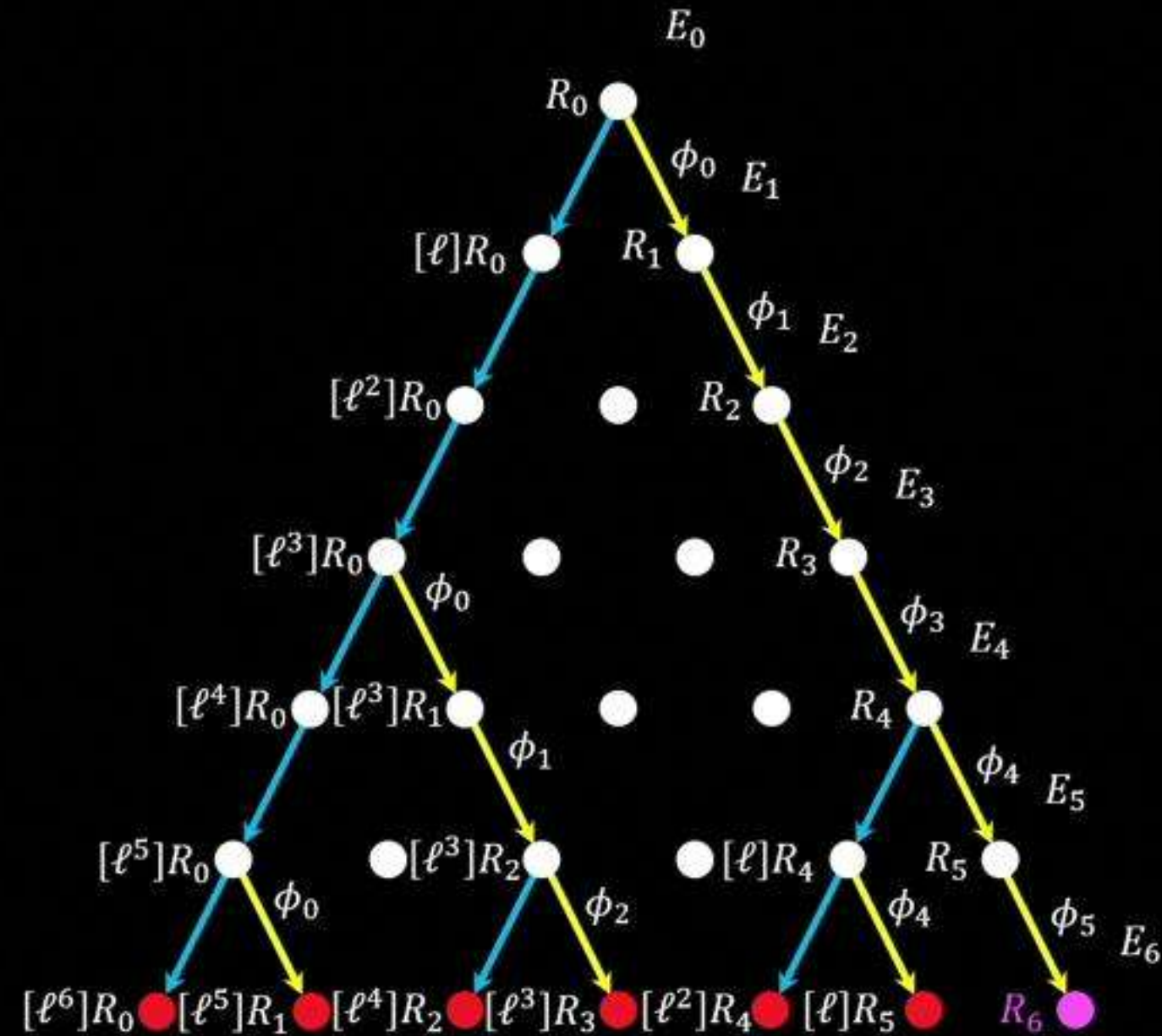


# Large degree isogeny computations

$e = 7$

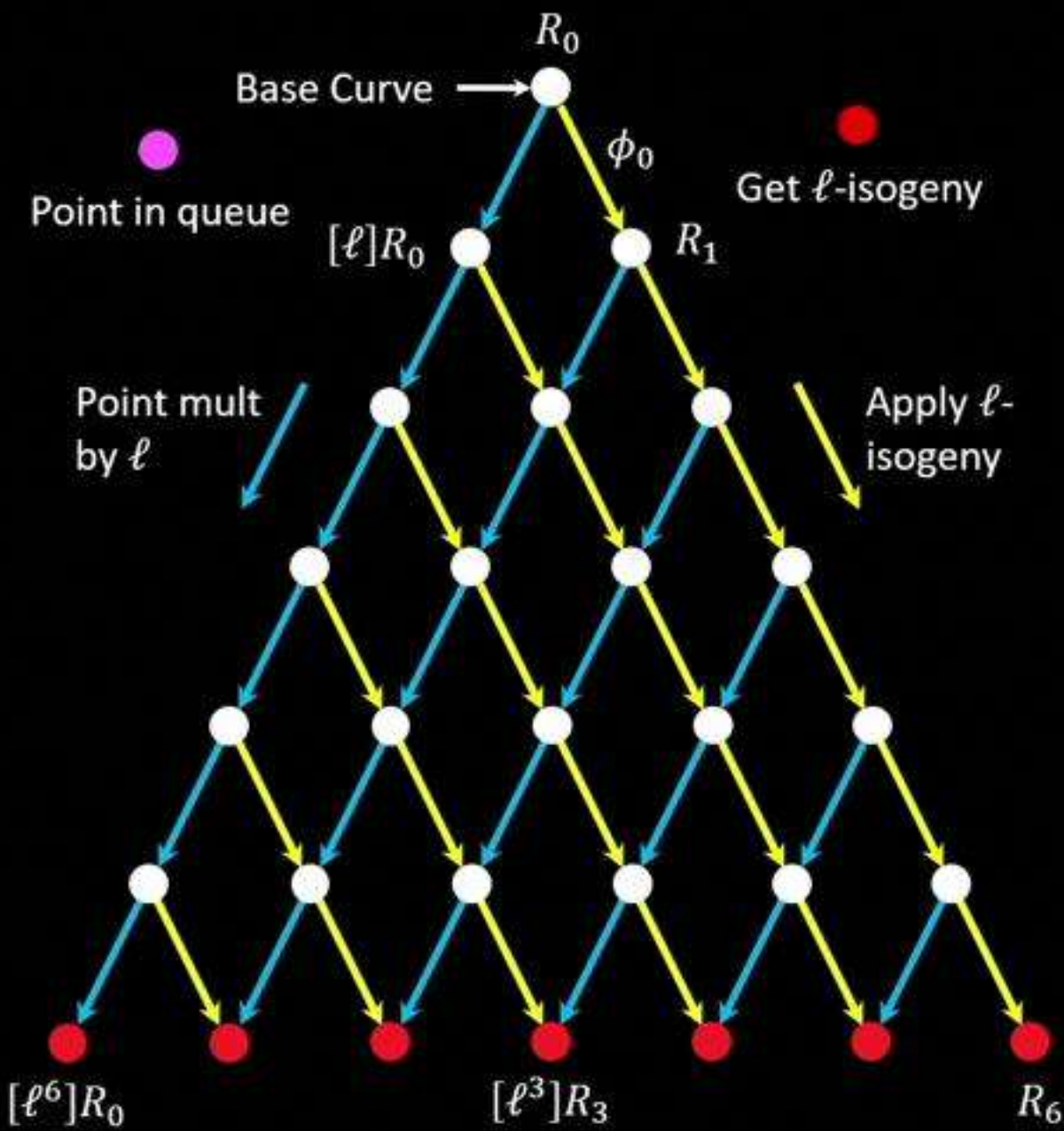


Order of  $R_6$  is  $\ell$



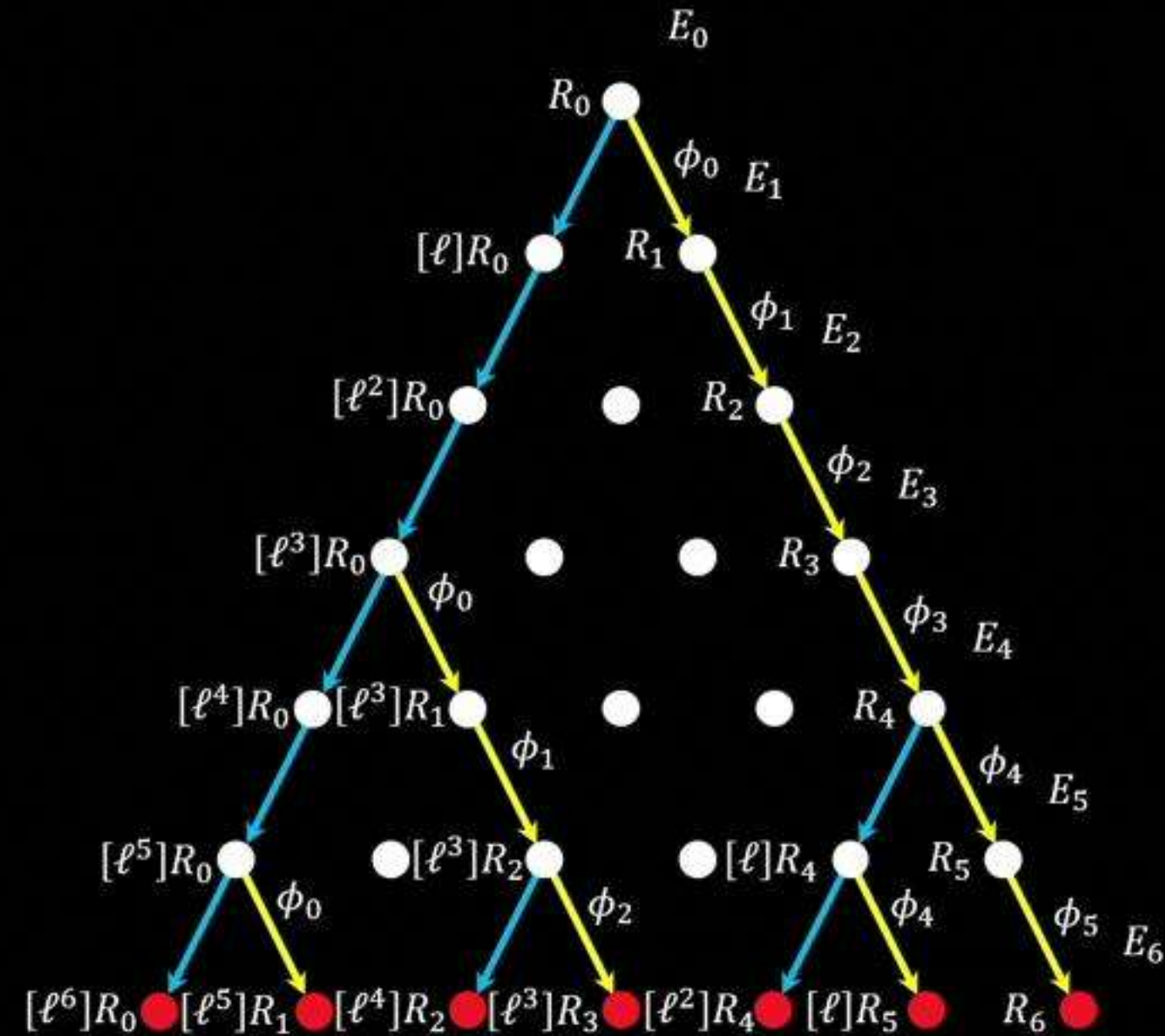
# Large degree isogeny computations

$e = 7$



$$\phi_6 := E_6 / \langle R_6 \rangle$$

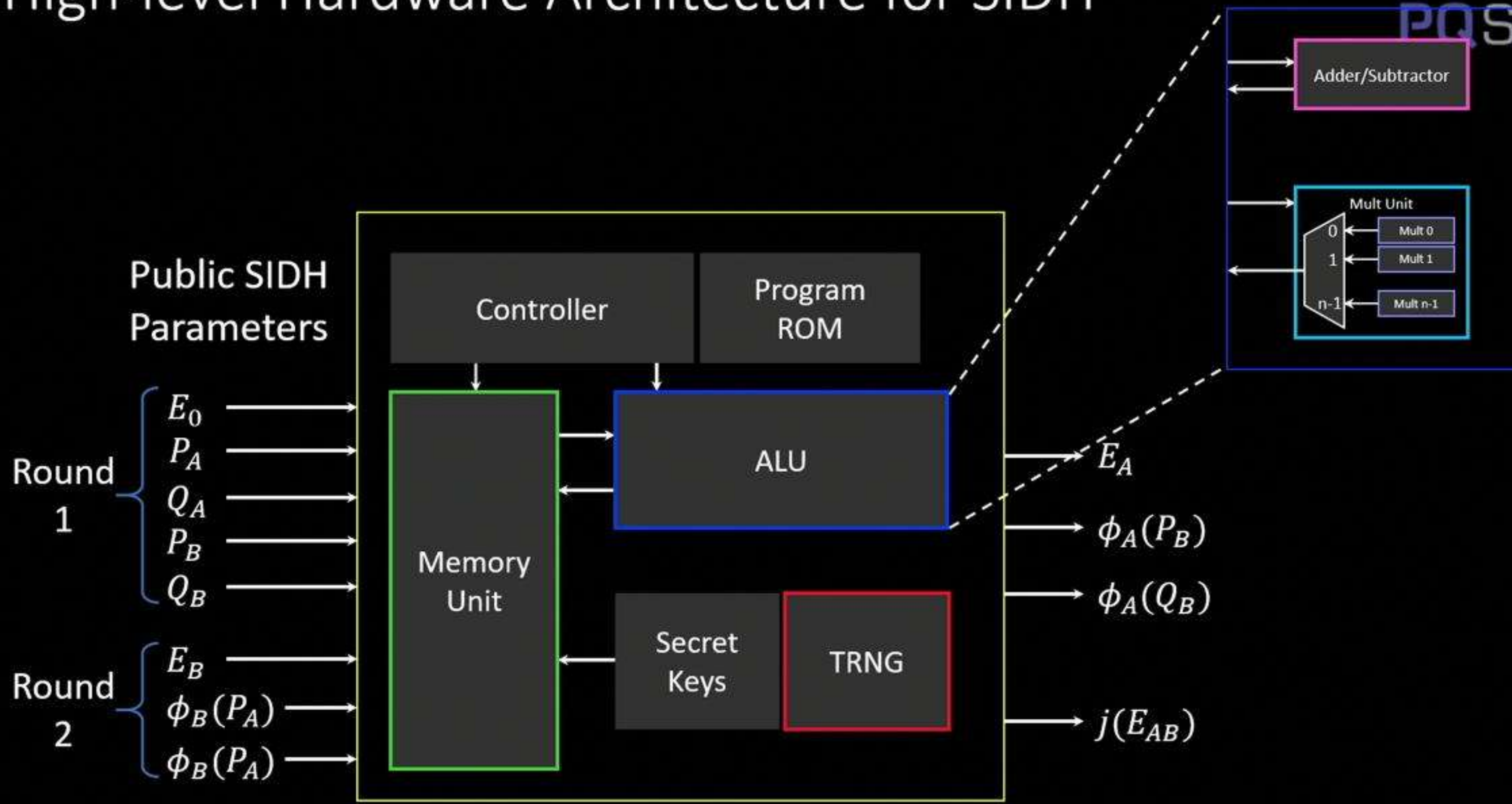
$$E_7 = \phi_6(E_6)$$



# High-level Hardware Architecture for SIDH



PQ Secure

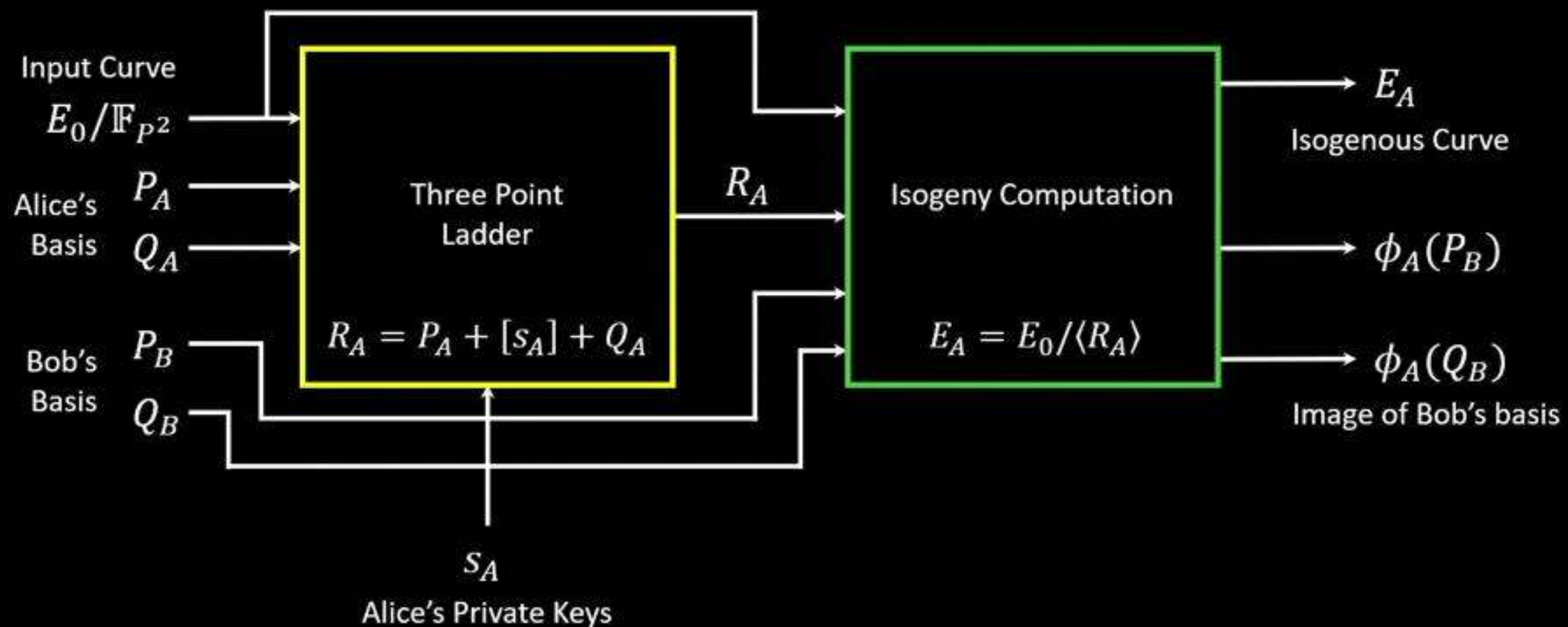


# Fast Kernel Computations

$$R = \ker(\phi) = \langle P + [s]Q \rangle$$

Public SIDH  
Parameters

Ephemeral Public  
Key to Bob



# Arithmetic over $\mathbb{F}_{p^2}$

Each of the  $\mathbb{F}_{p^2}$  arithmetic are built upon a series of  $\mathbb{F}_p$  arithmetic

$\mathbb{F}_{p^2}$	$\mathbb{F}_p$	ops
$a + b =$	$(a_0 + b_0, a_1 + b_1)$	$2A$
$a - b =$	$(a_0 - b_0, a_1 - b_1)$	$2A$
