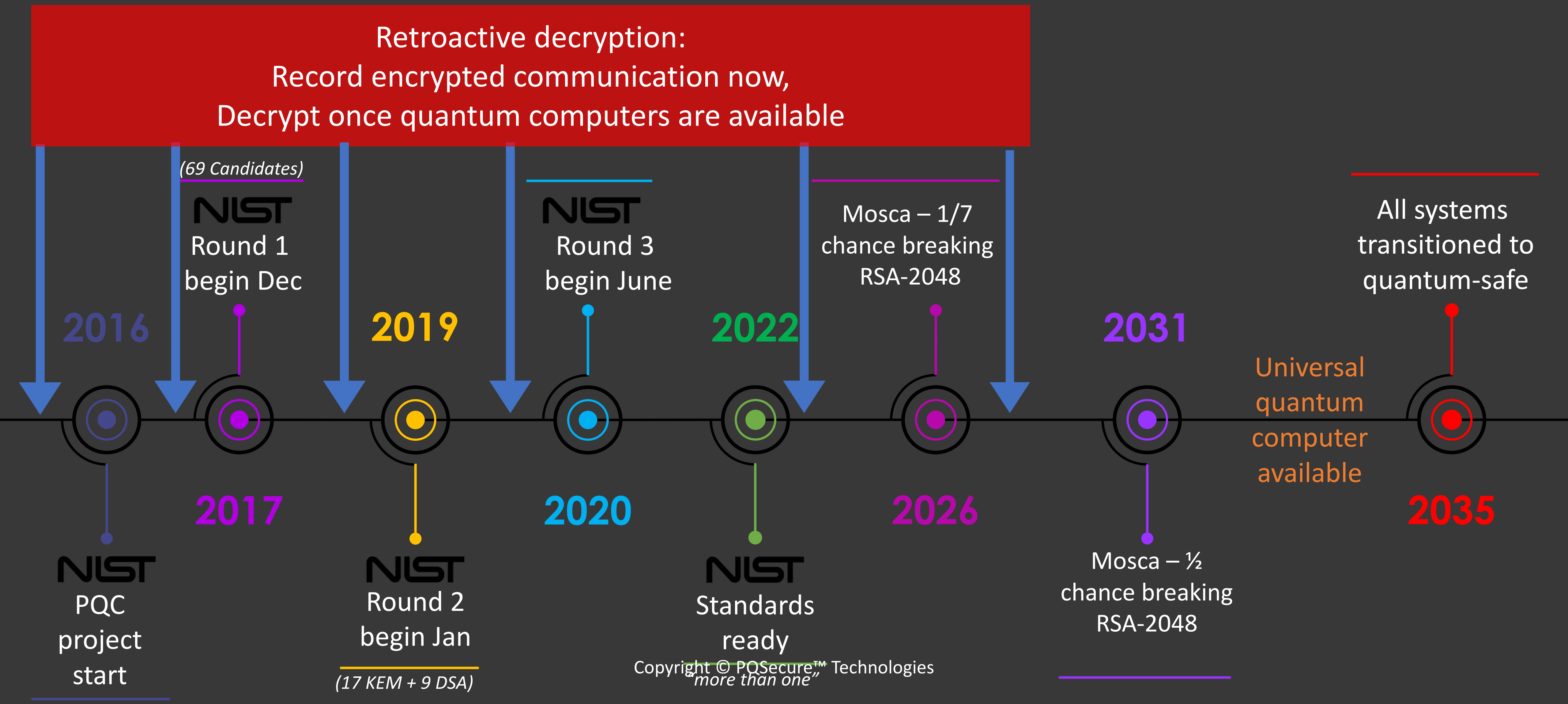
Post-Quantum Cryptography: SIKE

Reza Azarderakhsh
Florida Atlantic University
Founder and CEO
PQSecure Technologies

Timeline of Quantum Computing Threat

PQSecure



Post-Quantum Key-Exchange

- Lattice-based
- Code-based
- Isogeny-based

Post-Quantum Signatures

- Lattice-based
- Hash-based
- Multivariate-based
- Zero-Knowledge based

Open Questions about Post-Quantum Cryptography

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

SIKE Team

UNIVERSITY OF WATERLOO

FAU FLORIDA ATLANTIC UNIVERSITY

UNIVERSITY OF TORONTO

Microsoft Research

NXP

TEXAS INSTRUMENTS

amazon

LinkedIn

IBM Research | Zurich

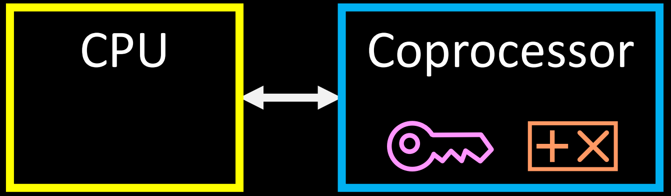
INFOSEC GLOBAL

PQSecure

evolutionQ

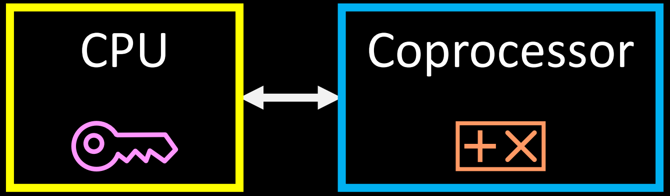
Architecture Selection for Cryptographic Design

HW only



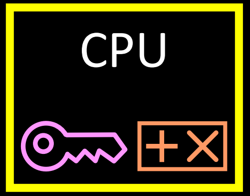
+ Highly optimized for dedicated purpose (power consumption, execution time, security)
- Extra HW costs
- limited flexibility
- HW design effort/complexity

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

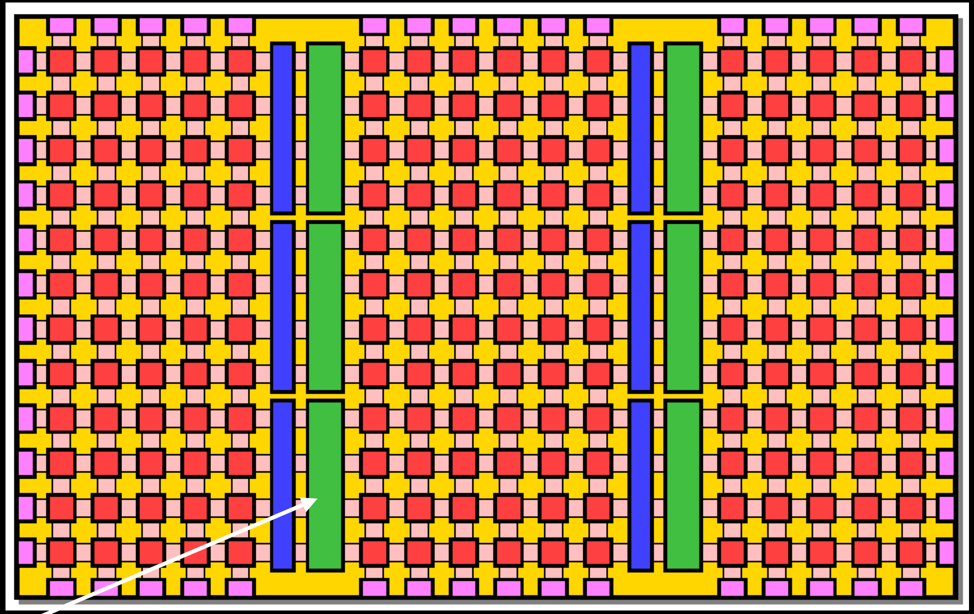
SW only



+ 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

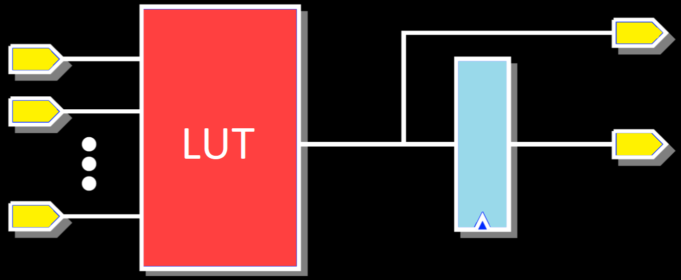
- FPGAs are composed of:
- Programmable logic cells
- A configurable routing matrix
- configurable Input/output cells
- Embedded memory blocks
- Small embedded multipliers
- etc.



18-bit×18-bit multiplier blocks

Inside a logic cell:

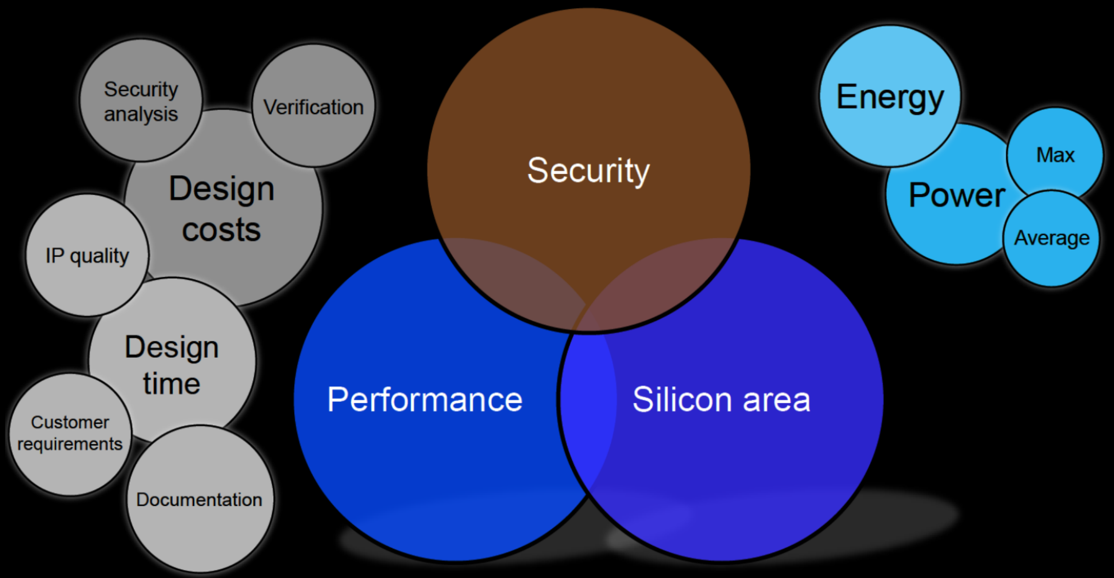
- Connections to the routing matrix
- Programmable lookup-tables
  - 4 inputs, 1 output
  - 6 inputs, 1 output
  - 6 inputs, 2 outputs
- optional registers
  - free pipelining
- more logic for fast carry-propagation



FPGAs vs. ASIC

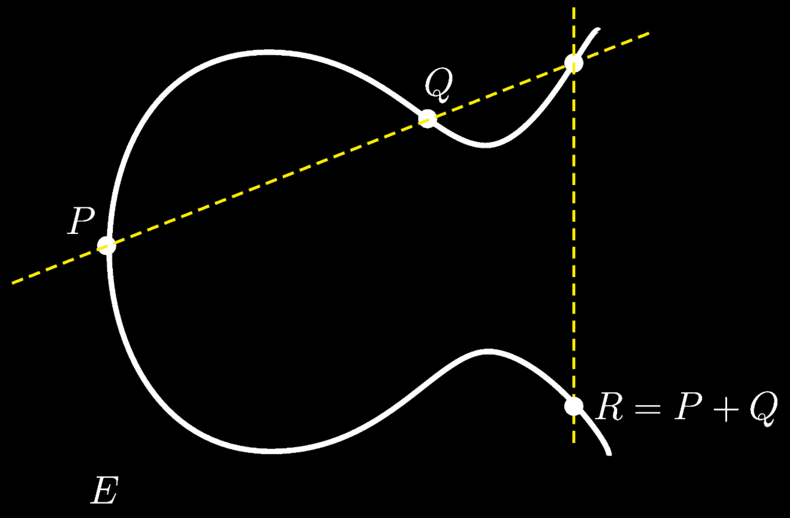
+ prototyping
+ re-usability
+ short time to market
+ simpler design cycle
+ Programmable in the field
+ hardware/software co-design

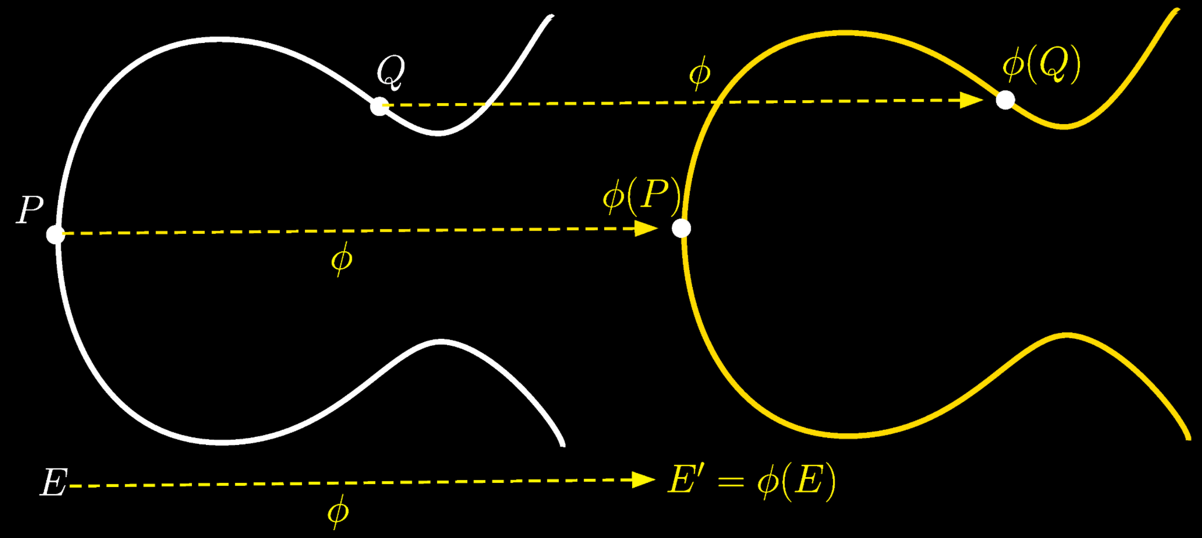
- speed
- silicon footprint
- power and energy consumption
- low cost for high volumes
- better performance
- reconfigurability and redundancy



Isogeny-Based Cryptography

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





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

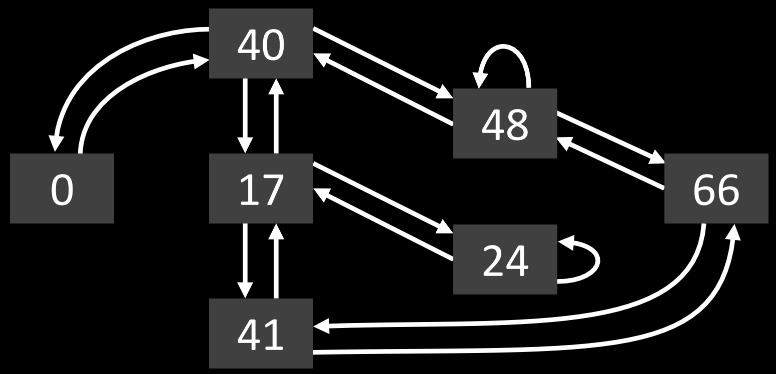
Isogeny Graphs

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

Edges: Isogenies of degree ℓ

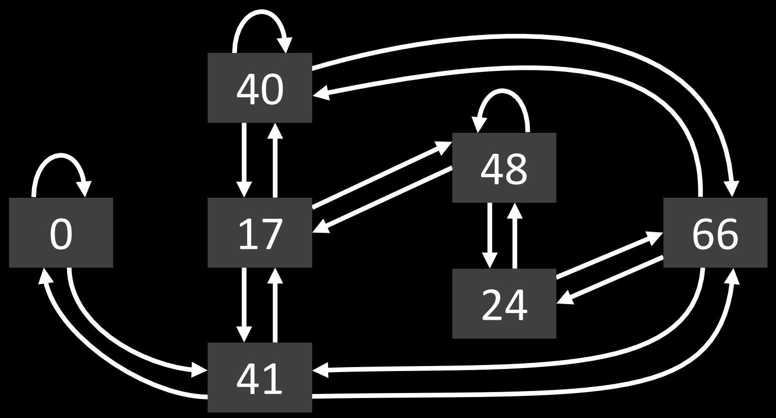
With isogeny of degree ℓ, we get a connected $(\ell + 1)$-regular graph.