$a \times b =$	$(a_0 \cdot b_0 - a_1 \cdot b_1, (a_0 + a_1) \cdot (b_0 + b_1) - a_0 \cdot b_0 - a_1 \cdot b_1)$	$3M + 5A$
$a^2 =$	$(a_0 + a_1)(a_0 - a_1), 2a_0 a_1)$	$2M + 3A$
$a^{-1} =$	$(a_0(a_0^2 + a_1^2)^{-1}, -a_1(a_0^2 + a_1^2)^{-1})$	$4M + 2A + 1I$

- Field multiplication performs  $C = A \times B \bmod p$
- Choice of modular multiplier is crucial: **Montgomery multiplication**
- **Systolic Montgomery** multiplier
  - PEs process various chunks of the results in **parallel**
  - For SIKE primes  $(2^{e_A} \cdot 3^{e_B} - 1)$ ,  $p = 1 \dots \underbrace{111 \dots 111}_{e_A}$  and  $p' = -p^{-1} = 1 \pmod{2^w}$  where  $w \leq e_A$

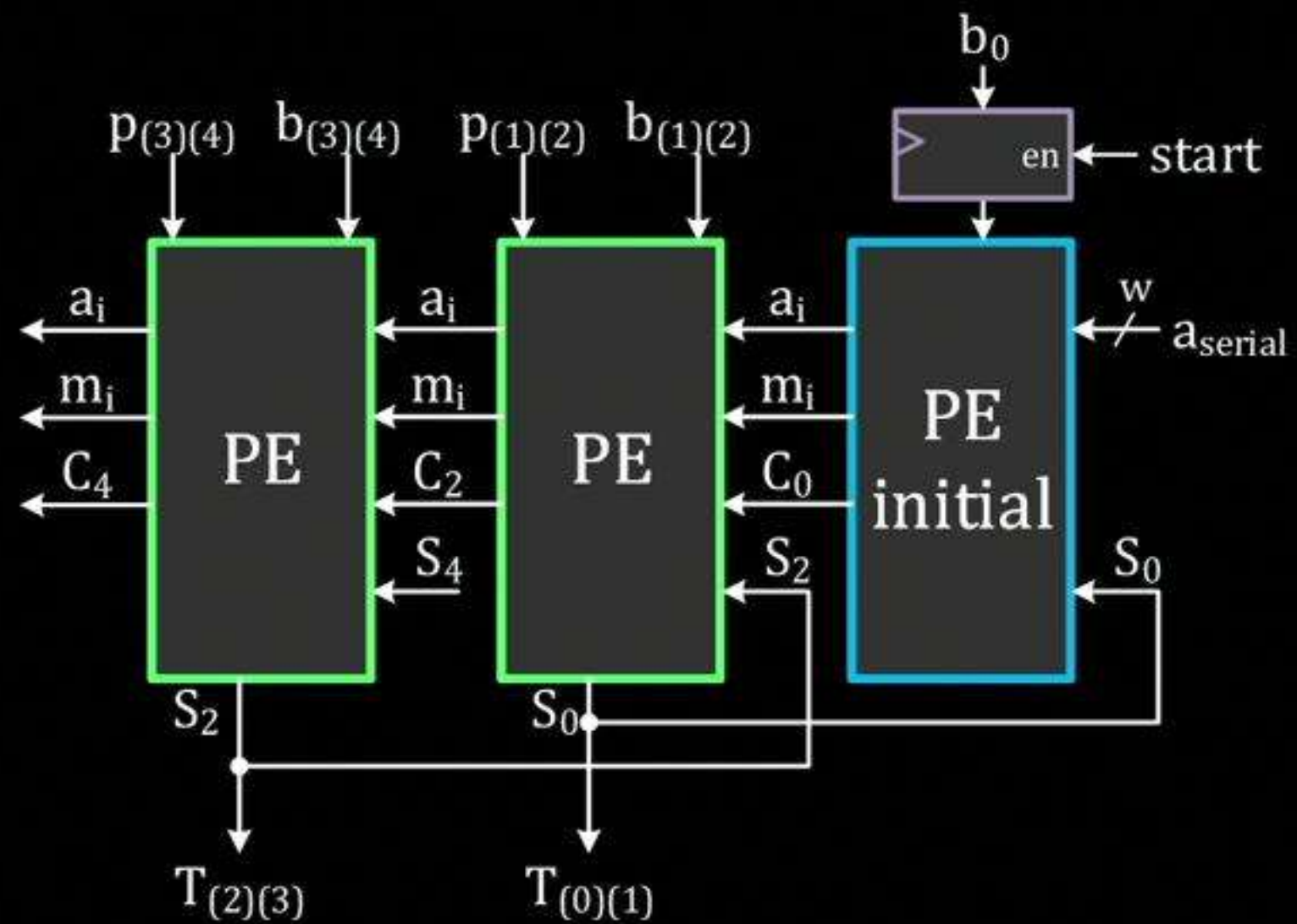
## Coarsely Integrated Operand Scanning (CIOS):

- Alternate between multiplication and reduction
- Shorter Critical Path: 1 Mult + 1 Addition
- More clock cycles: (**4** × Number of words)

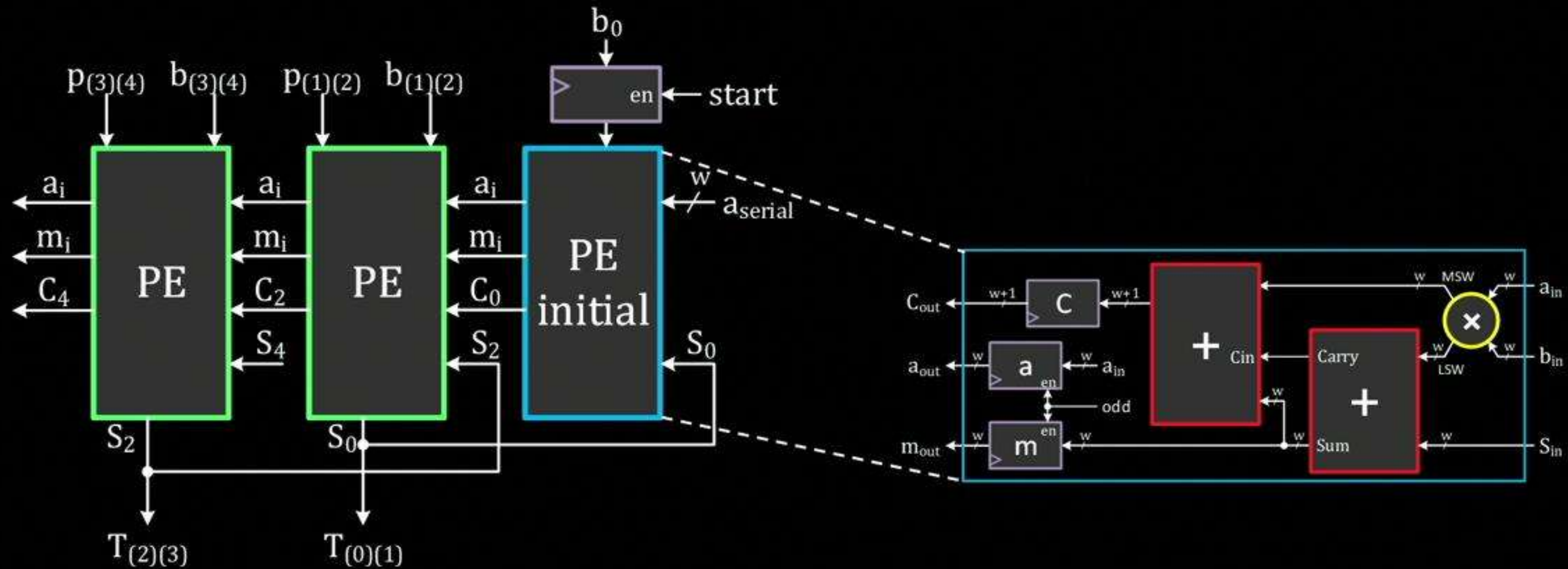
## Finely Integrated Operand Scanning (FIOS):

- Parallelize Multiplication and reduction
- Longer Critical Path: 1 Mult + 2 Additions
- Less clock cycles: (**3** × Number of words)

# FIOS Design (Number of words = 4)

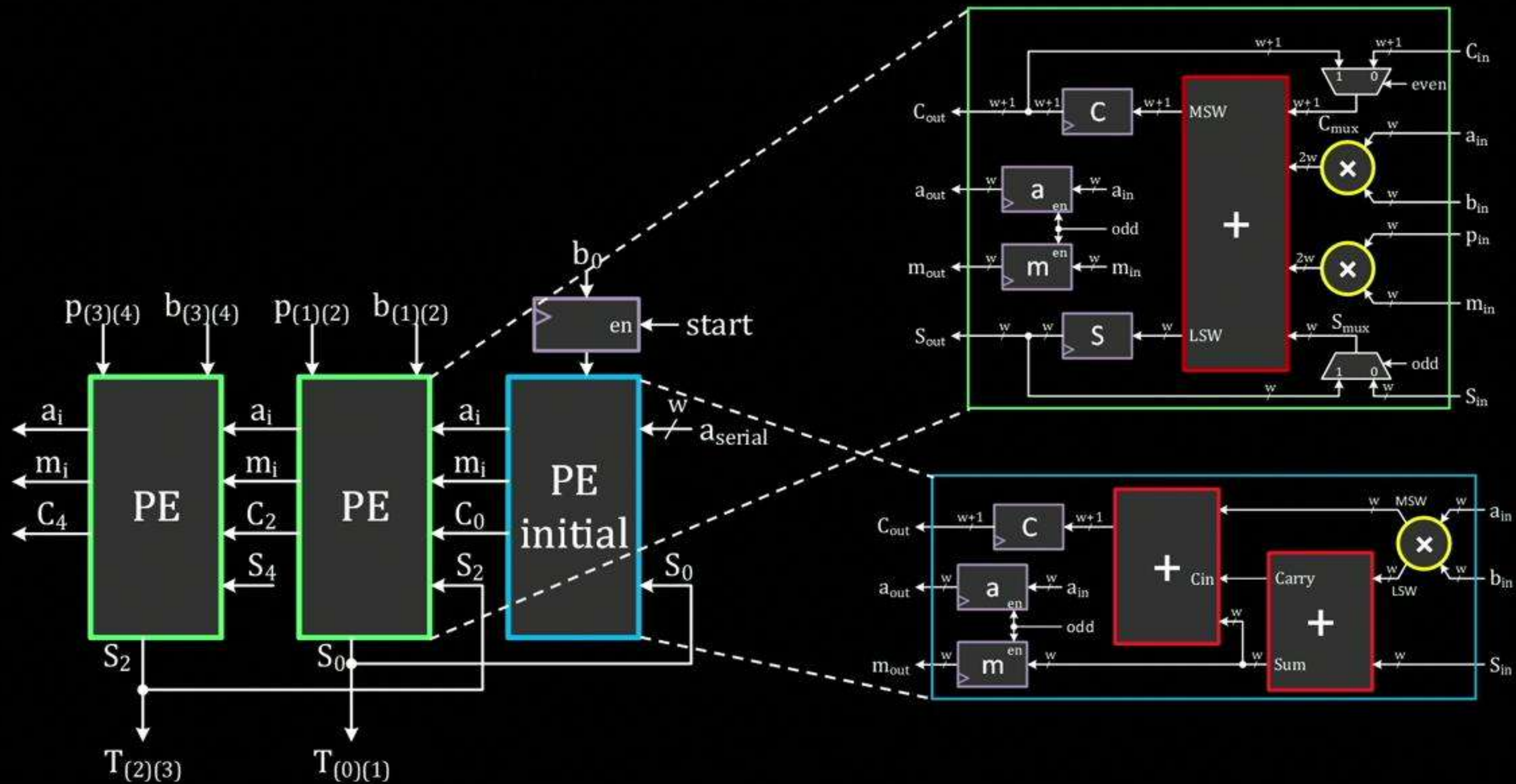


# FIOS Design (Number of words = 4)





# FIOS Design (Number of words = 4)



# SIKE Architecture

## Legend

Public Parameters

Alice's values

Bob's values

## KEY GENERATION (Bob)

Bob's secret key  $s_B$

# SIKE Architecture

## Legend

Public Parameters  
Alice's values  
Bob's values

## KEY GENERATION (Bob)

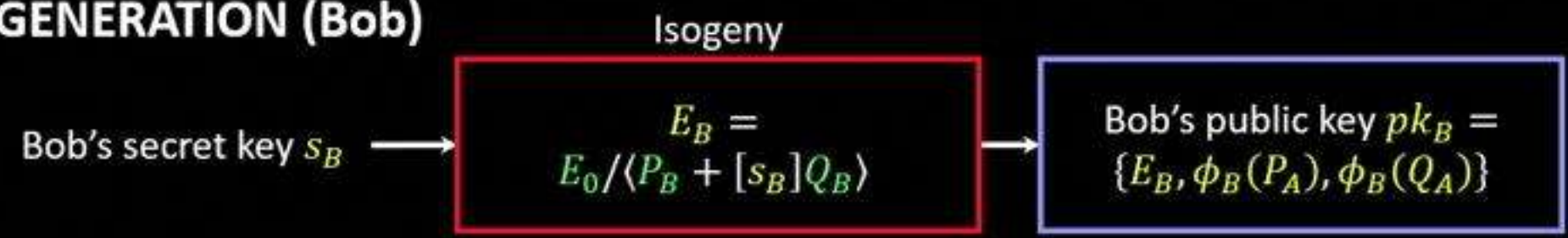
Bob's secret key  $s_B$  →

Isogeny

$$E_B = E_0 / \langle P_B + [s_B]Q_B \rangle$$

# SIKE Architecture

## KEY GENERATION (Bob)



Legend

Public Parameters  
Alice's values  
Bob's values

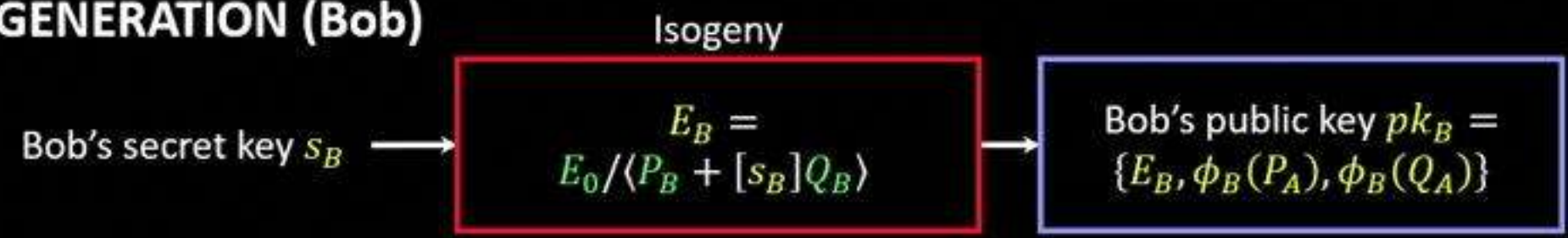
## KEY ENCAPSULATION (Alice)

Alice's secret message  $m$

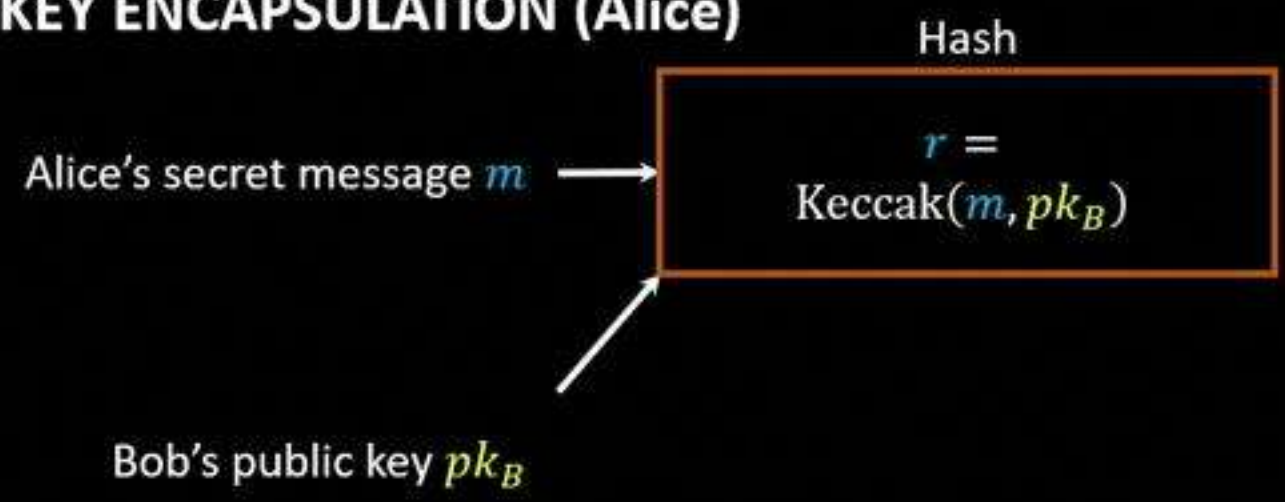
Bob's public key  $pk_B$

# SIKE Architecture

## KEY GENERATION (Bob)



## KEY ENCAPSULATION (Alice)



Legend

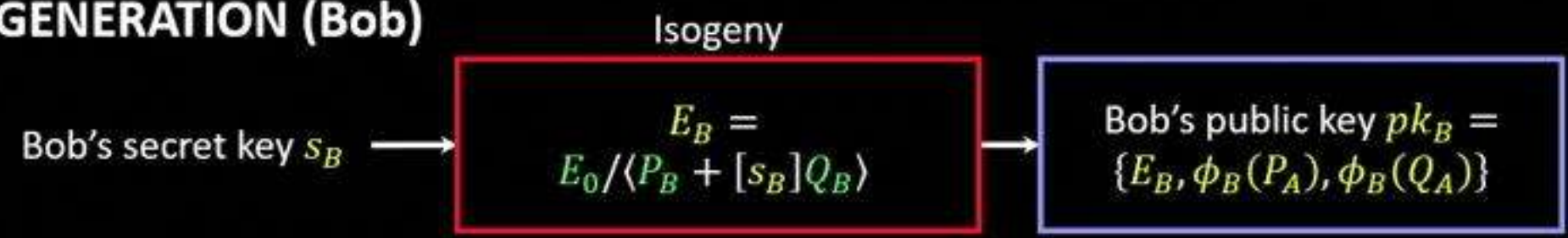
- Public Parameters
- Alice's values
- Bob's values