Alice



2-isogeny graph

Bob



3-isogeny graph

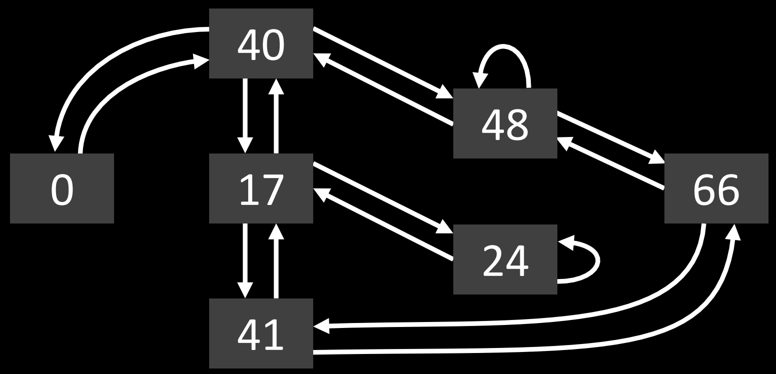
Public Parameters

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

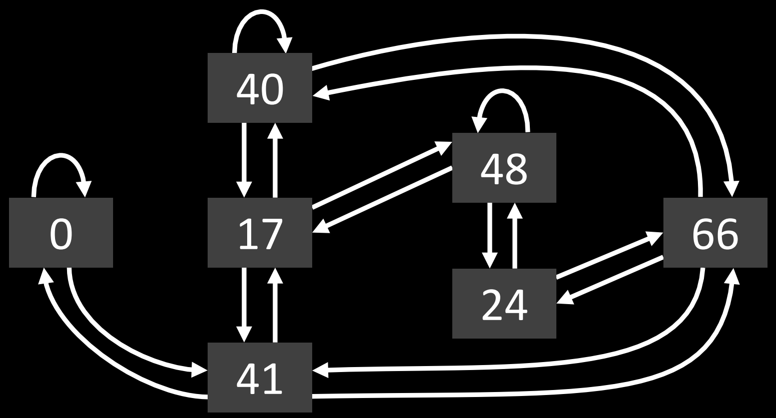
$$\{P_A, Q_A\} \in E_0[2^{e_A}]$$

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

Alice



Bob



$P_A = (53, 55)$

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

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

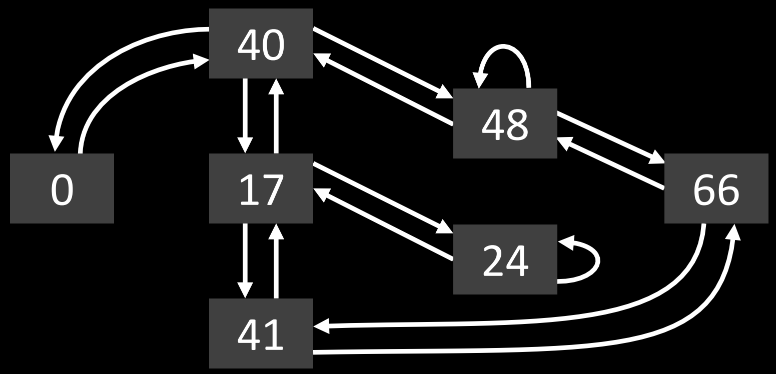
$P_B = (7w + 20, 31w + 50)$

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

Secret Key

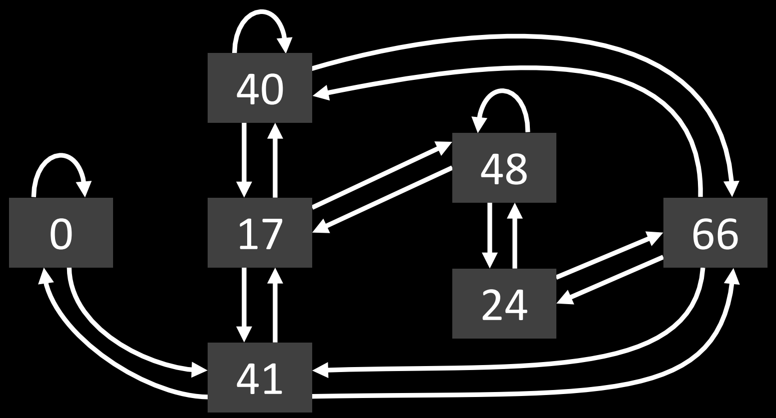
$s_A \in [0,2^{e_A})$
$s_B \in [0,3^{e_B})$

Alice



$s_A = 6$

Bob

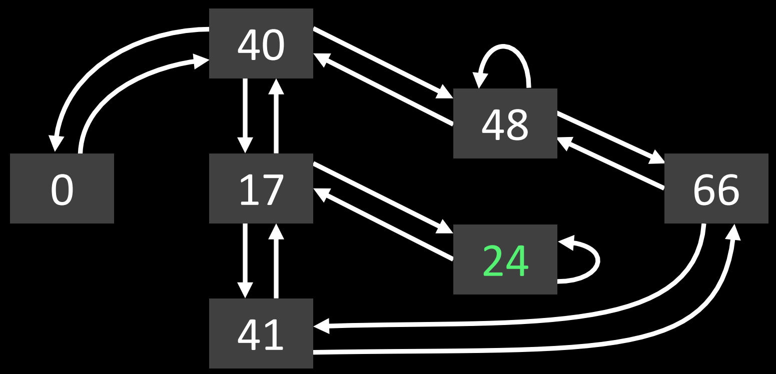


$s_B = 3$

Public Key Generation

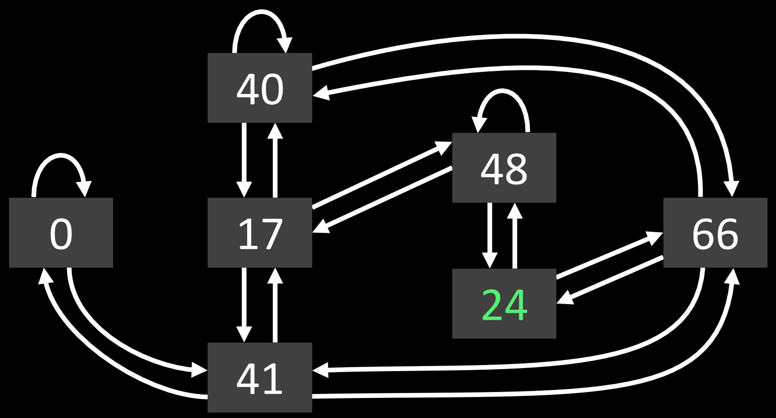
E_0

Alice



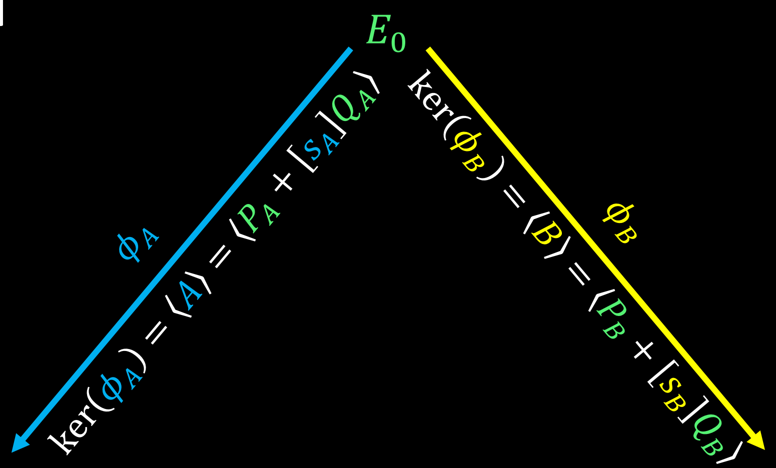
$E_0\colon y^2 = x^3 + x$

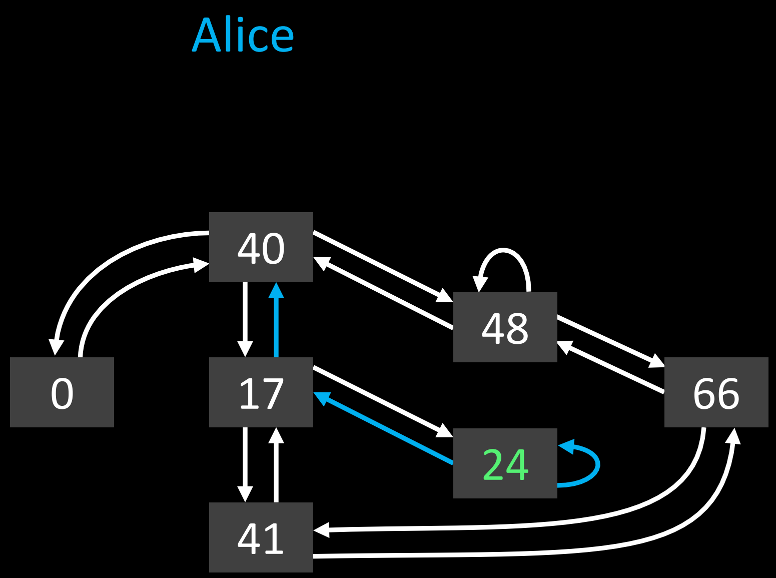
Bob



$E_0\colon y^2 = x^3 + x$

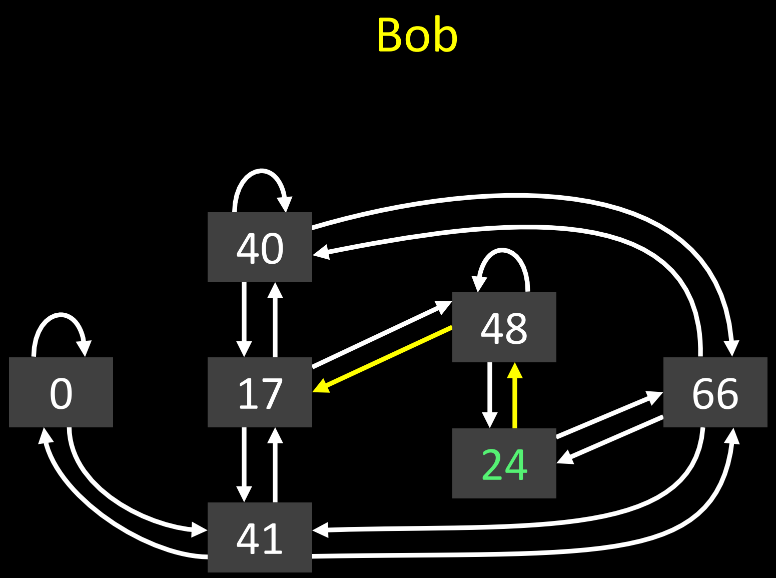
Public Key Generation





$E_0 \colon y^2 = x^3 + x$

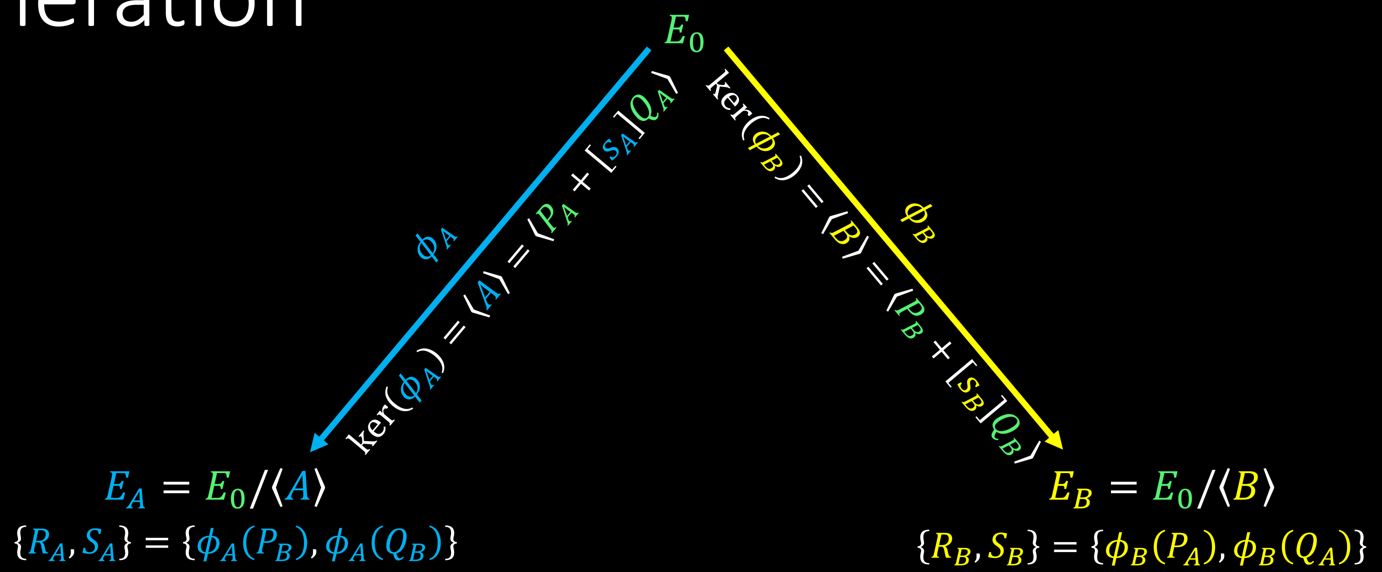
$\phi_A \colon E_0 \to E_A$

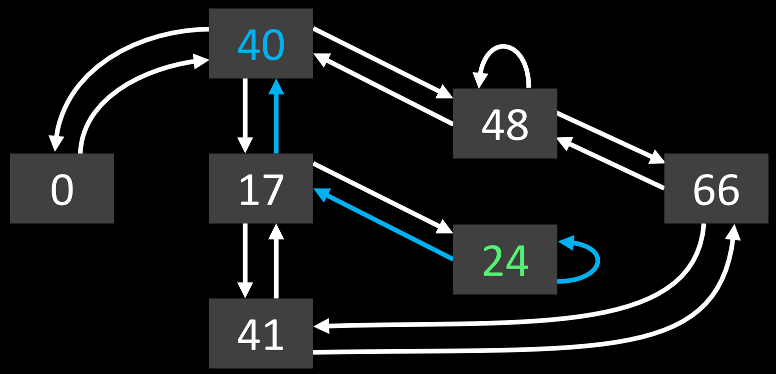


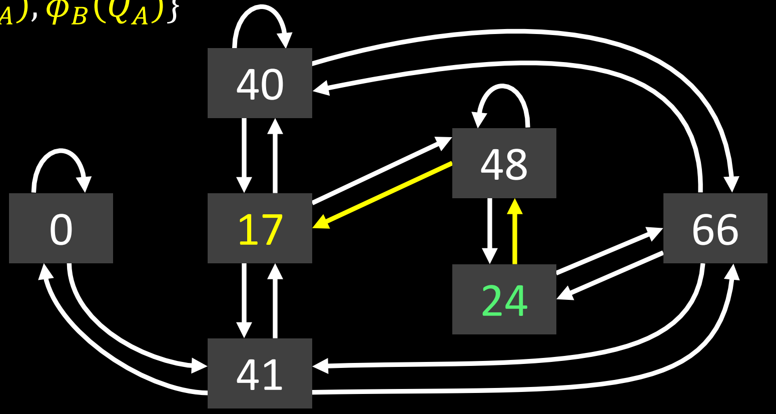
$E_0 \colon y^2 = x^3 + x$

$\phi_B \colon E_0 \to E_B$

Public Key Generation







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

$\phi_A: E_0 \to E_A$

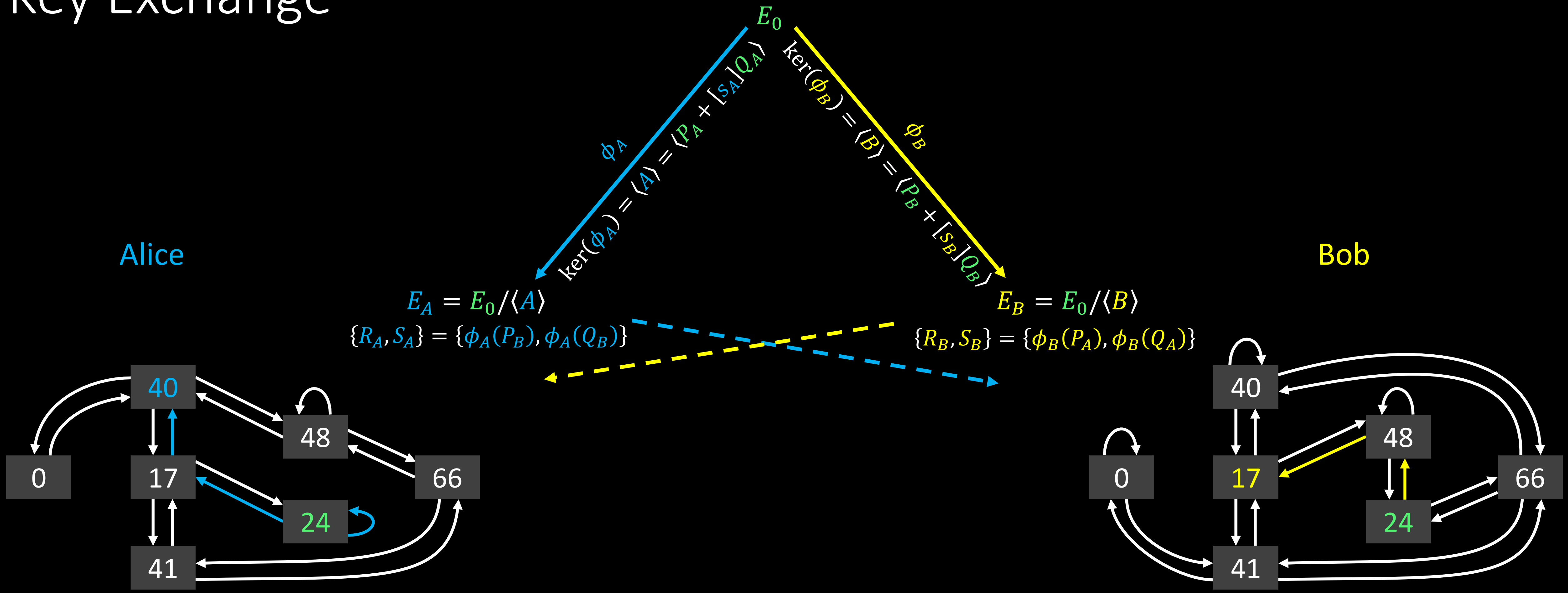
$E_A: y^2 = x^3 + 22x + 35$

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

$\phi_B: E_0 \to E_B$

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

Key Exchange



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