# SIKE Architecture

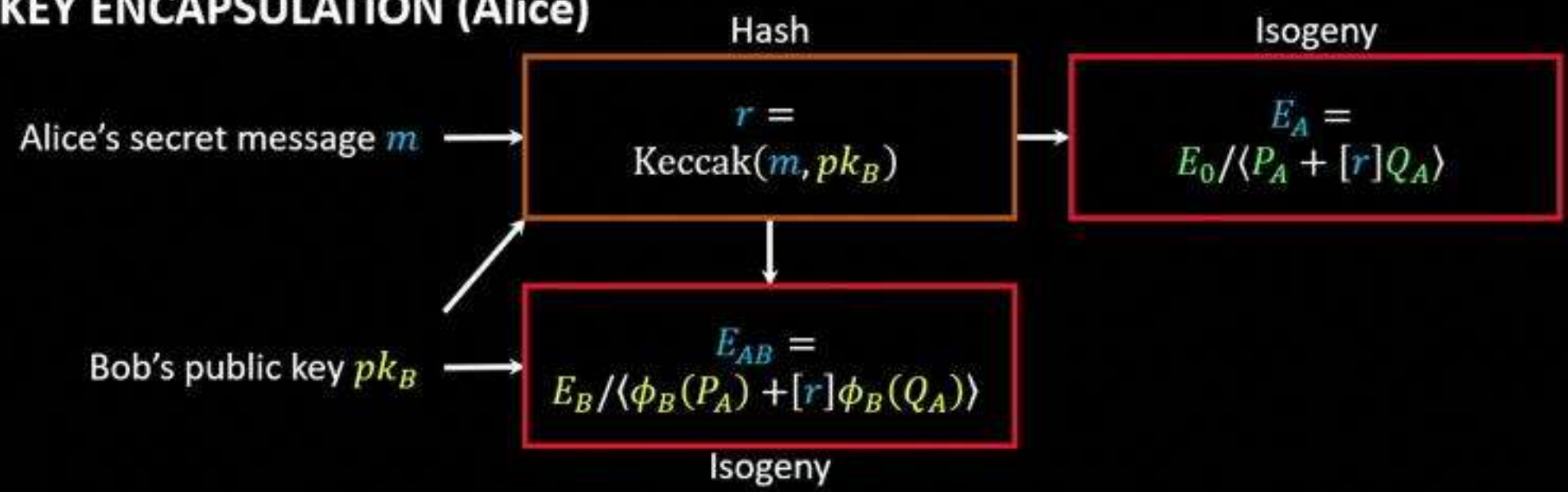
## Legend

- Public Parameters
- Alice's values
- Bob's values

### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)

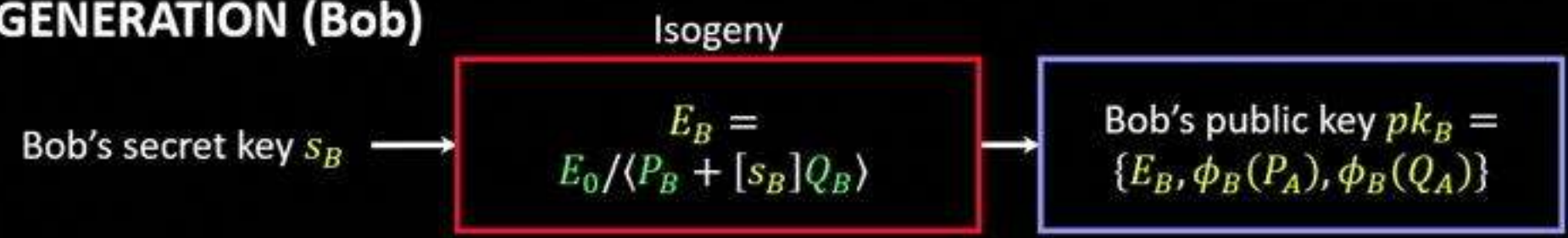


# SIKE Architecture

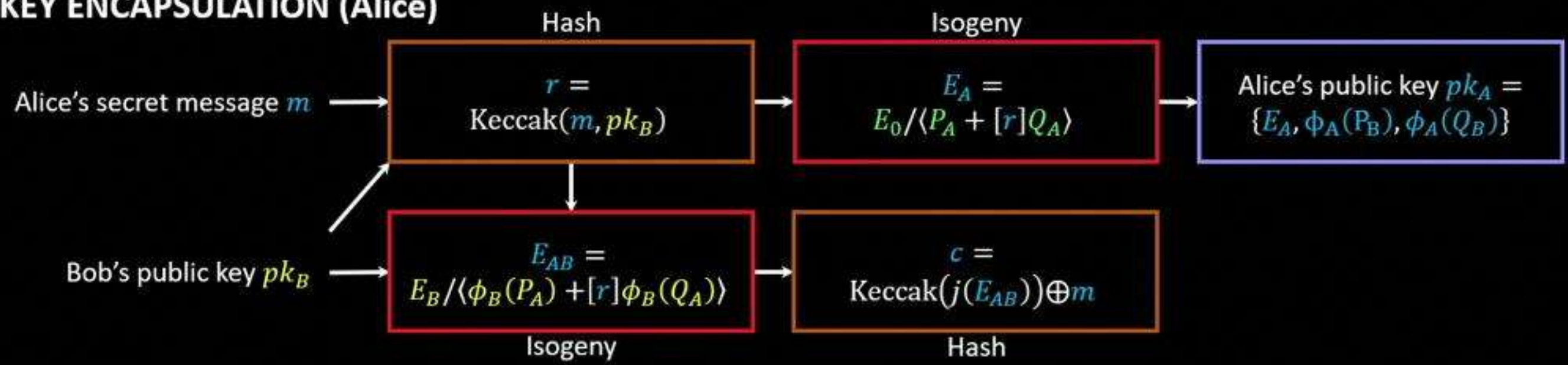
## Legend

- Public Parameters
- Alice's values
- Bob's values

### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)

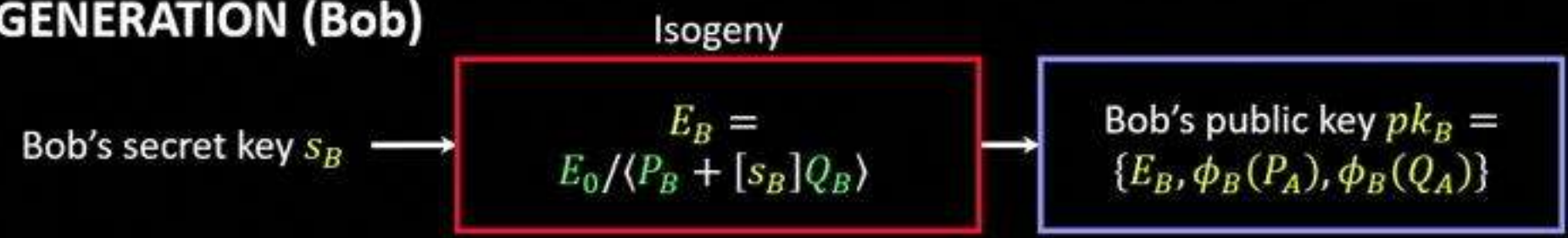


# SIKE Architecture

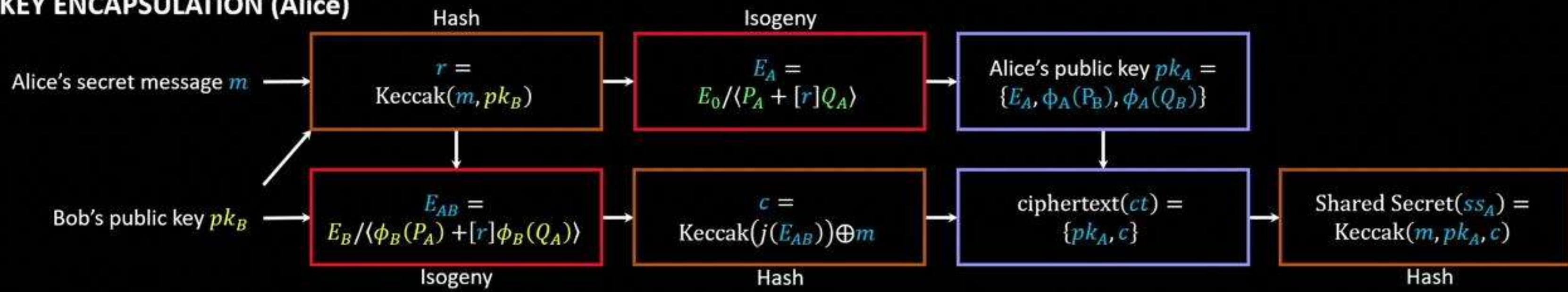
## Legend

Public Parameters  
 Alice's values  
 Bob's values

### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)





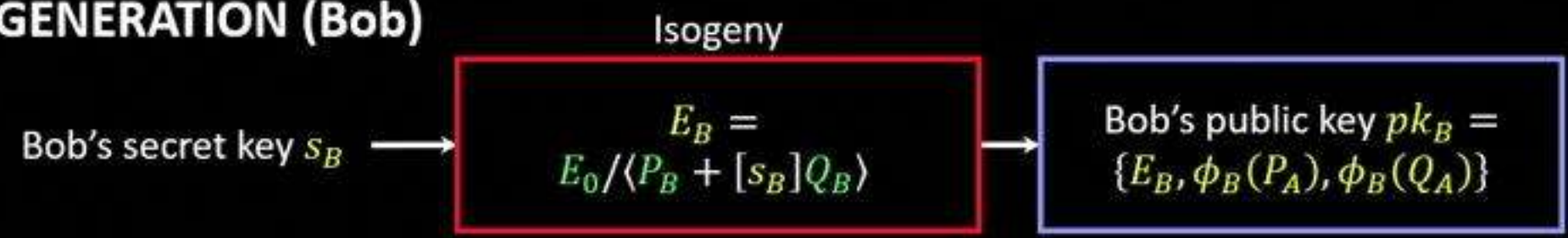
# SIKE Architecture



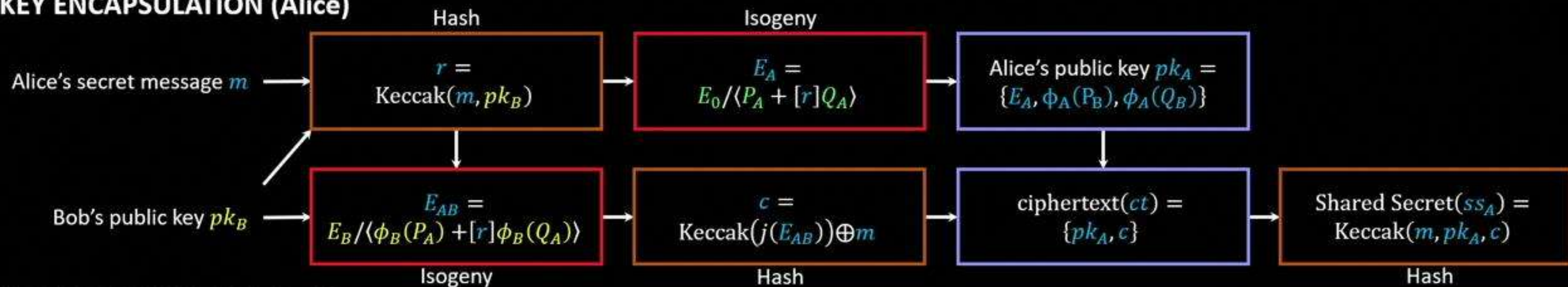
## Legend

Public Parameters  
 Alice's values  
 Bob's values

### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)



### KEY DECAPSULATION (Bob)

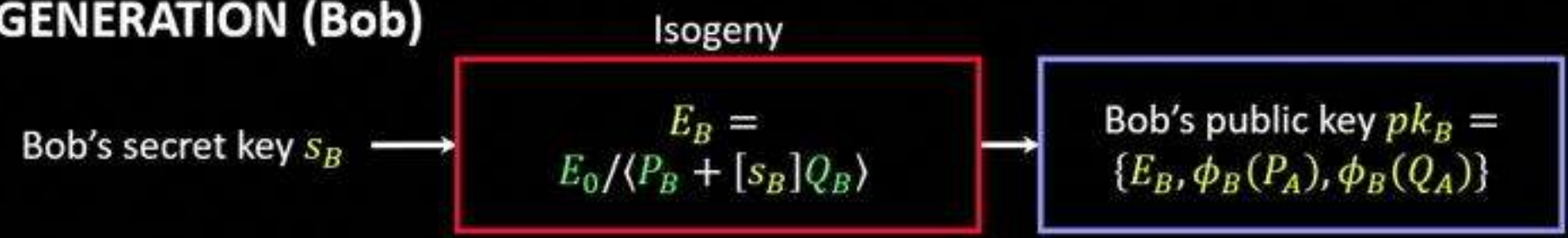
ciphertext( $ct$ )