$\phi_A: E_0 \to E_A$

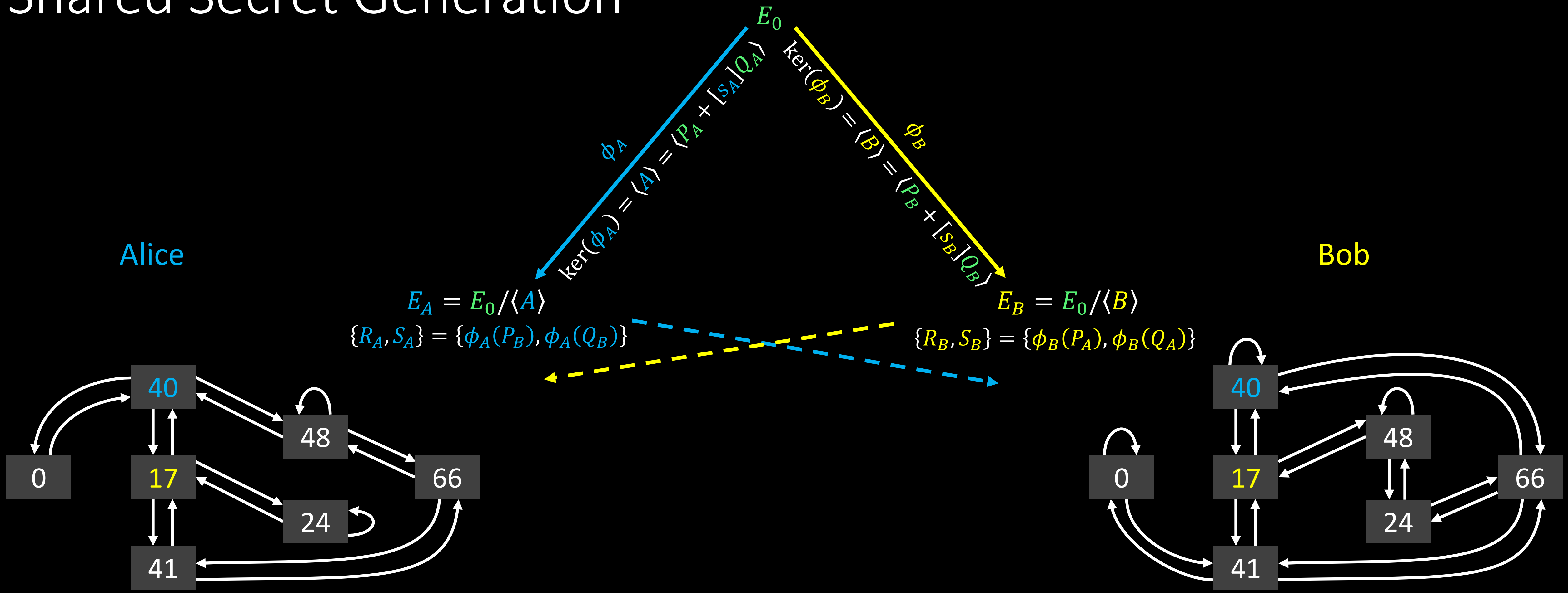
$E_A: y^2 = x^3 + 22x + 35$

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

$\phi_B: E_0 \to E_B$

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

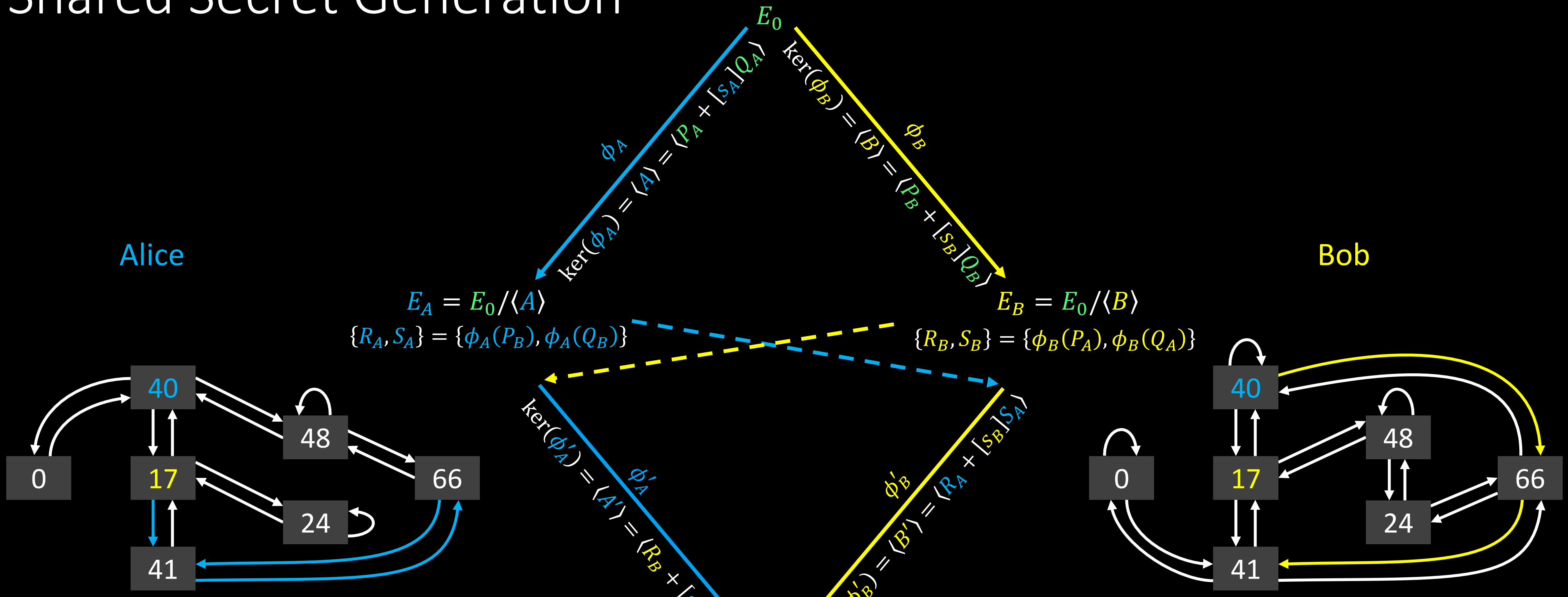
Shared Secret Generation



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

$E_A: y^2 = x^3 + 22x + 35$

Shared Secret Generation



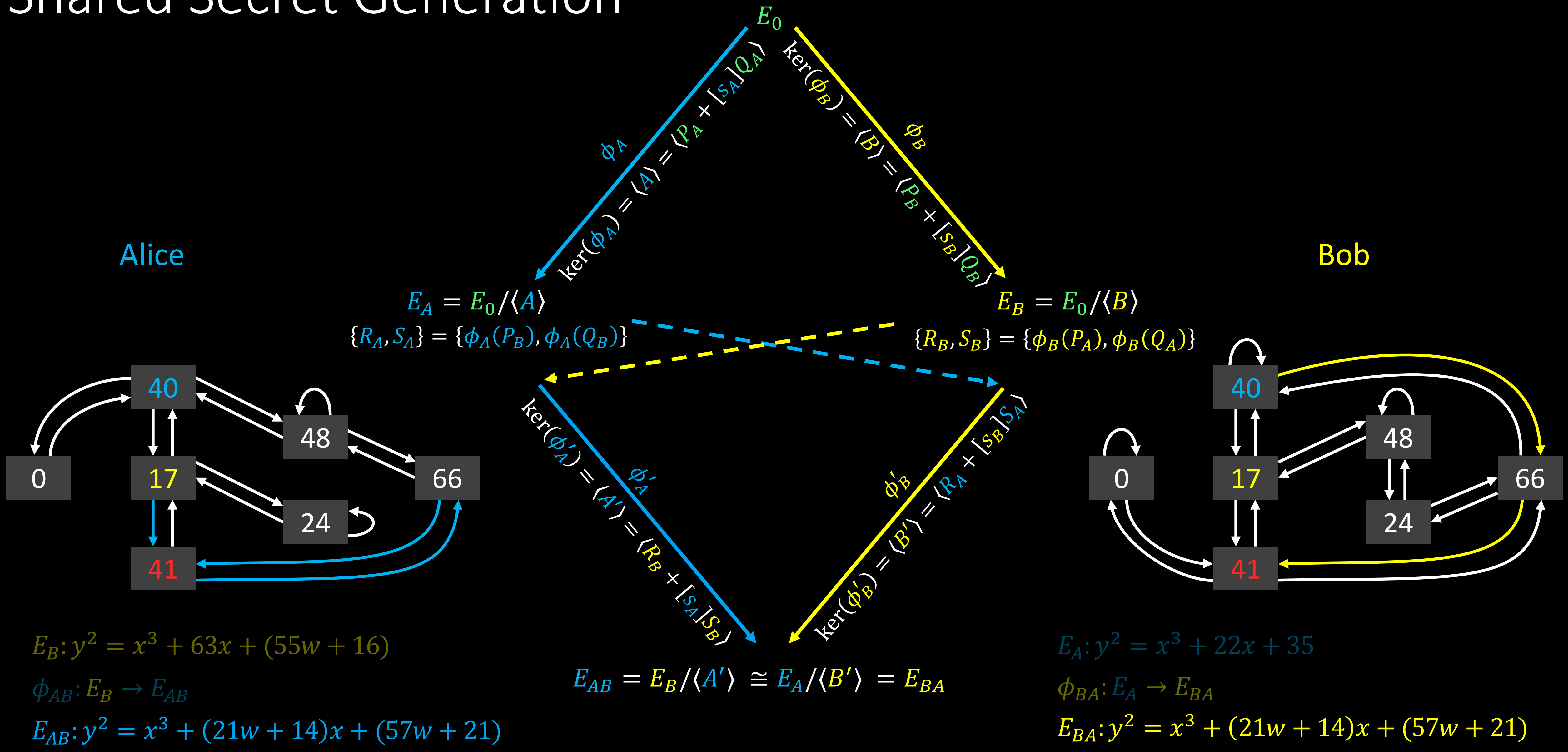
$E_B: y^2 = x^3 + 63x + (55w + 16)$

$\phi_{AB}: E_B \to E_{AB}$

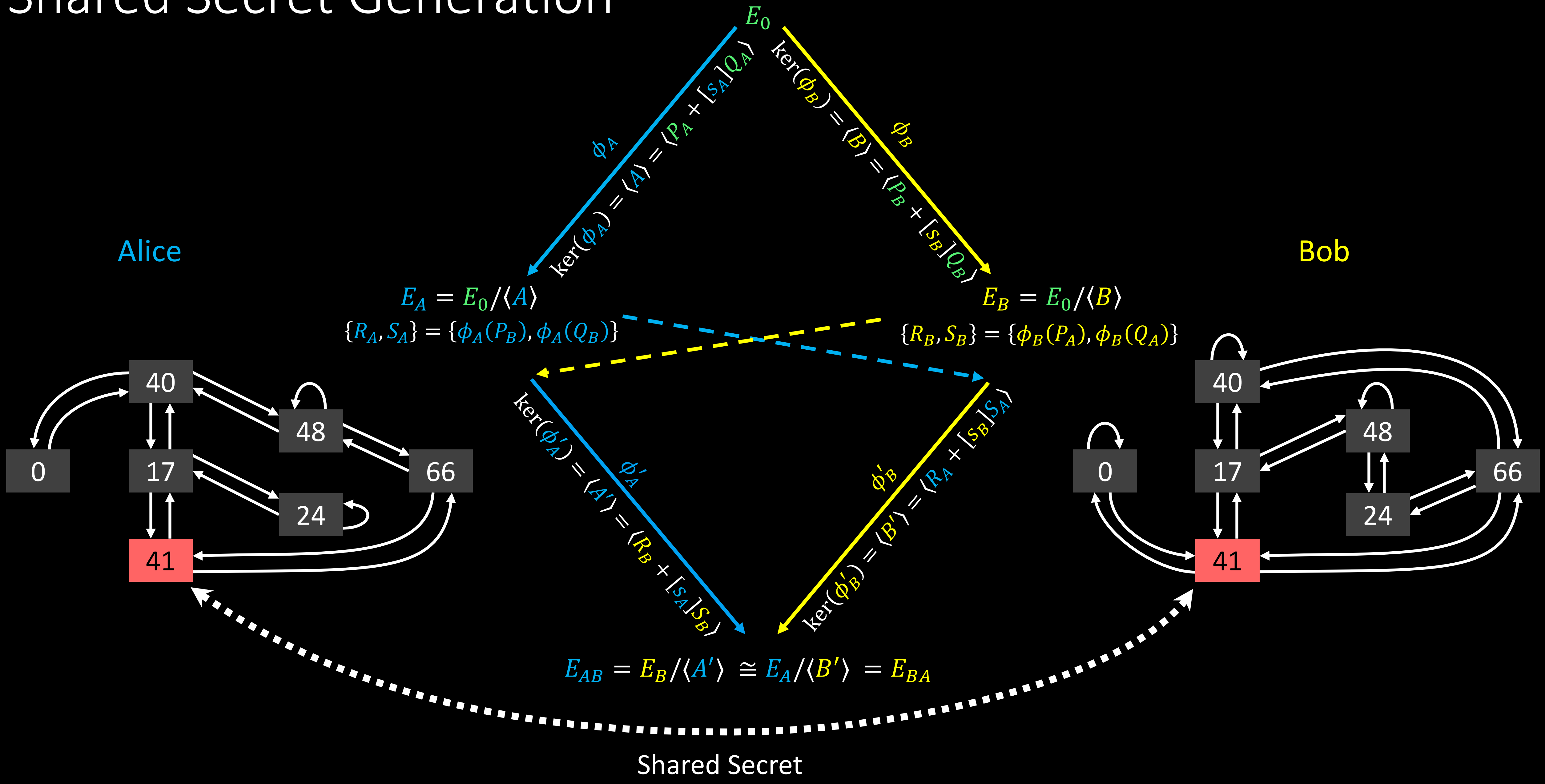
$E_A: y^2 = x^3 + 22x + 35$

$\phi_{BA}: E_A \to E_{BA}$

Shared Secret Generation



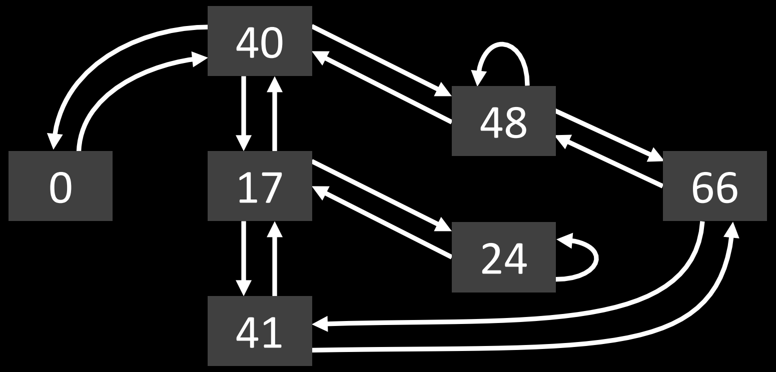
Shared Secret Generation



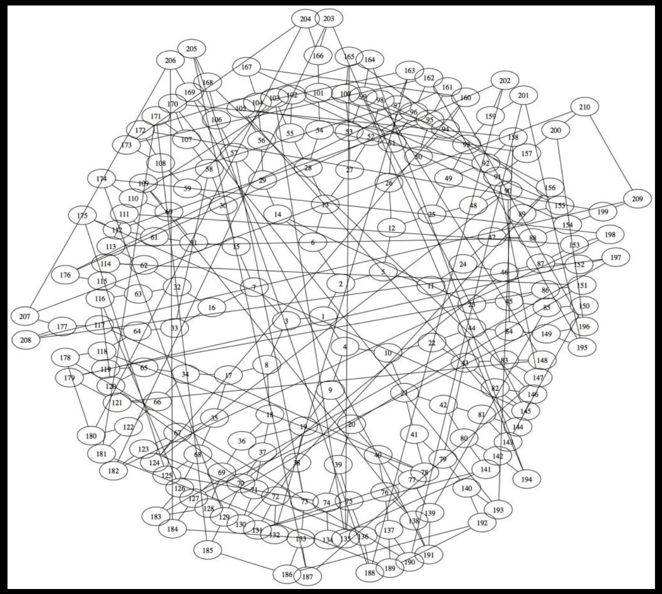
SIKE Round 2 Key sizes

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

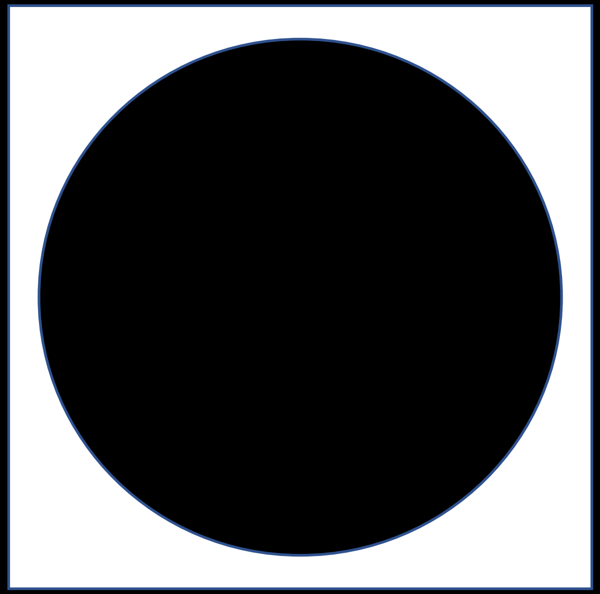
Isogeny Graphs



$p = 71 = 2^3 \cdot 3^2 - 1$
nodes $= 7$

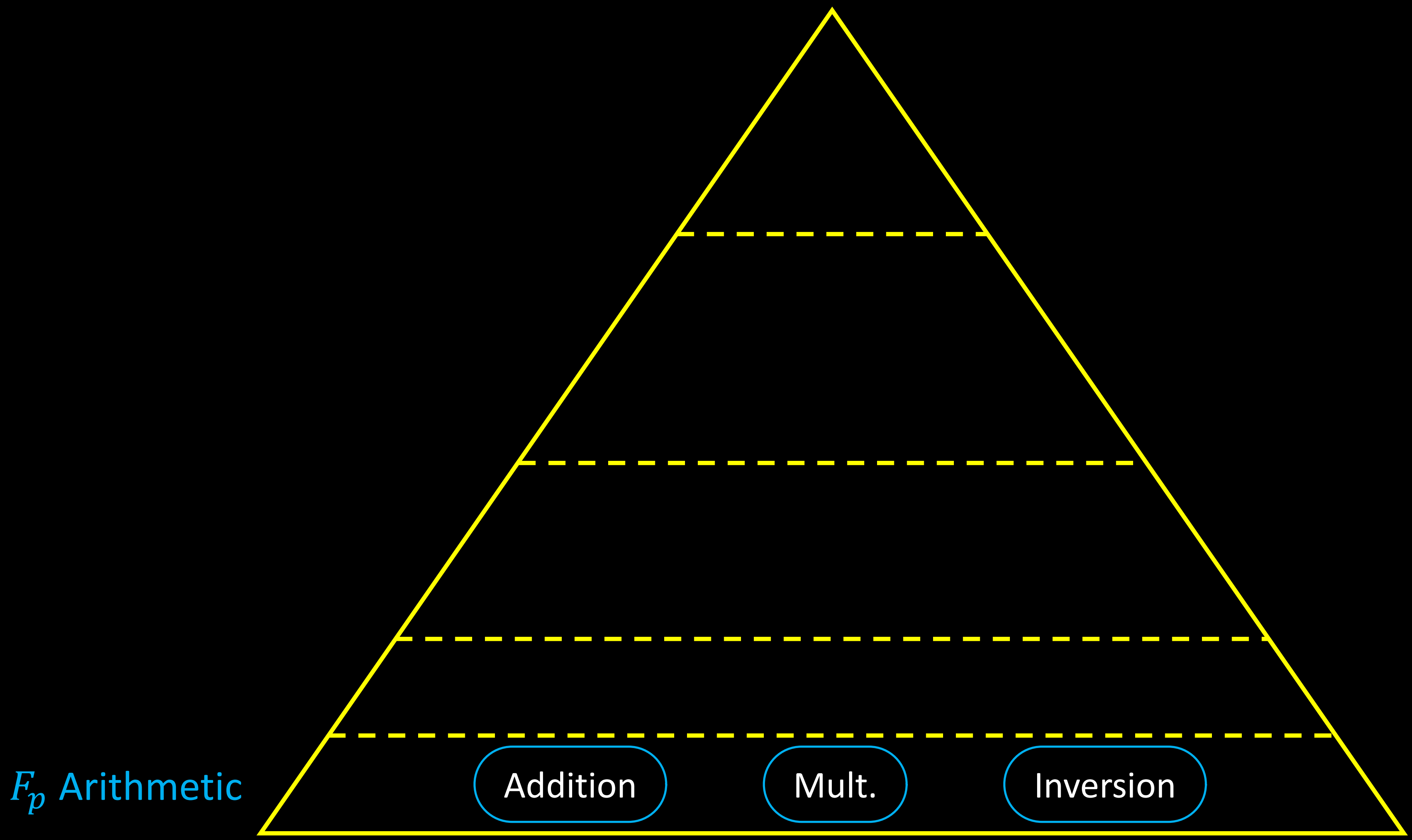


$p = 2521$
nodes $= 210$
[CLG06]

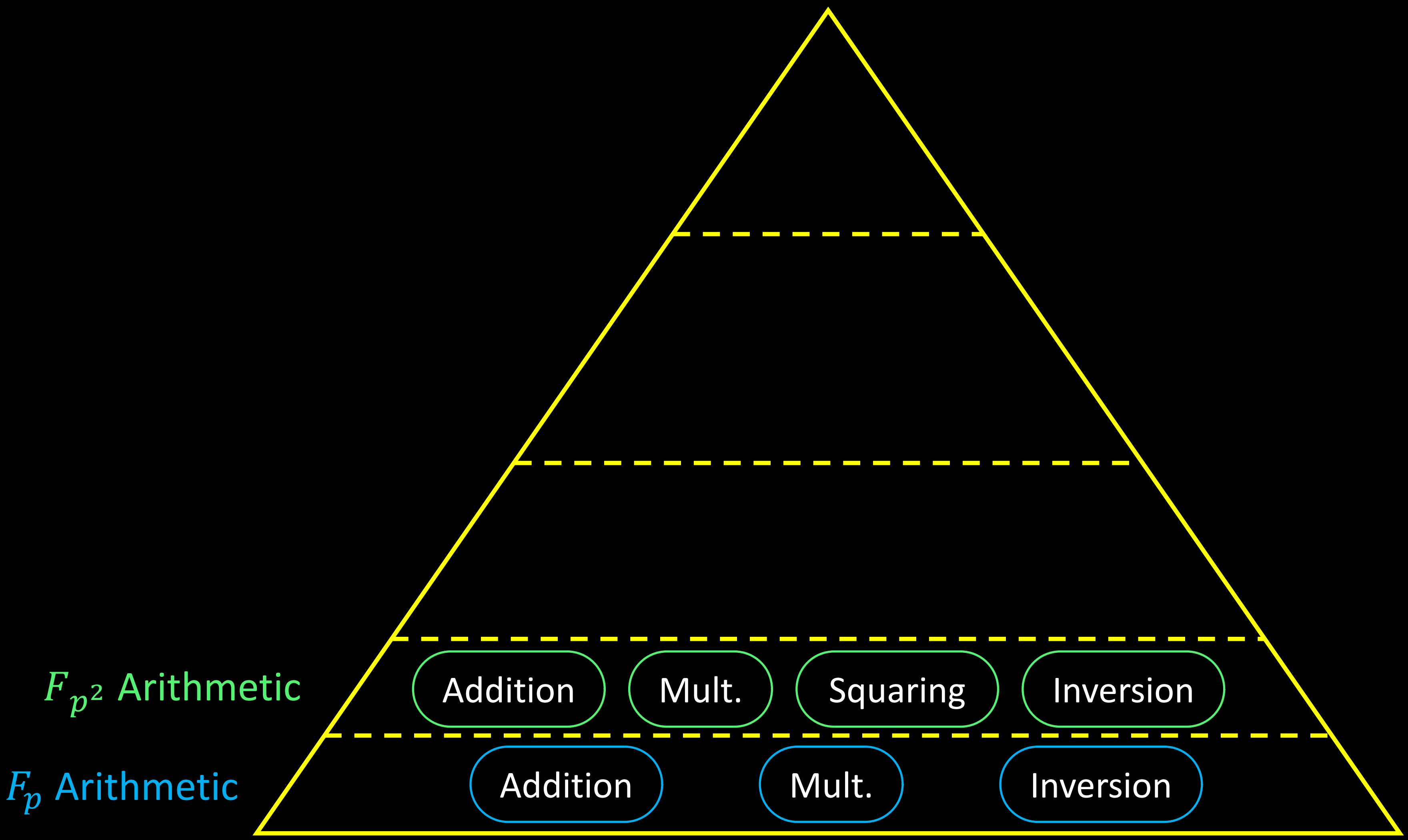


SIKEp434 $\approx 2^{216} \cdot 3^{137} - 1$
nodes $\approx 2^{430}$

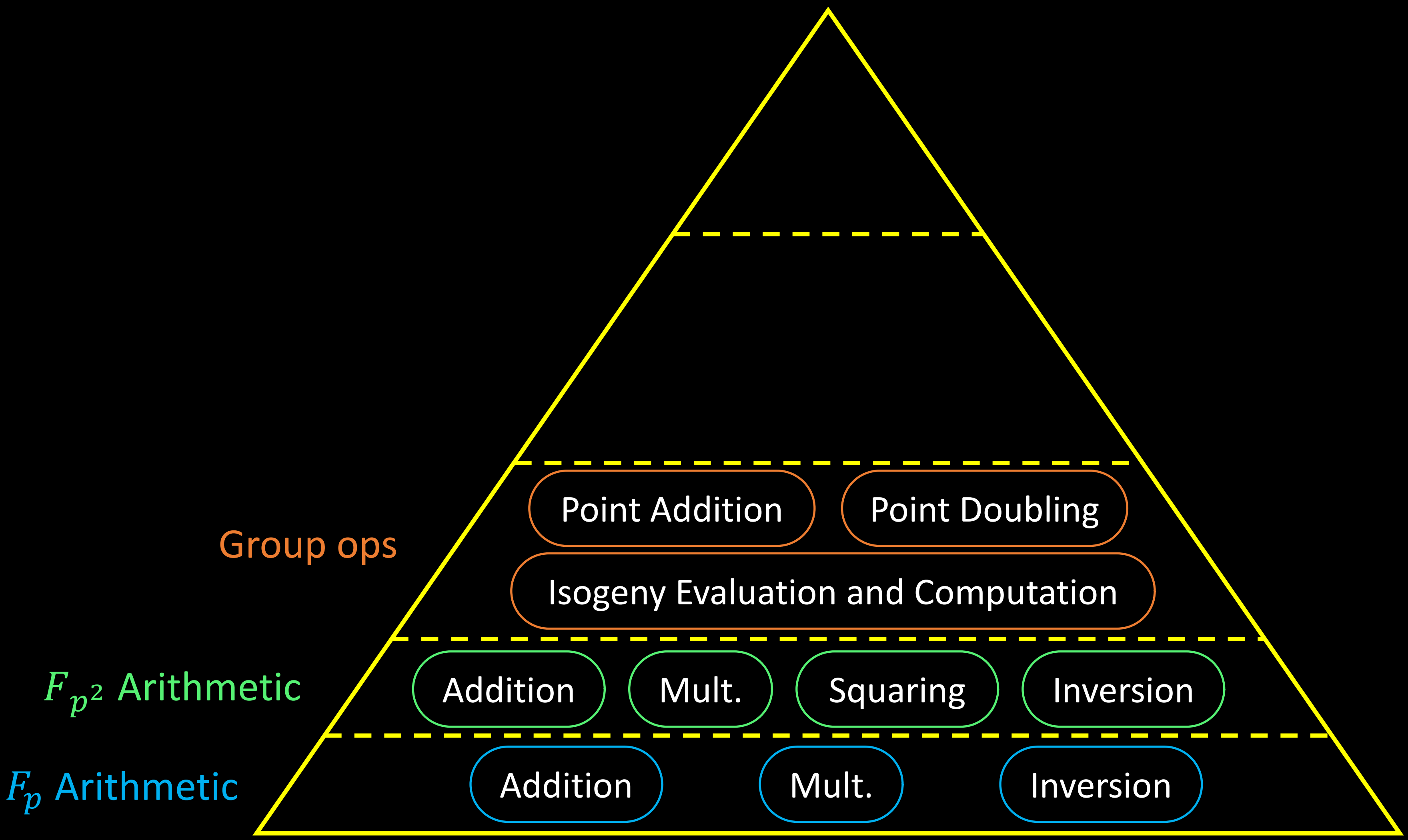
SIDH Computations



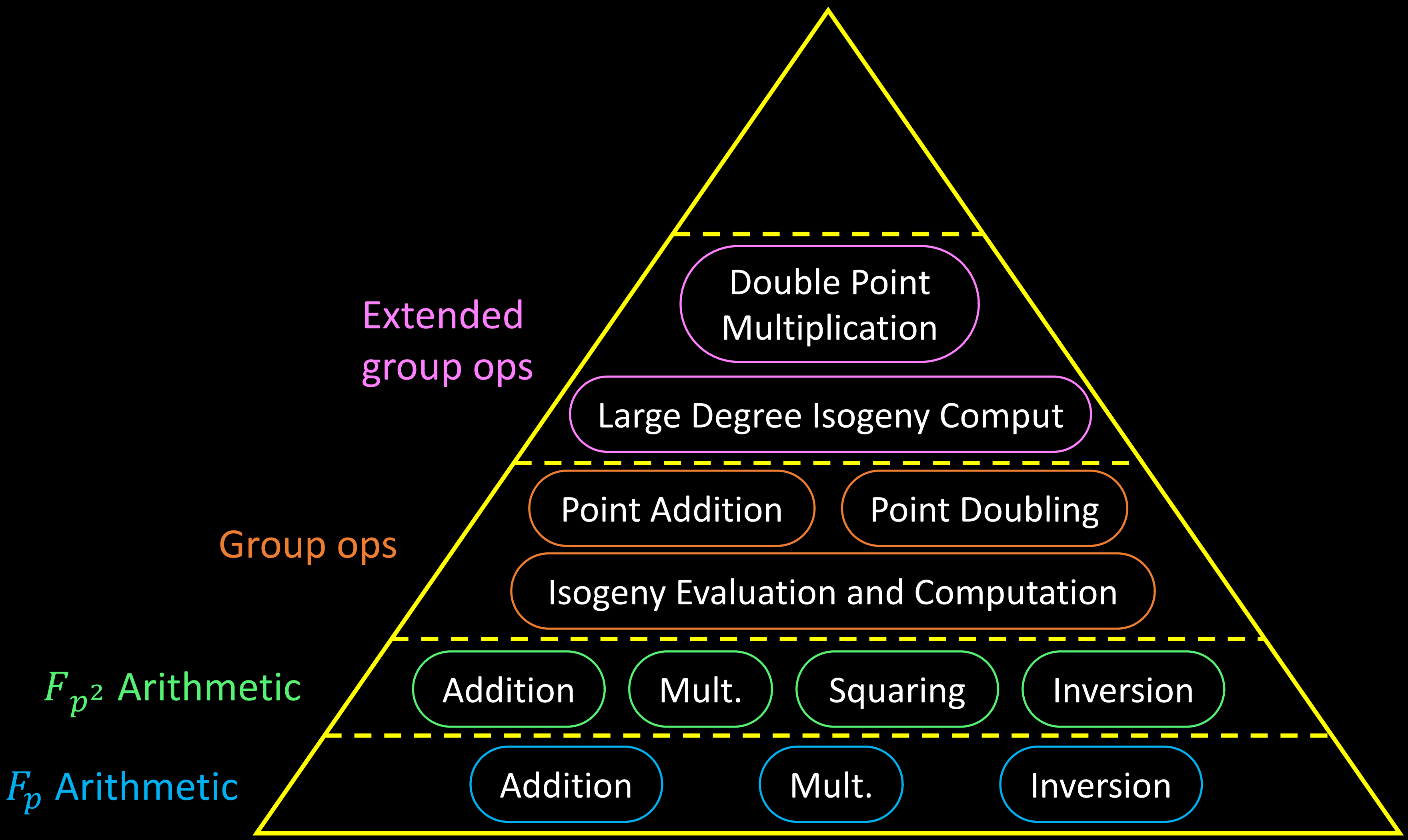
SIDH Computations



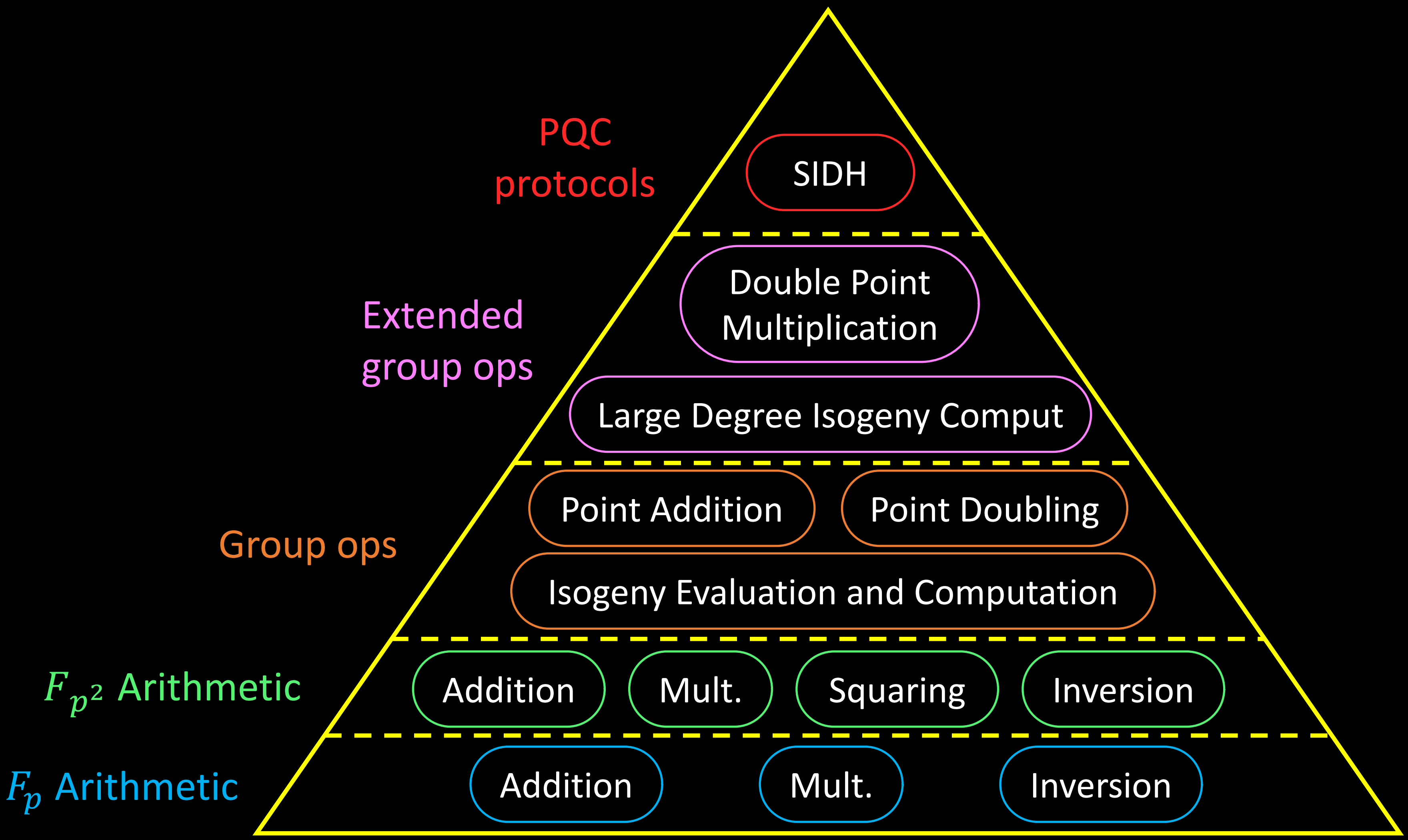
SIDH Computations



SIDH Computations



SIDH Computations

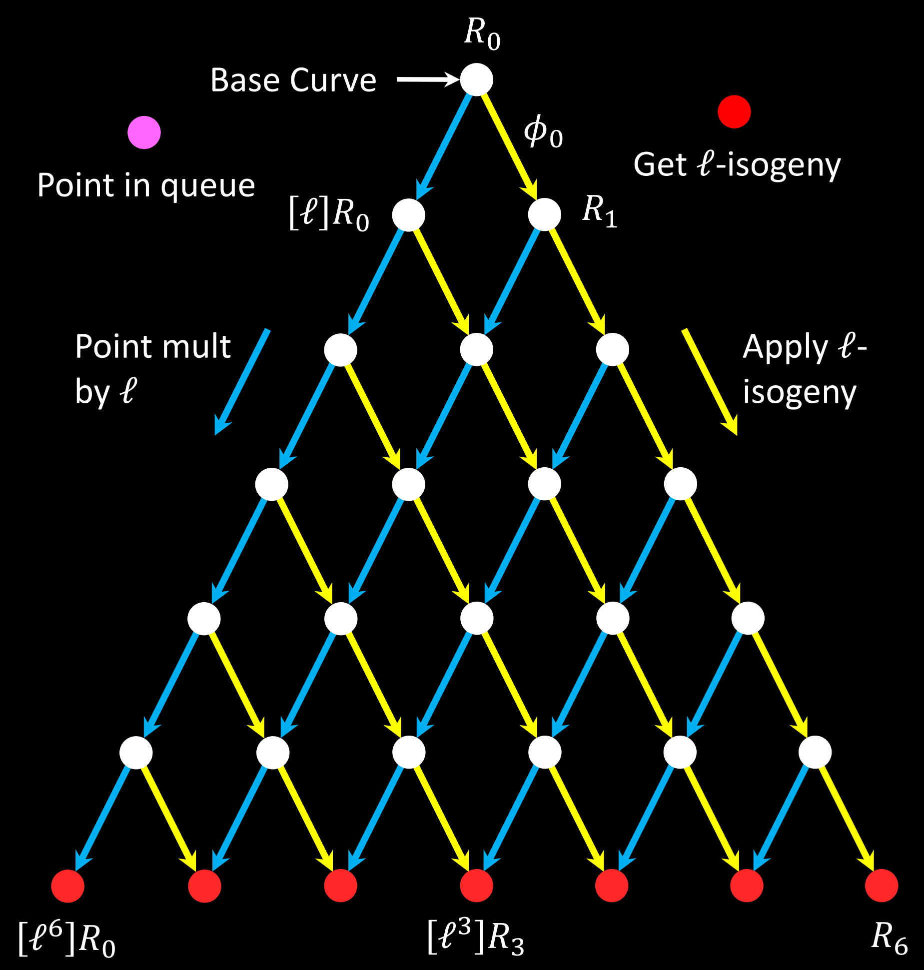


Large degree isogeny computations

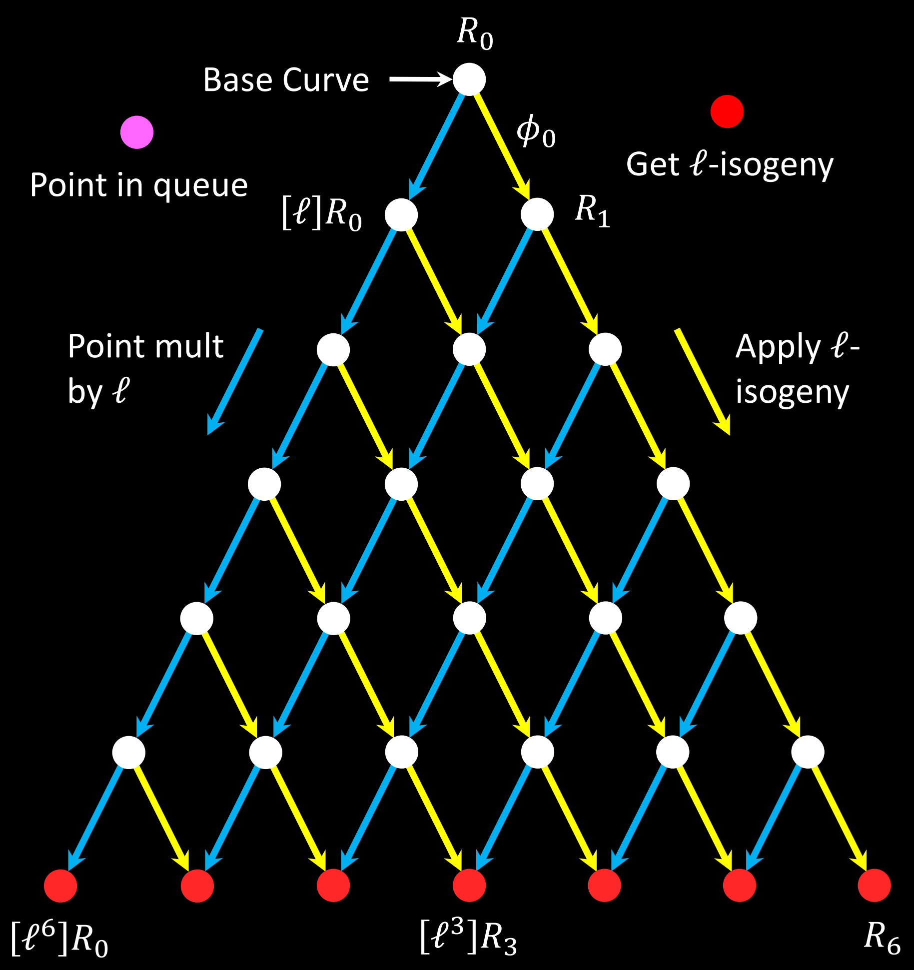
- Get isogeny Kernel $\left[\ell^{e-i-1}\right]R_i$
- Compute Isogenies $\phi_i \coloneqq E_i/\langle\left[\ell^{e-i-1}\right]R_i\rangle$
- Compute $E_{i+1} = \phi_i(E_i)$
- Push points to new curve $R_{i+1} = \phi_i(R_i)$