# SIKE Architecture

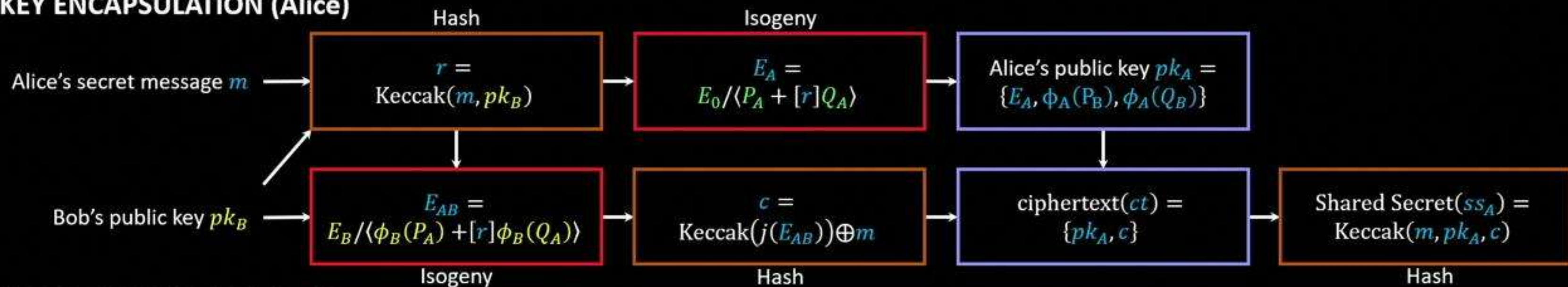
## Legend

Public Parameters  
 Alice's values  
 Bob's values

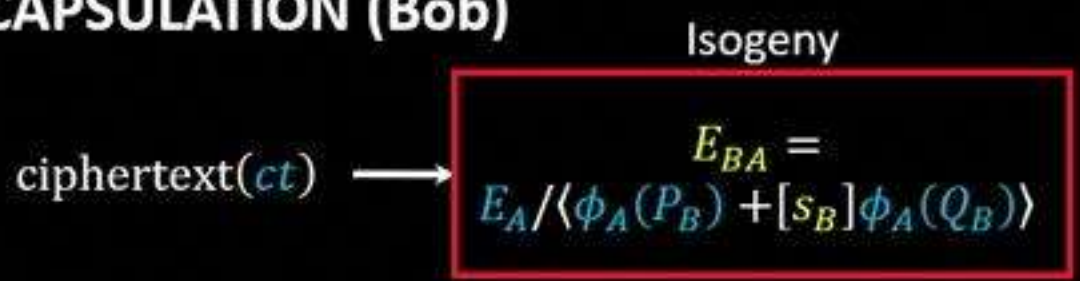
### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)



### KEY DECAPSULATION (Bob)

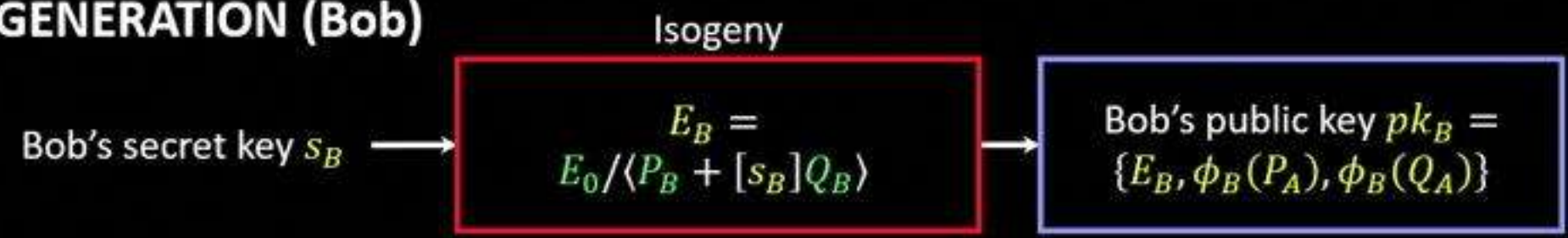


# SIKE Architecture

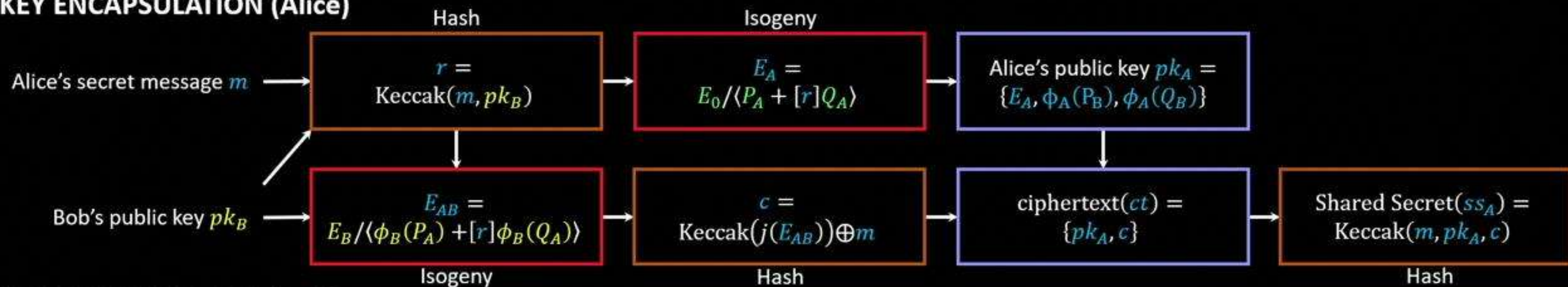
## Legend

Public Parameters  
 Alice's values  
 Bob's values

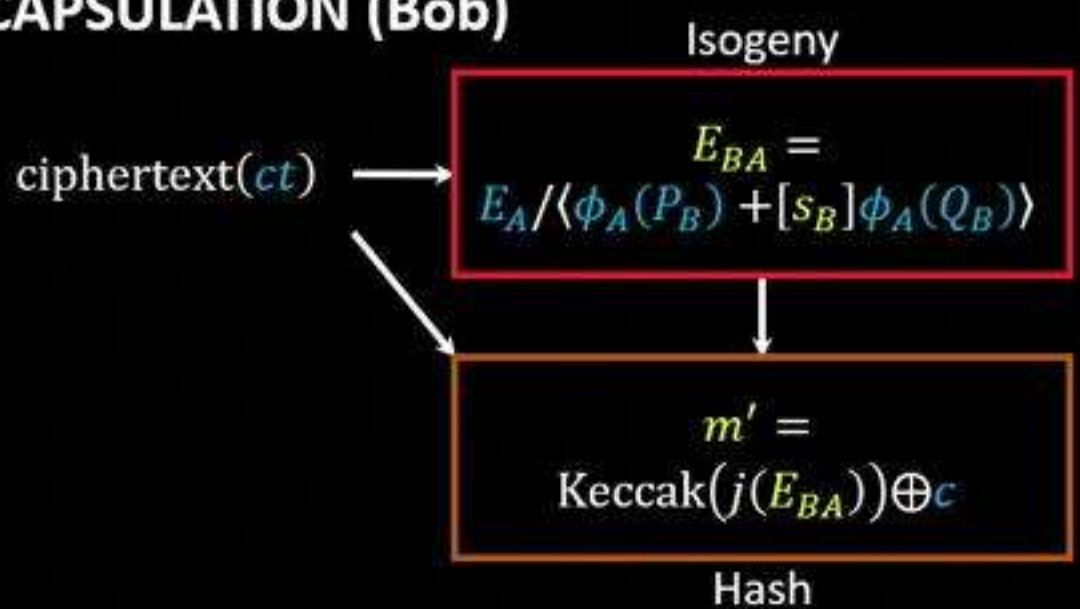
### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)



### KEY DECAPSULATION (Bob)

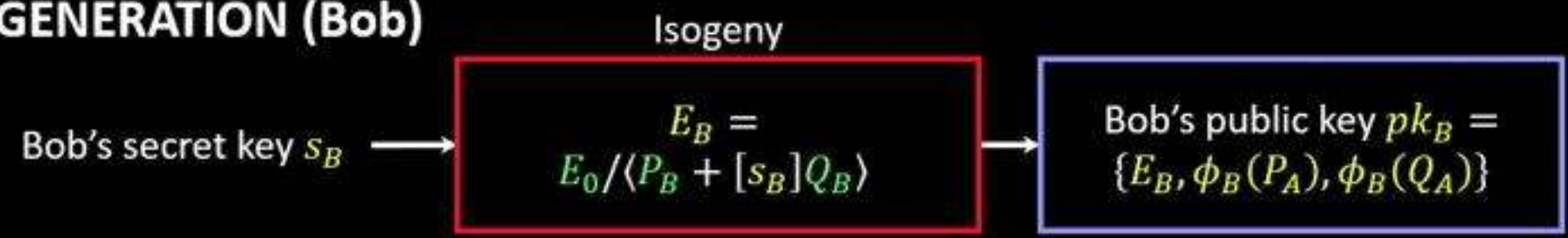


# SIKE Architecture

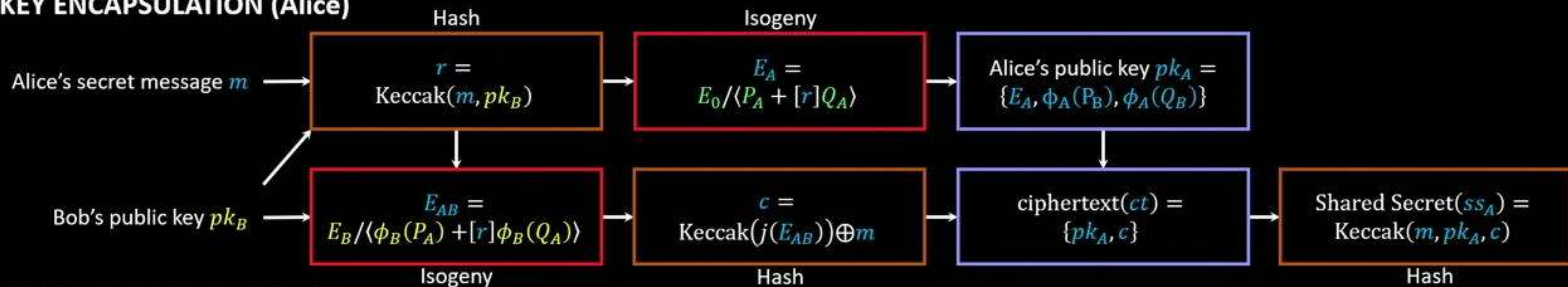
## Legend

Public Parameters  
 Alice's values  
 Bob's values

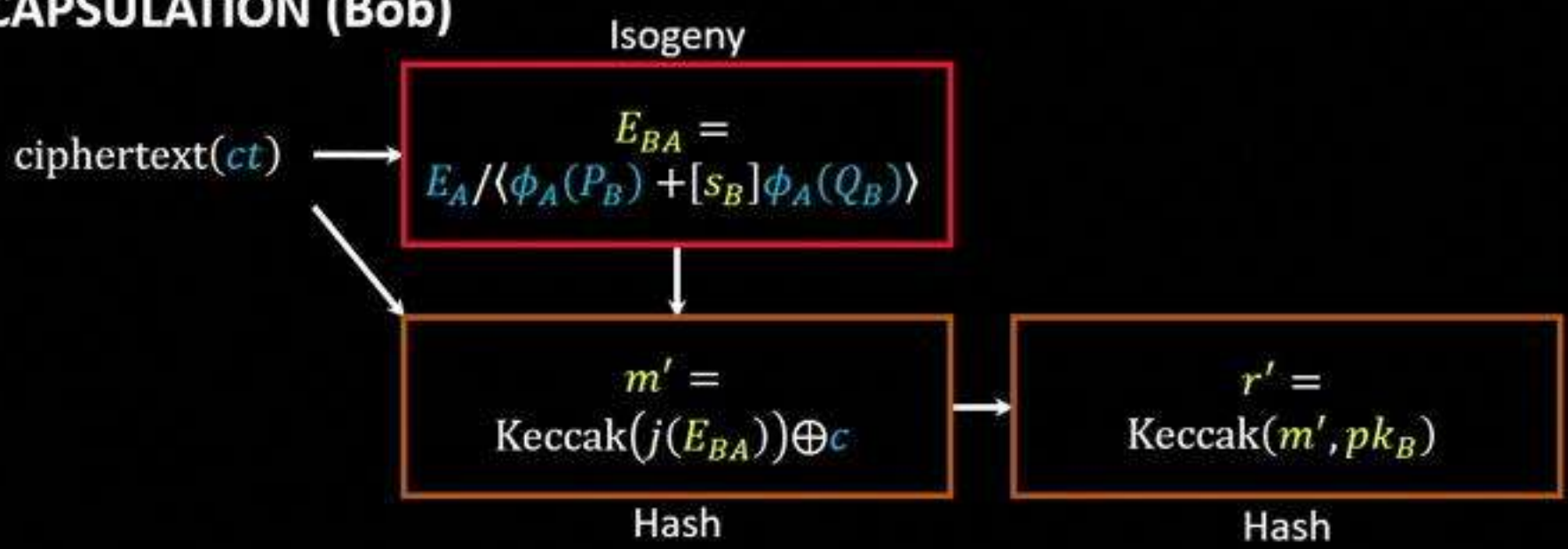
### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)