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

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

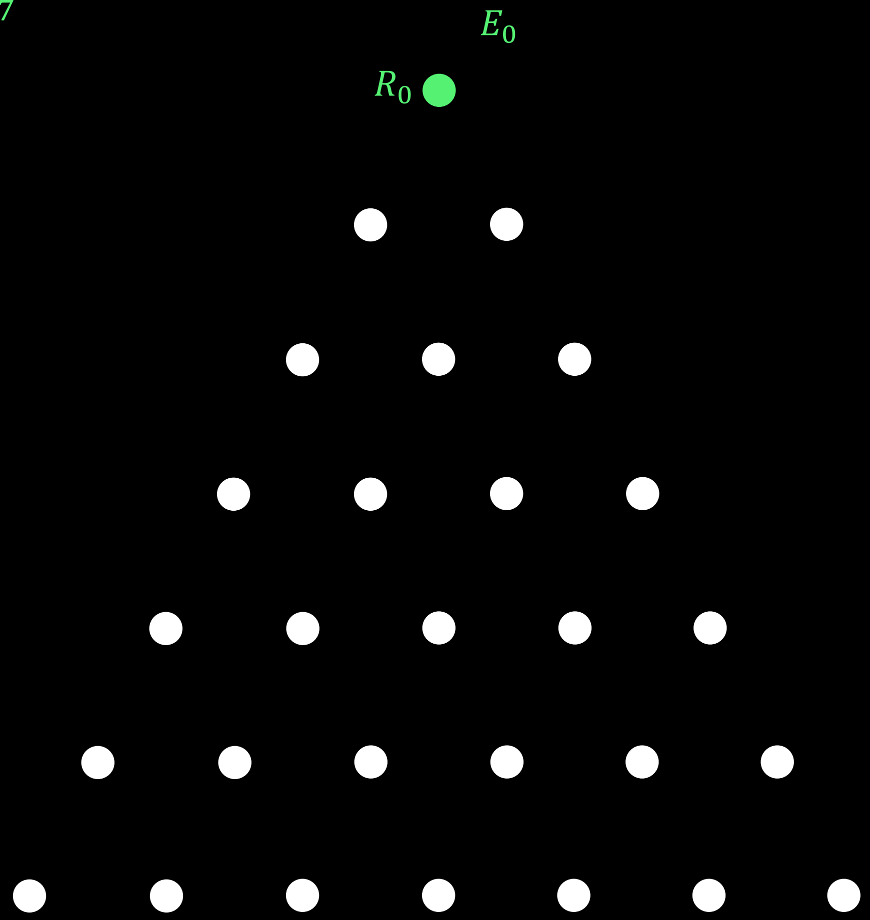


Large degree isogeny computations



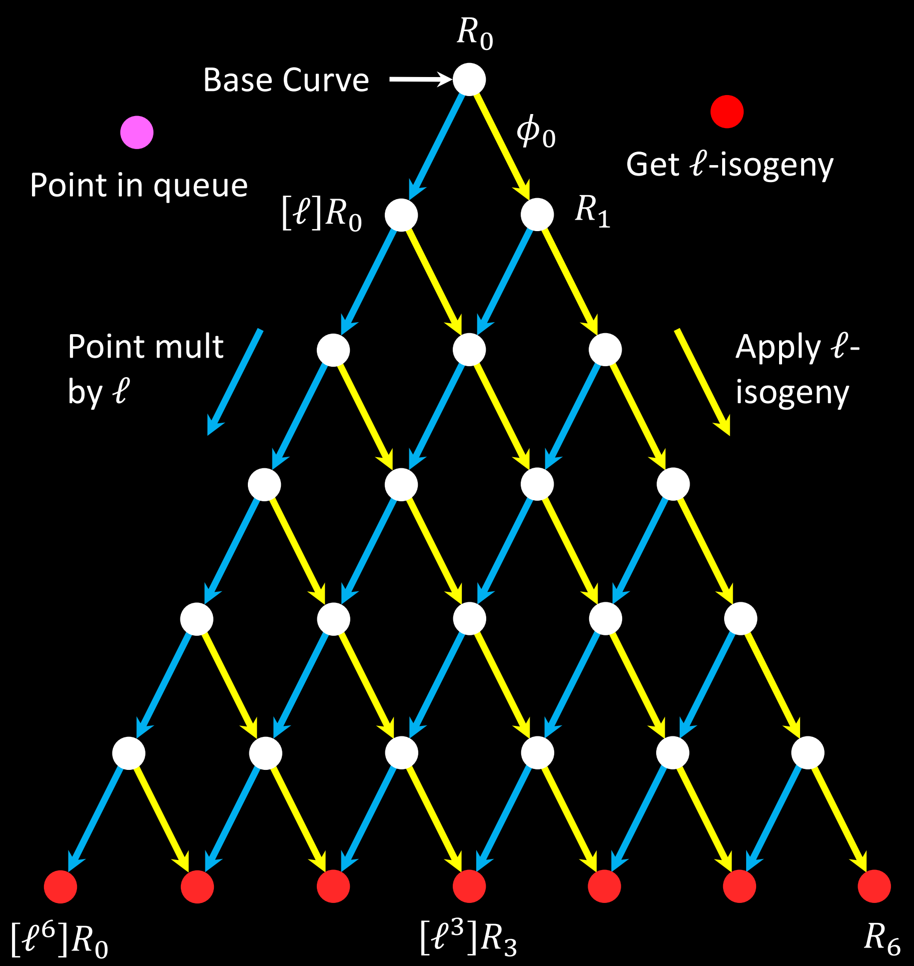
$e = 7$

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

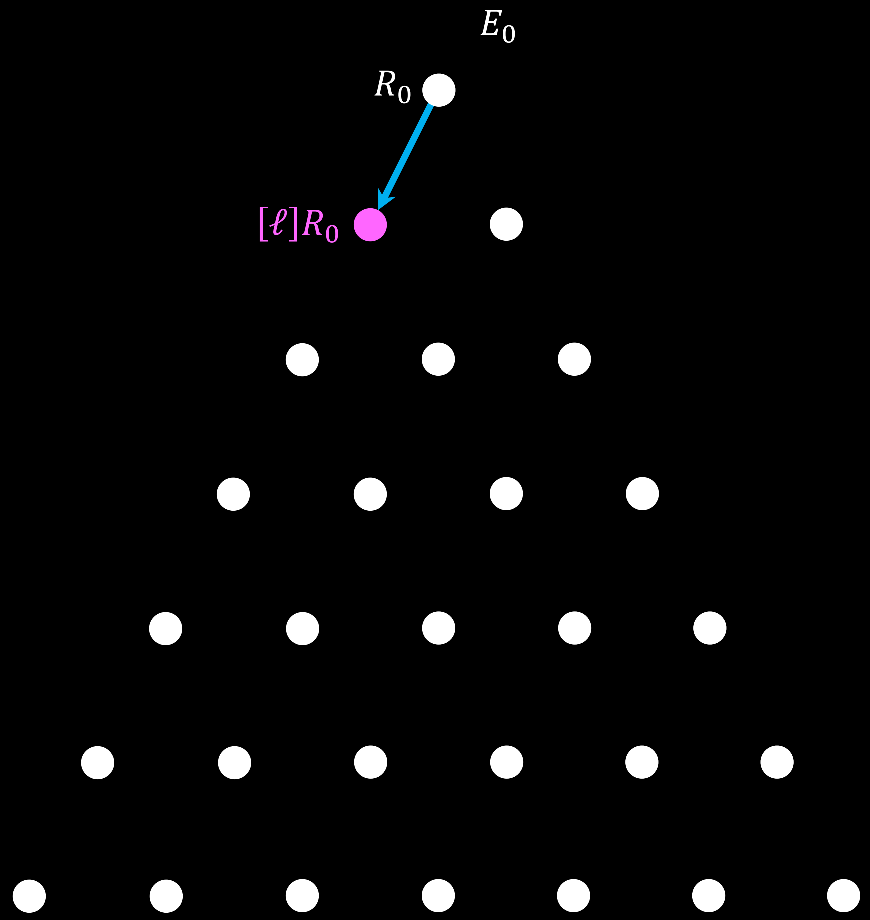


Large degree isogeny computations

$e = 7$

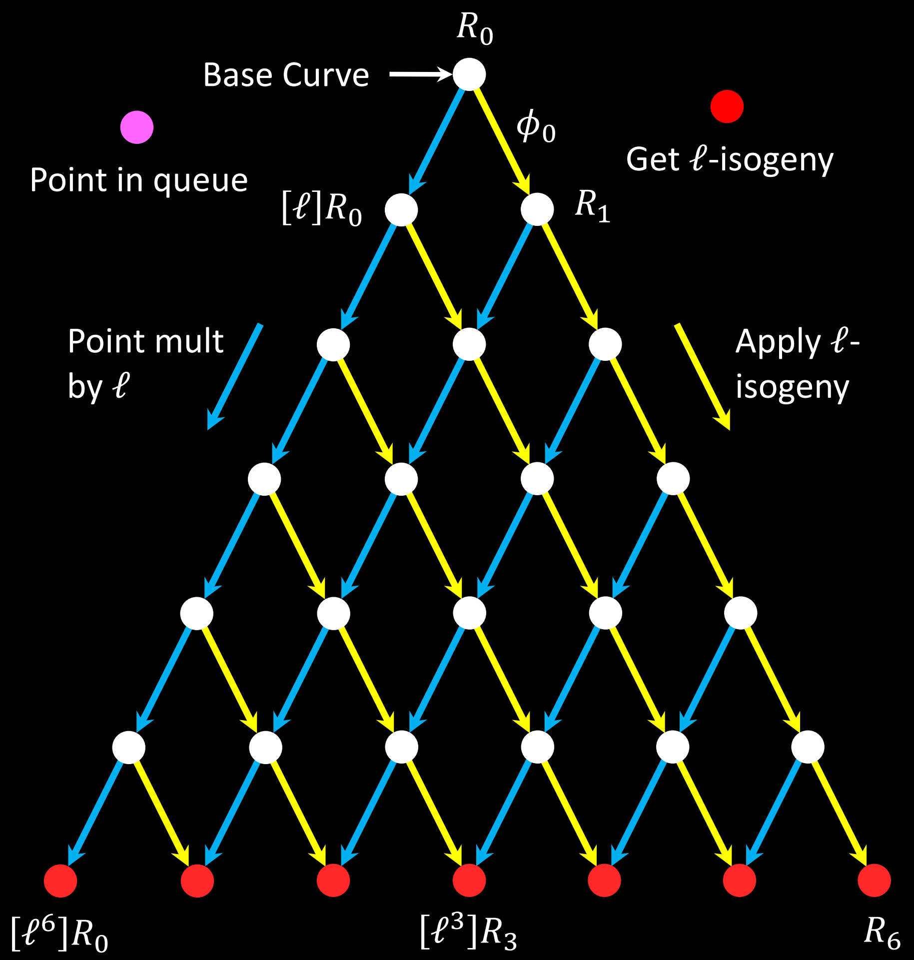


Order of $[\ell]R_0$ is ℓ^6

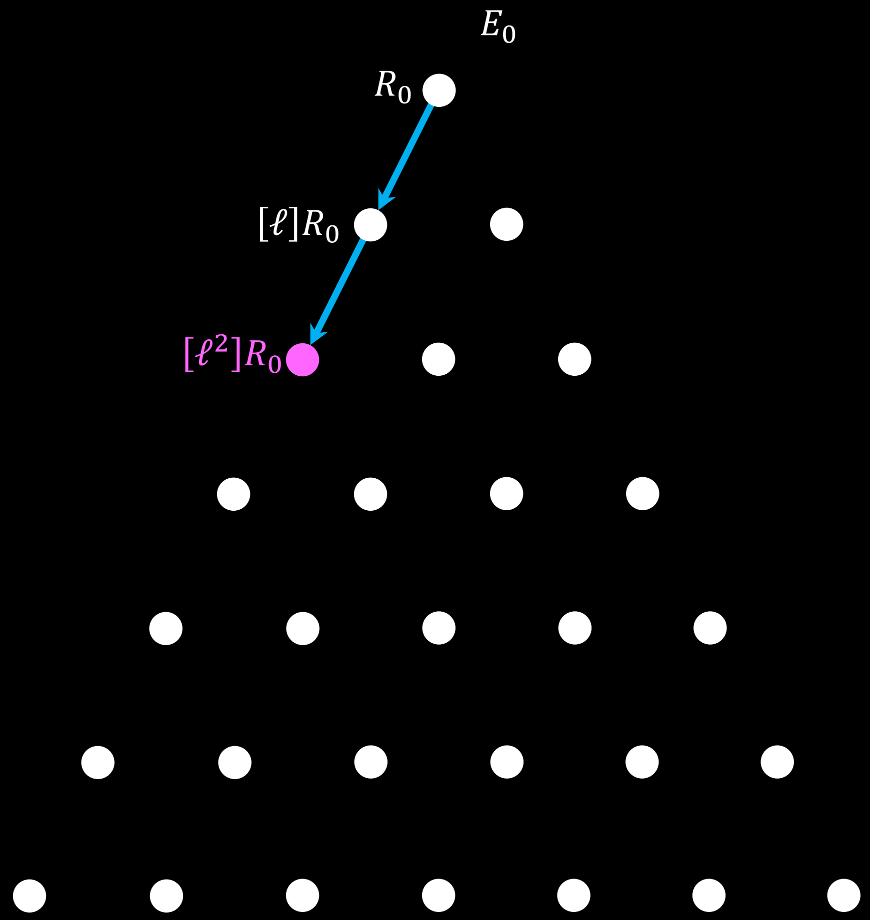


Large degree isogeny computations

$e = 7$

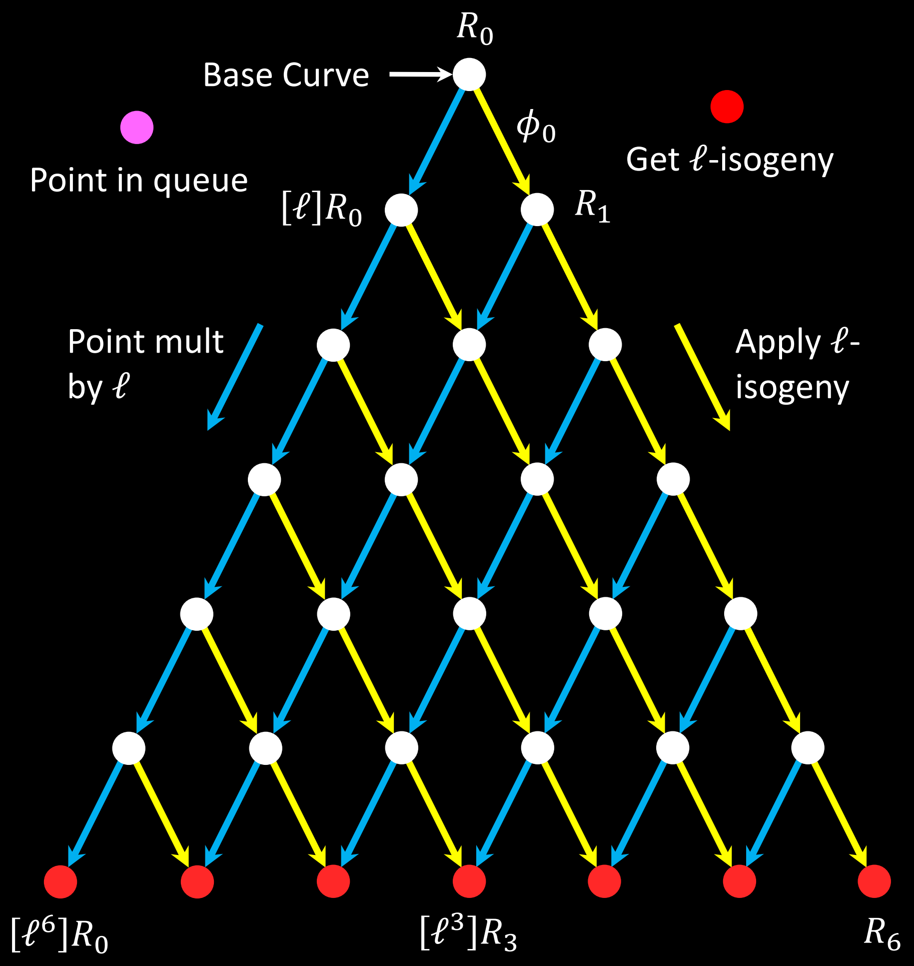


Order of $[\ell^2]R_0$ is ℓ^5

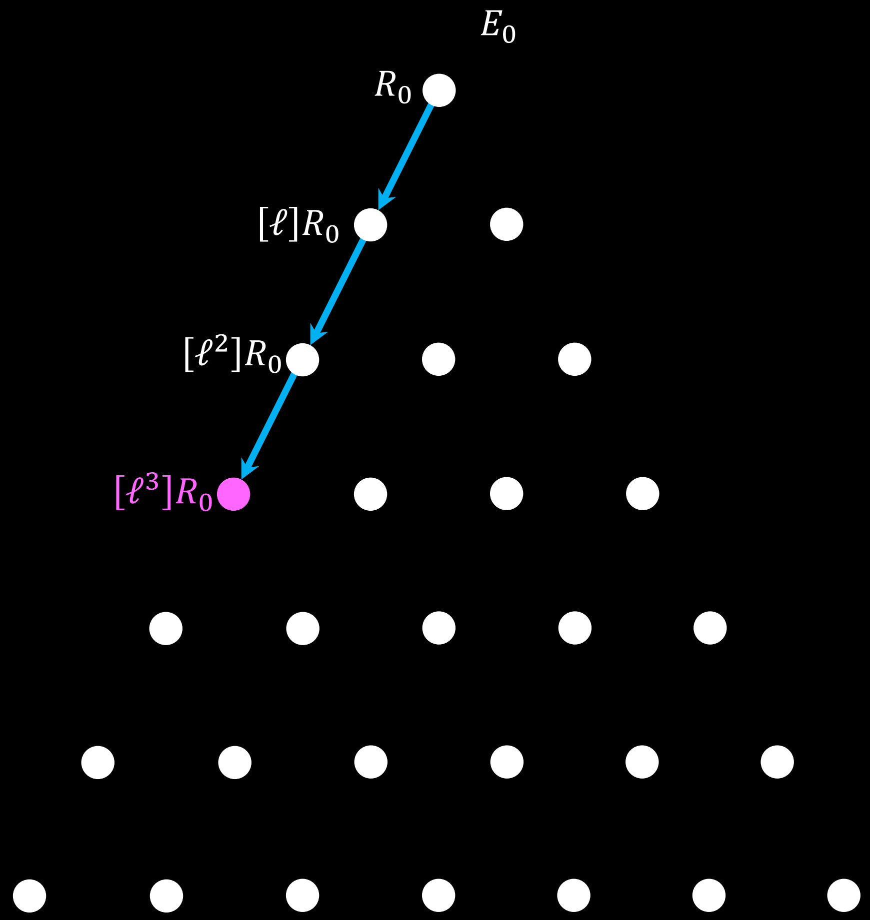


Large degree isogeny computations

$e = 7$

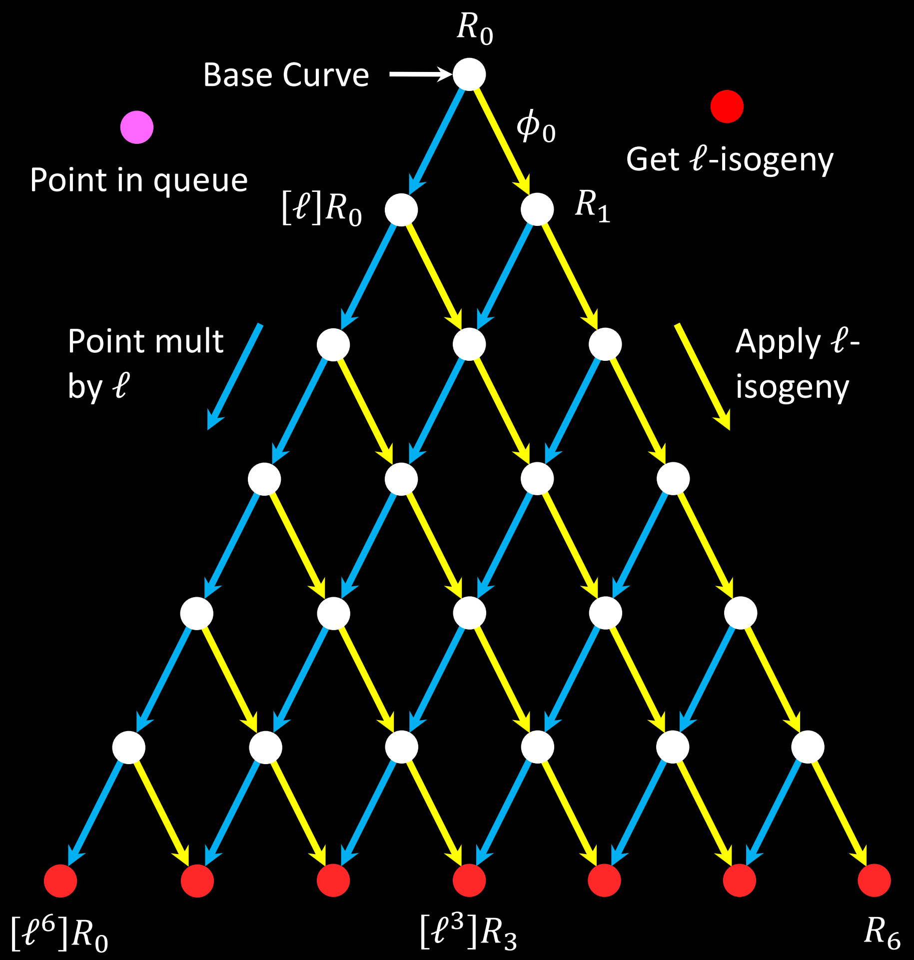


Order of $[\ell^3]R_0$ is ℓ^4

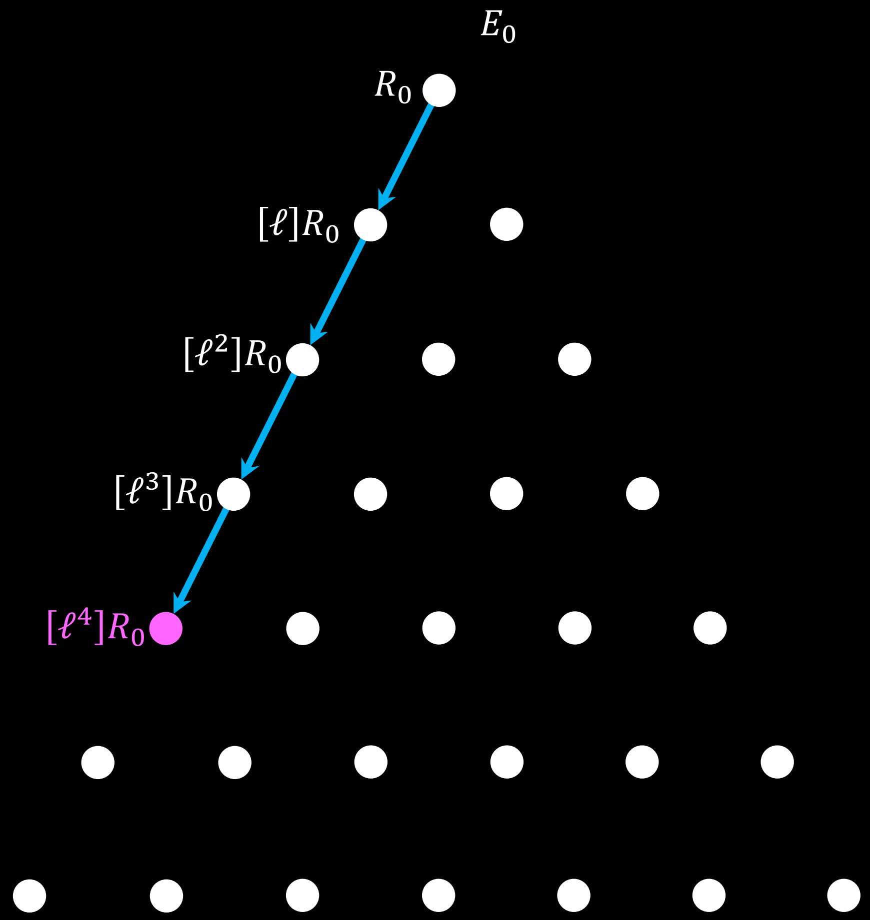


Large degree isogeny computations

$e = 7$

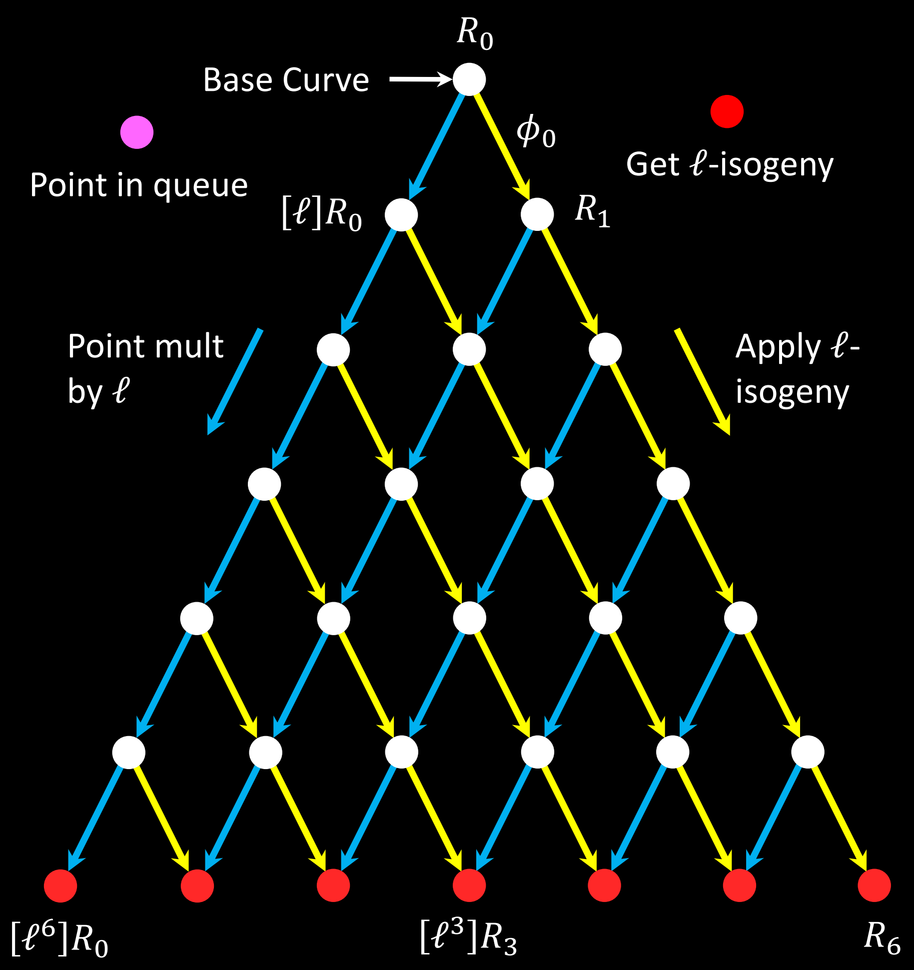