### KEY DECAPSULATION (Bob)



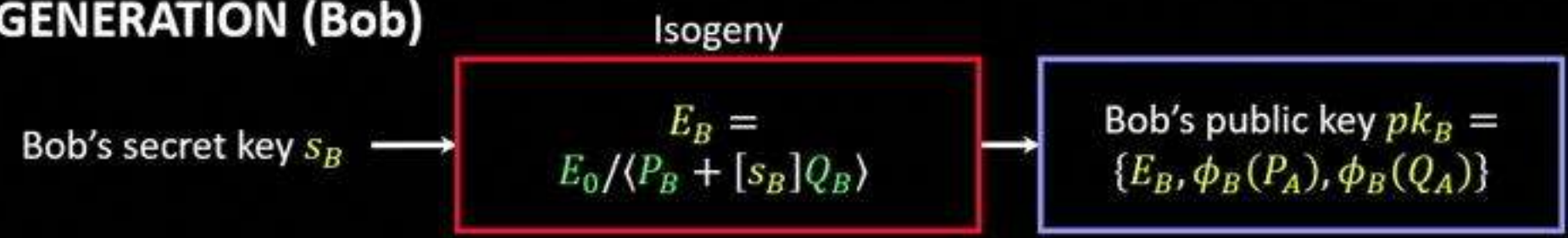
# SIKE Architecture



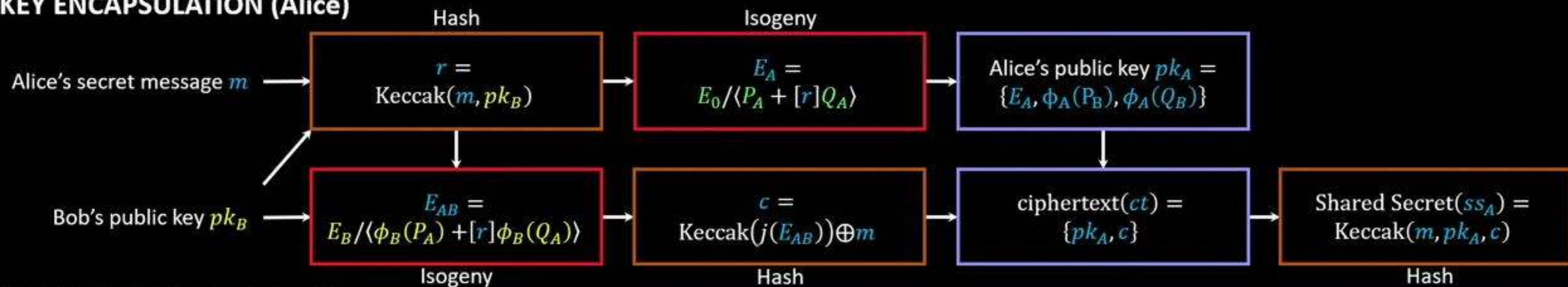
## Legend

Public Parameters  
 Alice's values  
 Bob's values

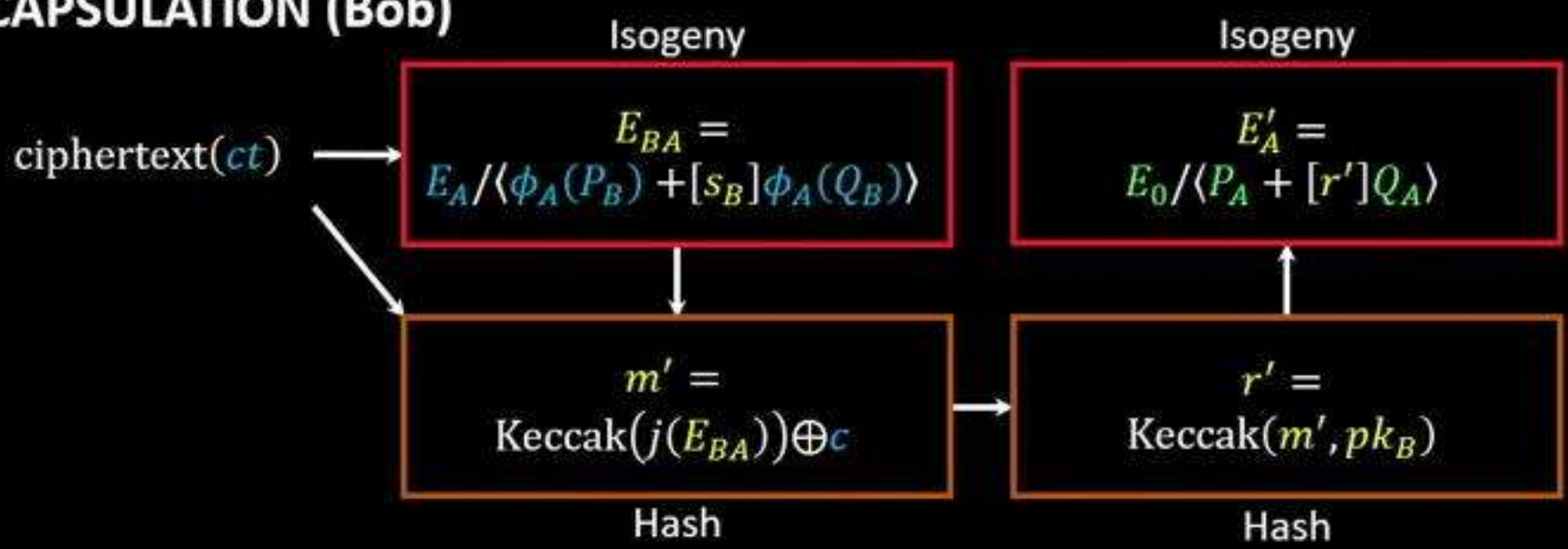
### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)



### KEY DECAPSULATION (Bob)

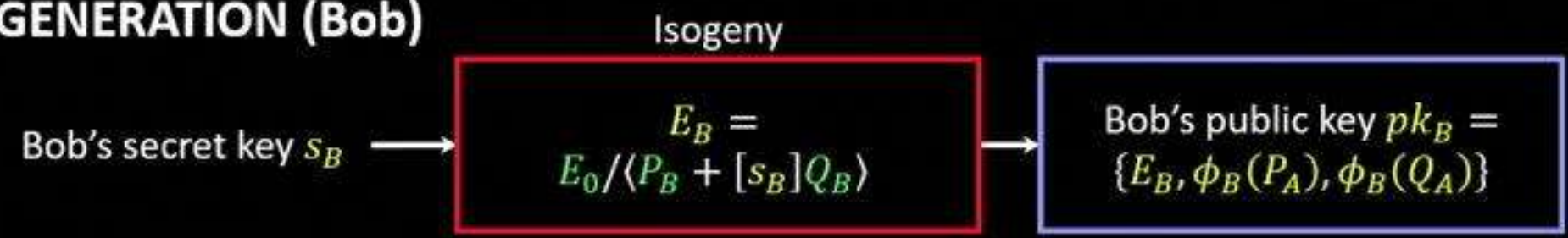


# SIKE Architecture

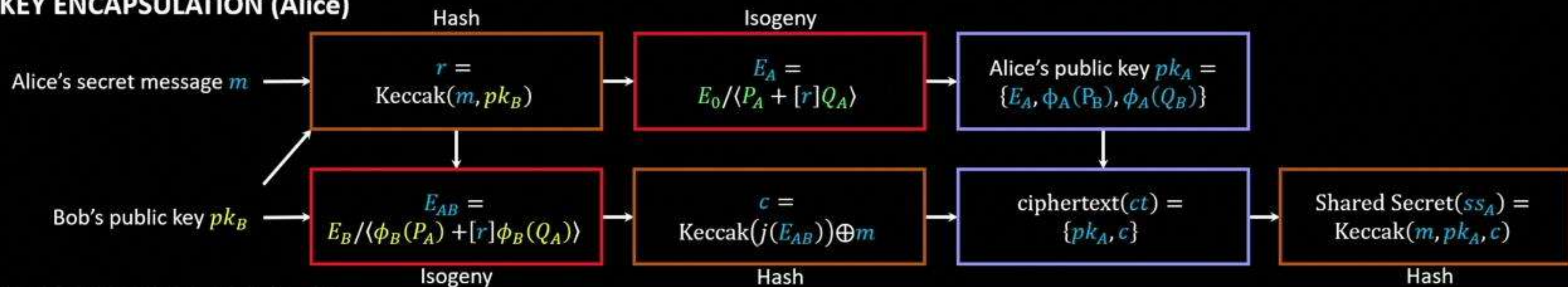
## Legend

Public Parameters  
 Alice's values  
 Bob's values

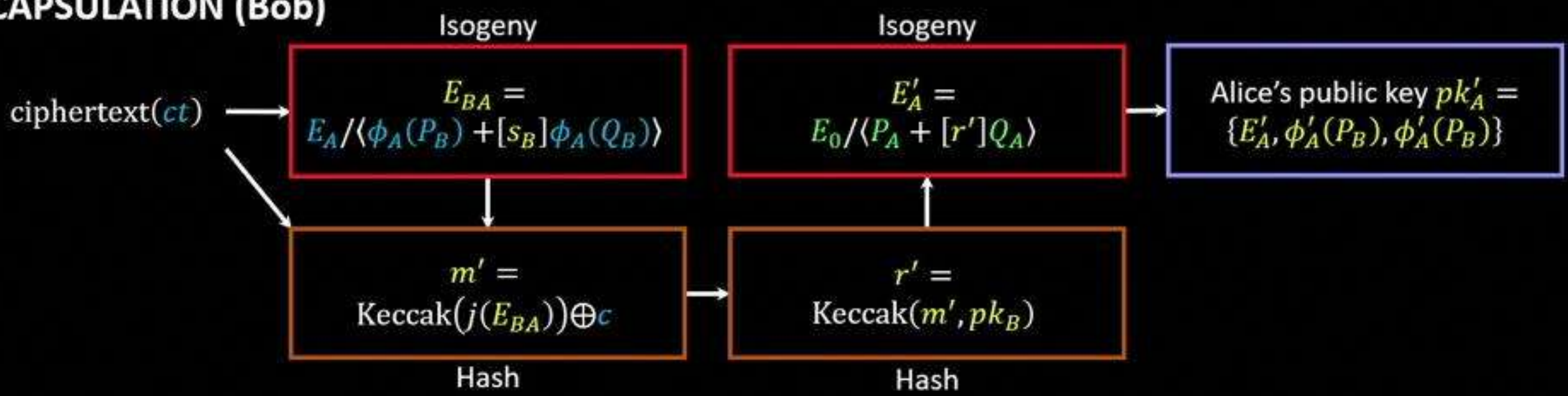
### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)



### KEY DECAPSULATION (Bob)

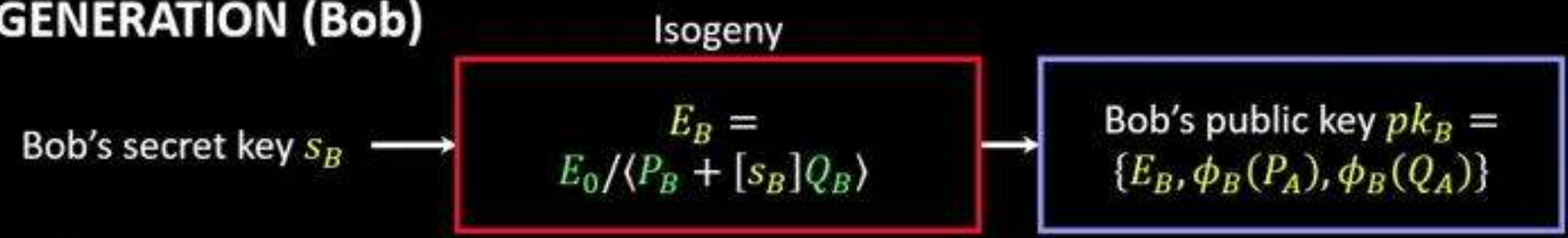


# SIKE Architecture

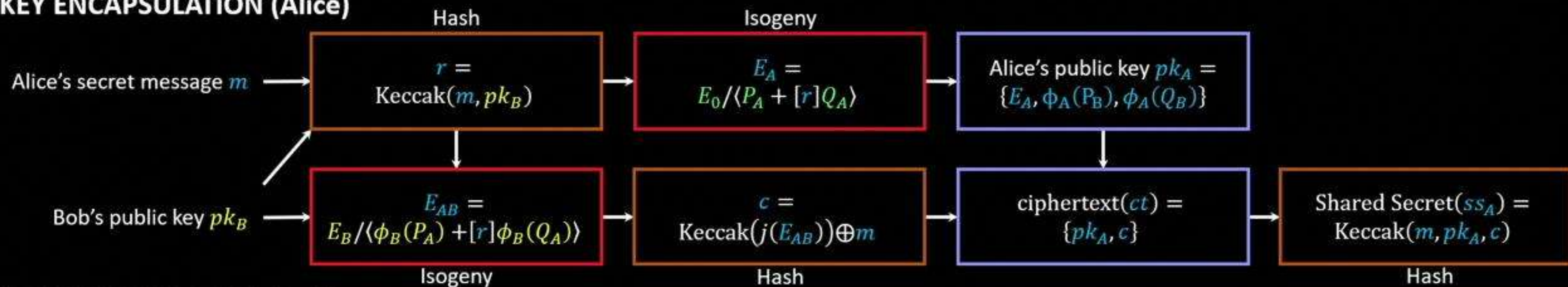
## Legend

Public Parameters  
 Alice's values  
 Bob's values

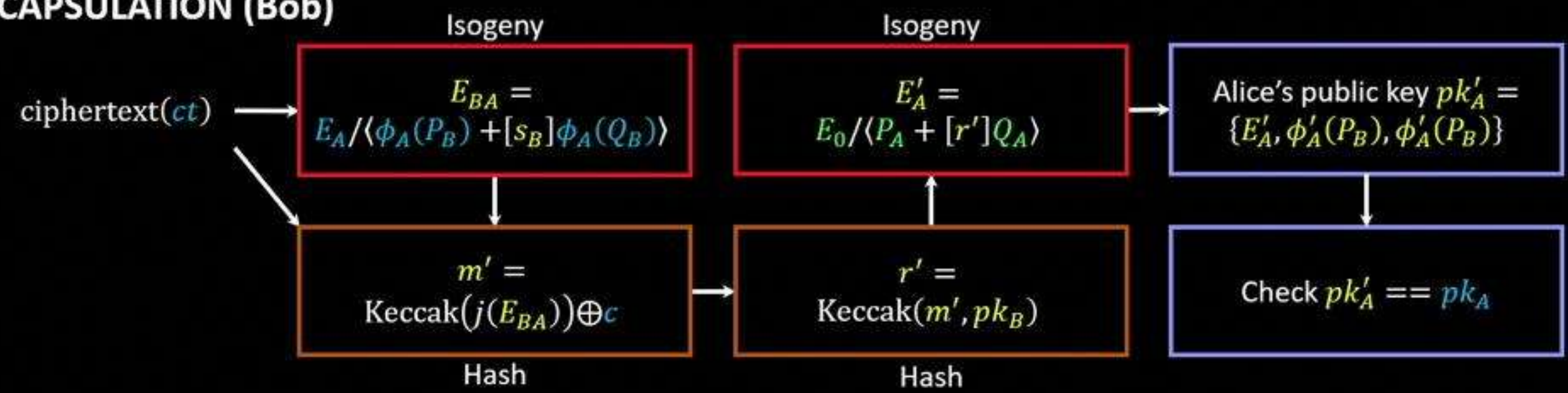
### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)



### KEY DECAPSULATION (Bob)

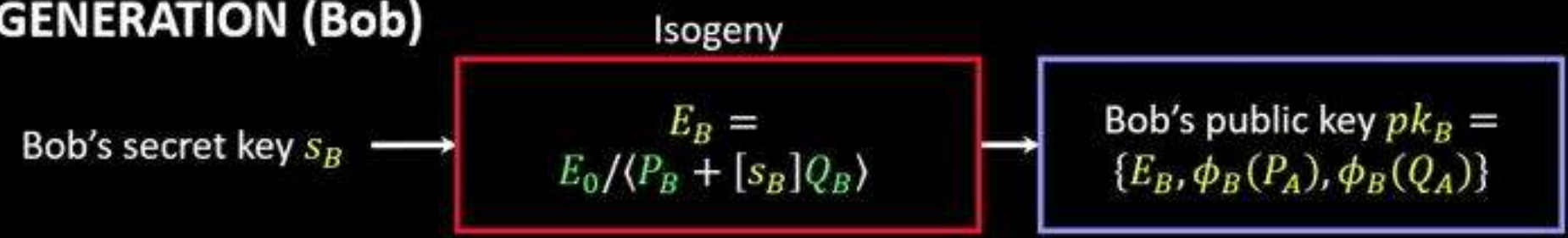


# SIKE Architecture

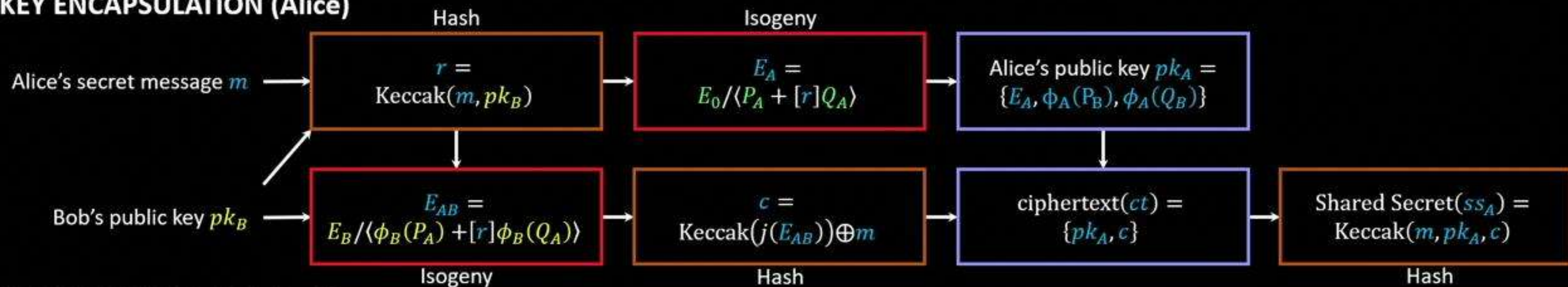
## Legend

Public Parameters  
 Alice's values  
 Bob's values

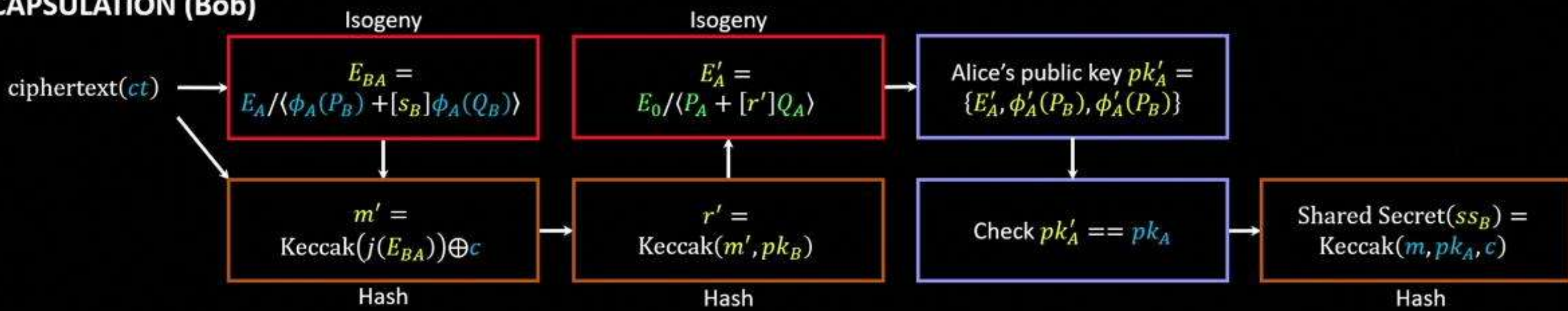
### KEY GENERATION (Bob)



### KEY ENCAPSULATION (Alice)

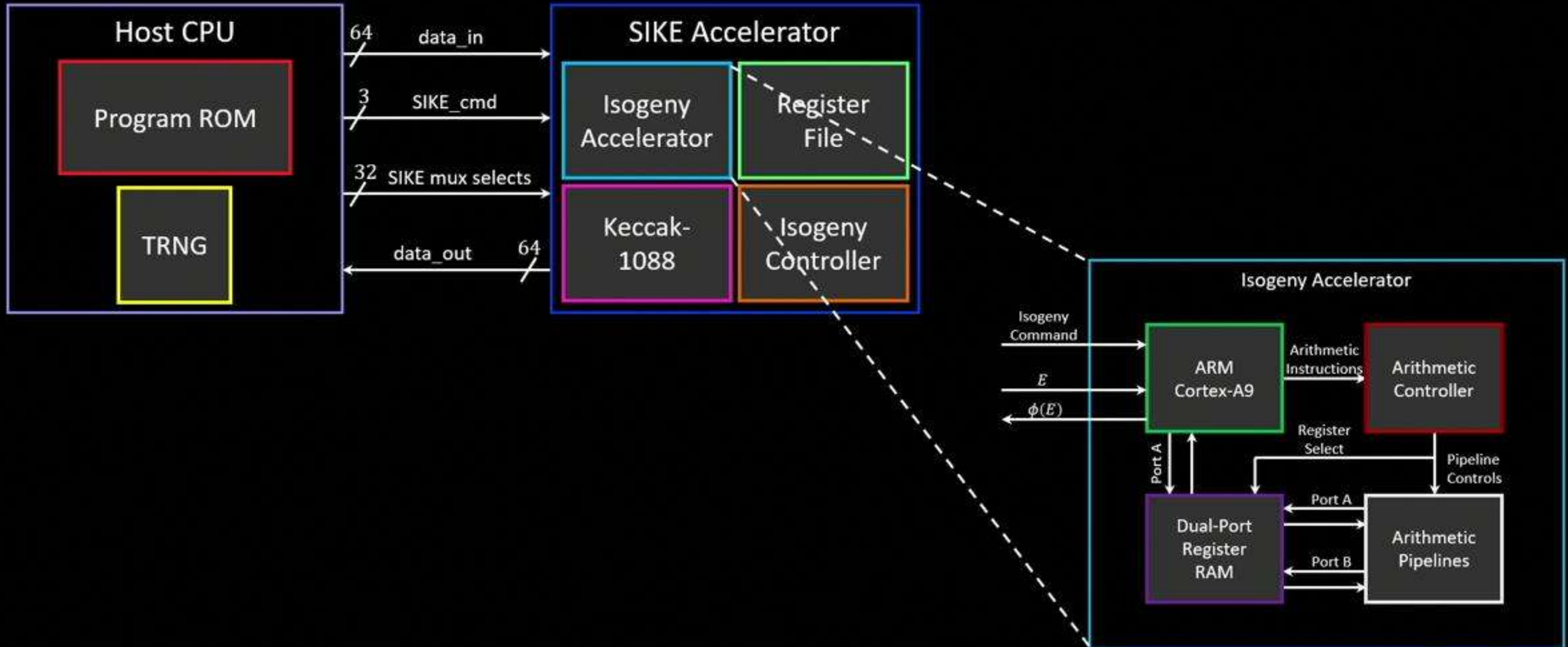


### KEY DECAPSULATION (Bob)





The host initializes any isogeny inputs  $x(P), x(Q), x(Q - P)$  and key  $k$



- Total number of  $\mathbb{F}_p$  arithmetic operations in SIKEp503

$\mathbb{F}_p$	Keygen	Encapsulation	Decapsulation
Addition	31,882	43,127	51,620
Multiplication	40,107	64,372	69,550
Inversion	1	3	3

## NIST-Round 1 Submission: Koziel and Azarderakhsh

Xilinx Virtex 7 FPGA

NIST	SIKE	Area					Freq	Time (ms)			
Level	Prime	#FFs	LUTs	#Slices	DSPs	BRAMs	(MHz)	KeyGen	Encaps	Decaps	Total (E+D)
5 (used to be 3)	SIKEp751	51,914	44,822	16,752	376	56	198	9.08	16.27	17.08	<b>33.35</b>

# SIKE in FPGA Improved

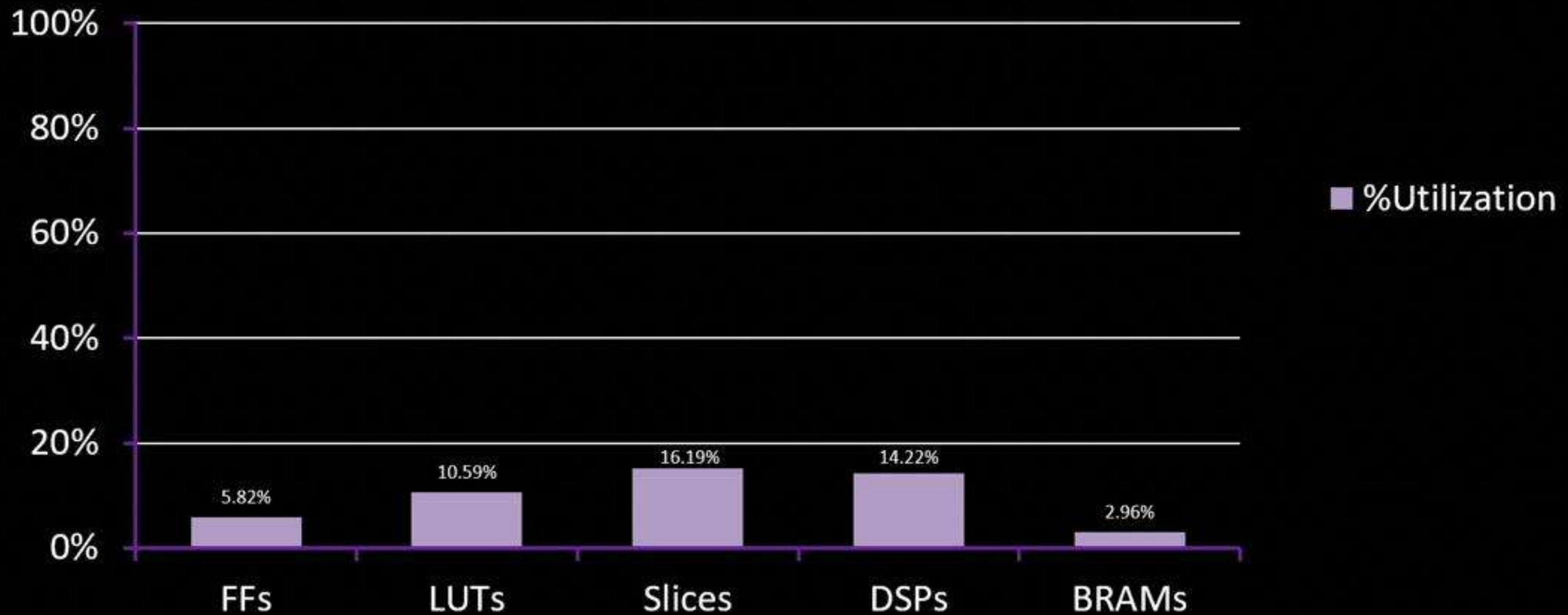
eprint: Koziel, Azarderakhsh, Kermani, El Khatib, Ackie

## Xilinx Virtex 7 FPGA

NIST	SIKE	Area					Freq	Time (ms)			
Level	Prime	#FFs	LUTs	#Slices	DSPs	BRAMs	(MHz)	KeyGen	Encaps	Decaps	Total (E+D)
2	SIKEp503	26,971	25,094	9,514	264	34	171	3.74	7.07	6.6	13.6
5	SIKEp751	50,390	45,893	17,530	512	43	167.4	7.42	13	13.9	26.9

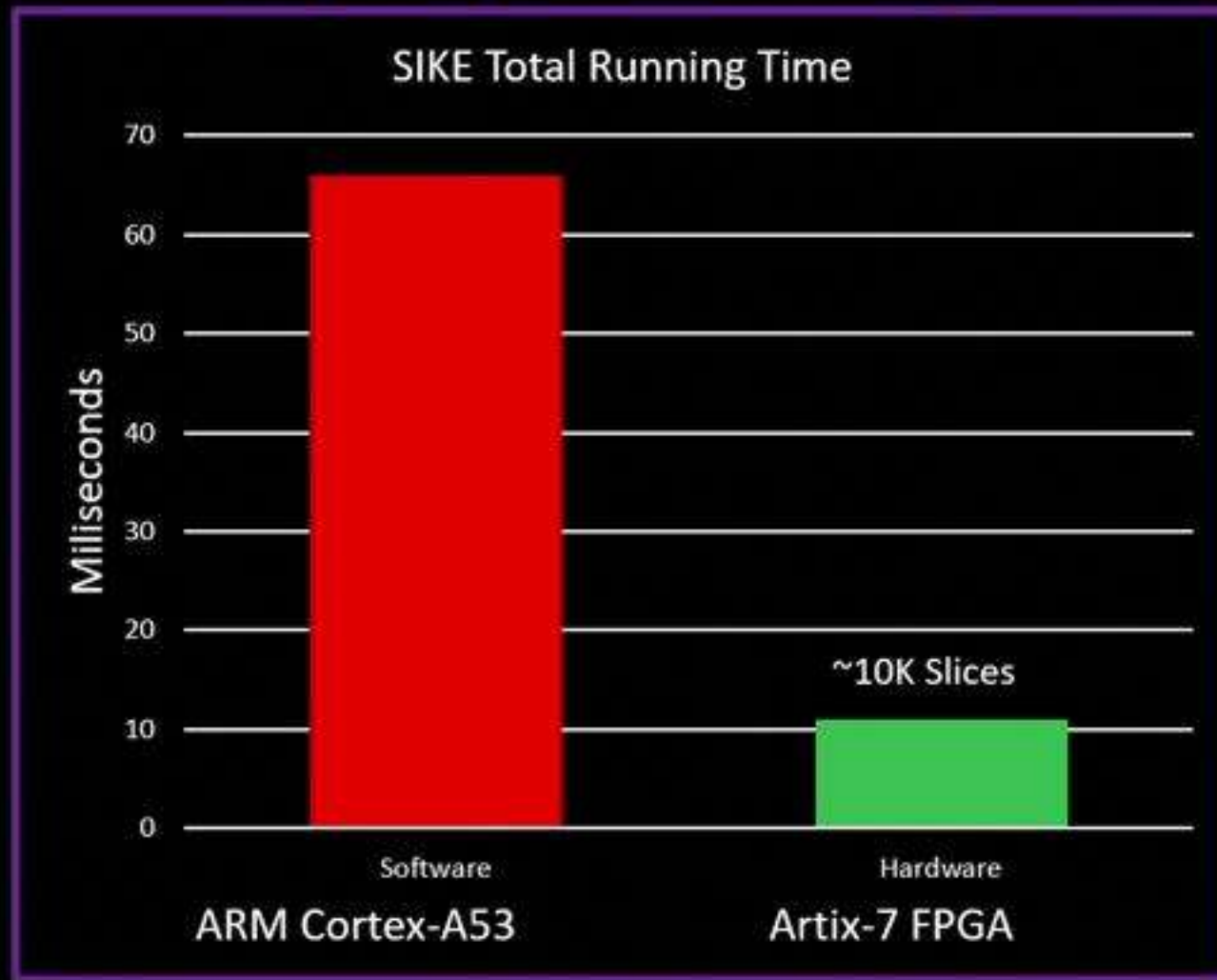
# SIKE in FPGA Area Results

- Area distribution of NIST level 5 SIKEp751 on Virtex-7 FPGA xc7vx690tffg1157-3

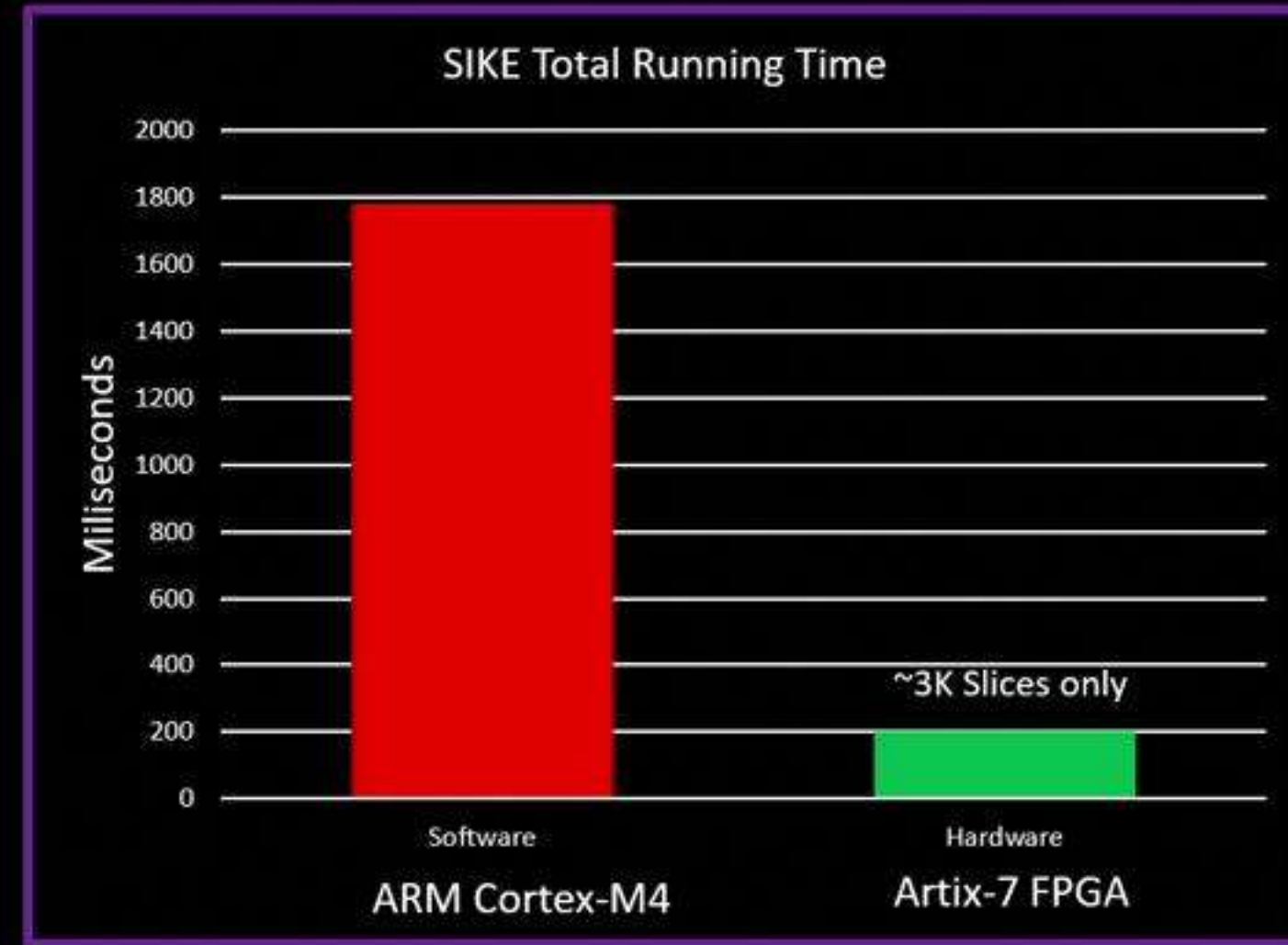


# SIKE: Results for NIST level 1

Target: High Performance Edge



Target: Resource-constrained IoT



- The post-quantum landscape is uncharted territory:
  - The smallest scheme is the slowest, and the fastest scheme is the largest.
  - Compare with traditional cryptography, where the fastest scheme (ECC) is also the smallest.
- This situation introduces a new set of tradeoffs.
  - SIKE's advantages will become **more** pronounced over time.
  - SIKE's disadvantages will become **less** pronounced over time.
- Why **not** CSIDH?
  - CSIDH has sub-exponential quantum security, compared to SIDH/SIKE which has exponential quantum security.
  - Over time, CSIDH becomes **less** attractive compared to SIKE.

# The future of SIKE: Computational Costs

- Hardware gets faster over time.
- Software also gets faster over time.
- The above happens naturally, without effort or expenditure.
- An across-the-board performance increase **reduces** the performance penalty of SIKE (in absolute terms).
- We can also spend more money for **faster** hardware.
- Certain expenditures (e.g. **hardware acceleration**) provide good value per unit cost.



# The future of SIKE: Computational Costs

- As hardware and software gets faster, **attacks get faster**.
- Faster attacks require larger keys to counteract.
- An across-the-board key size increase **enlarges** the communication cost benefits of SIKE (in absolute terms).
- Variance in communication channels is much higher than variance in cycle counts. SIKE **already wins** today on desktop browsers when including variance.

Thank you!  
Questions?