Order of $[\ell^4]R_0$ is ℓ^3

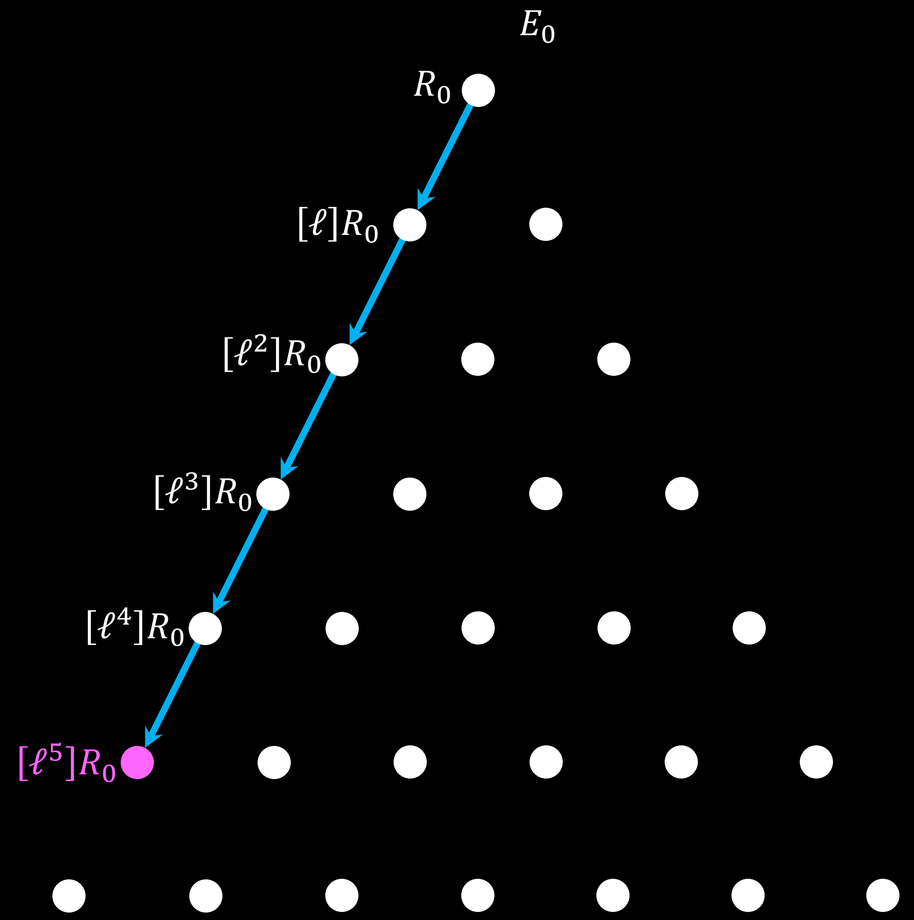


Large degree isogeny computations

$e = 7$

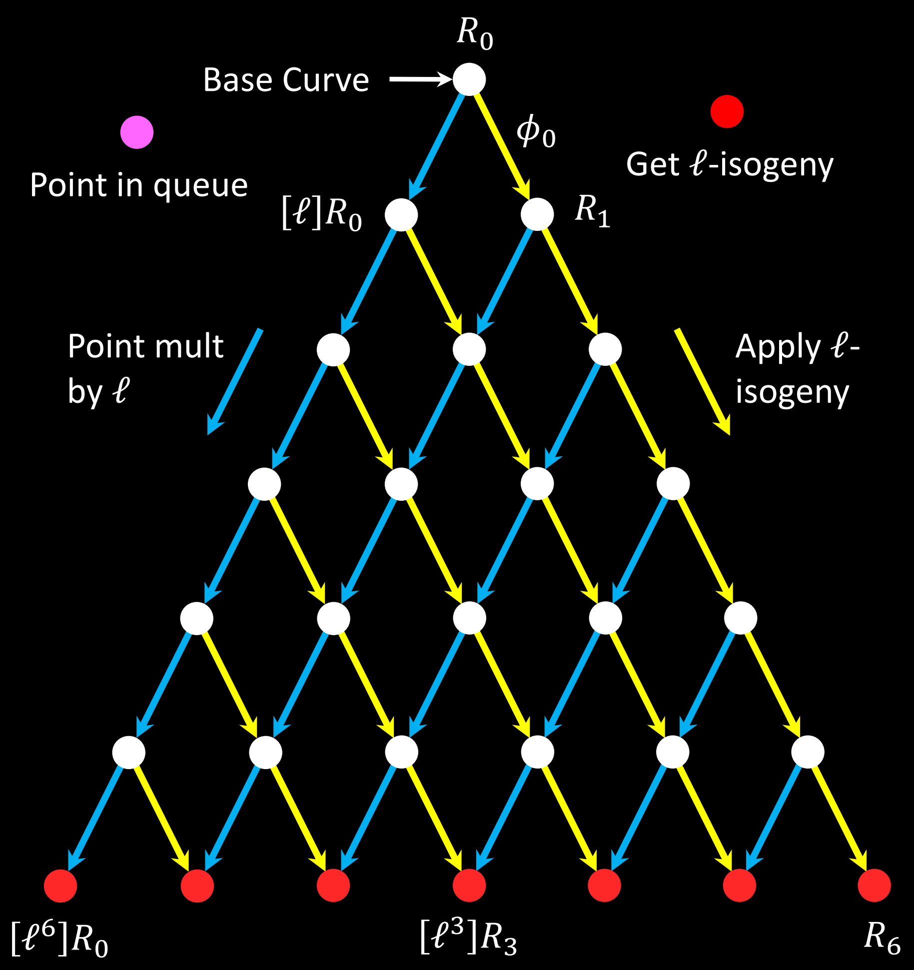


Order of $[\ell^5]R_0$ is ℓ^2

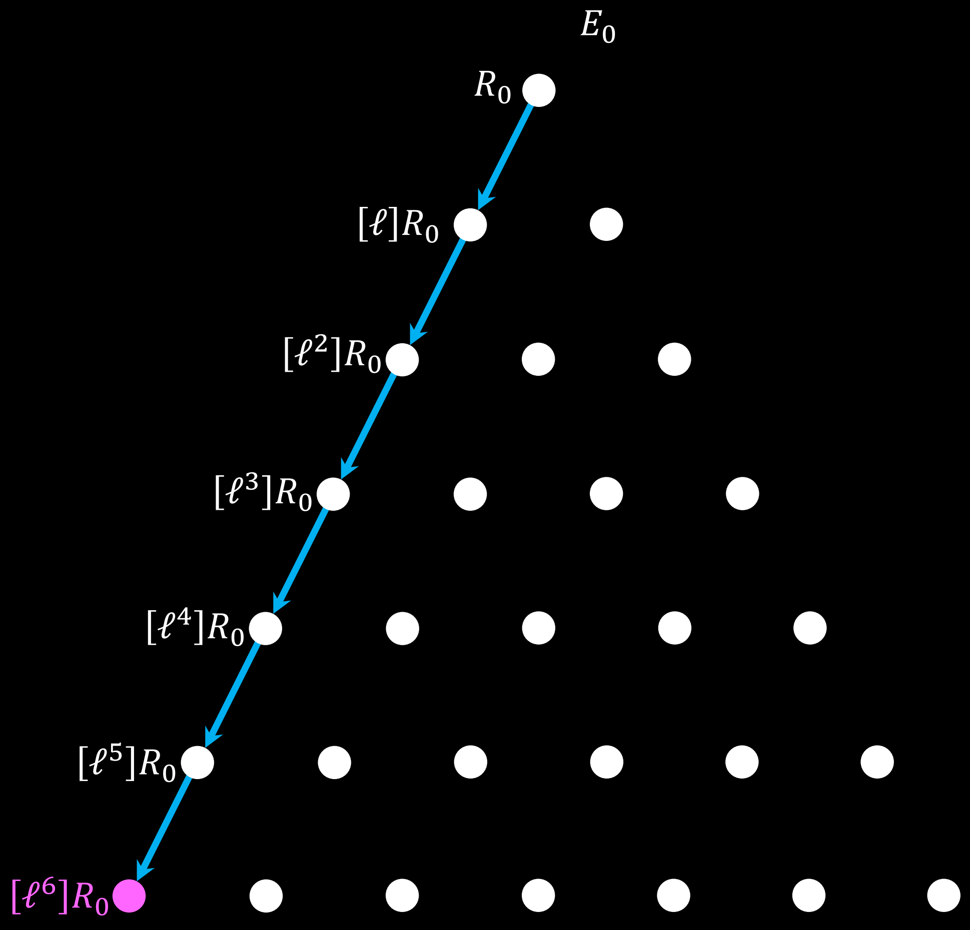


Large degree isogeny computations

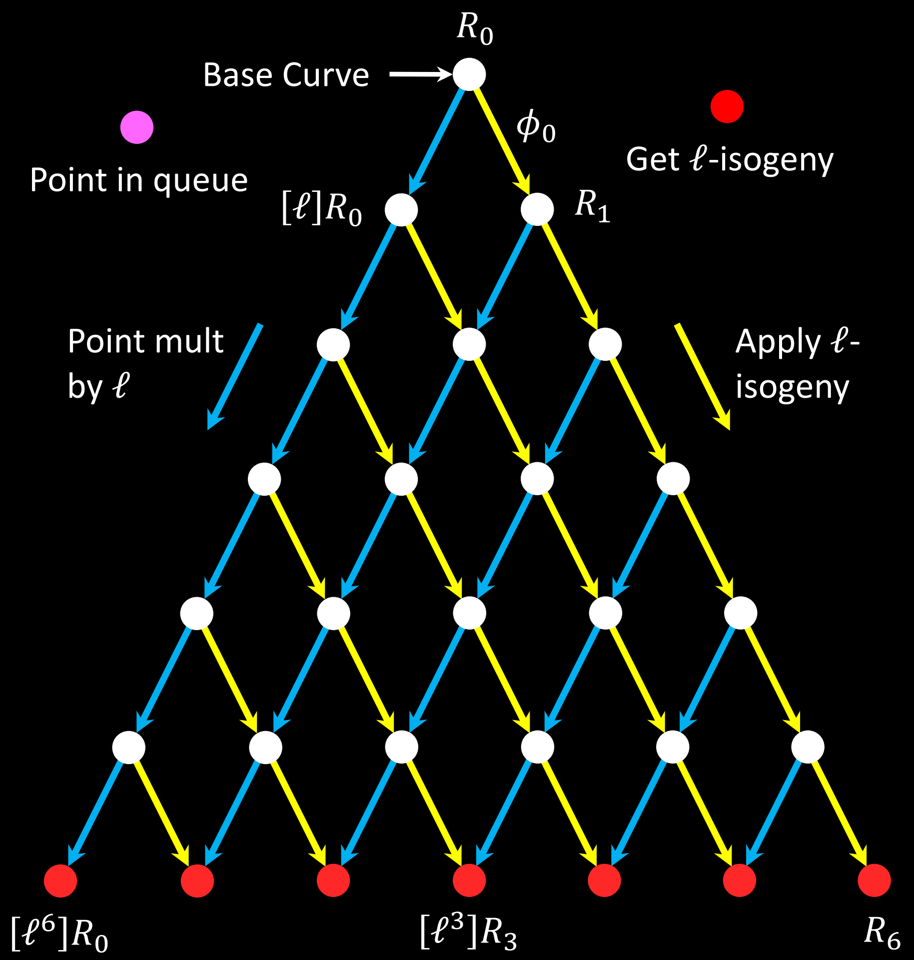
$e = 7$



Order of $[\ell^6]R_0$ is ℓ



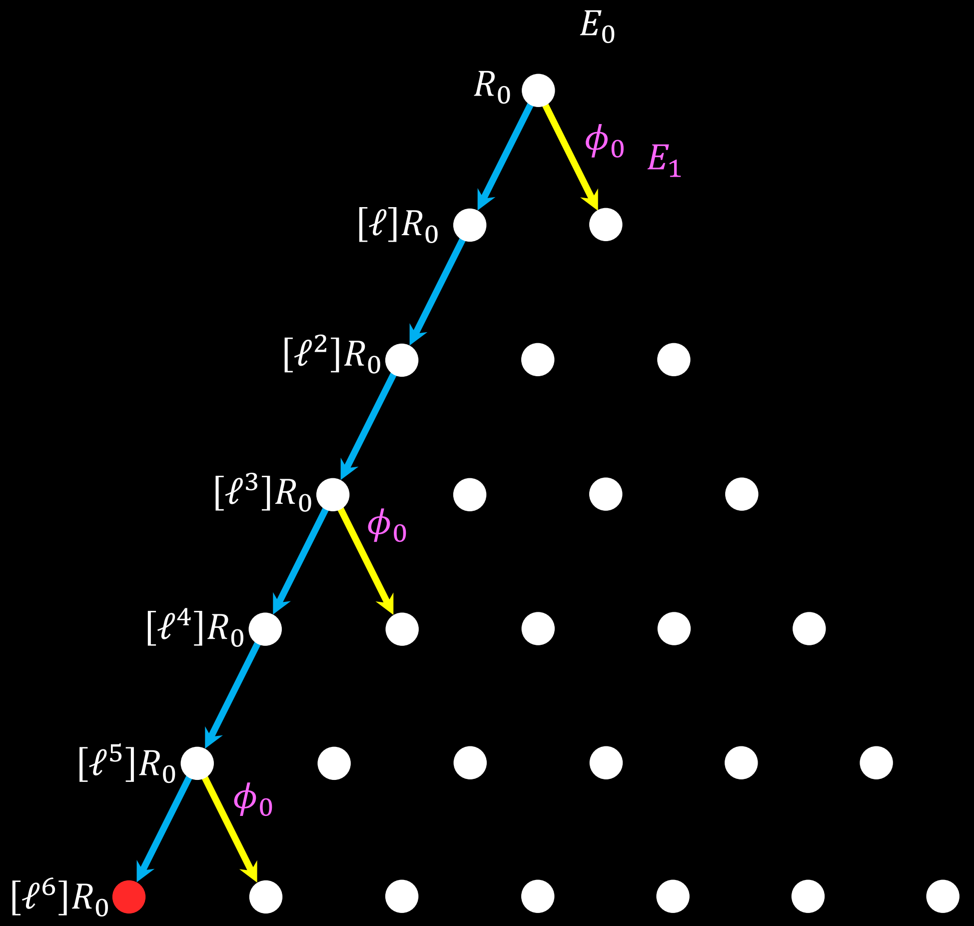
Large degree isogeny computations



$e = 7$

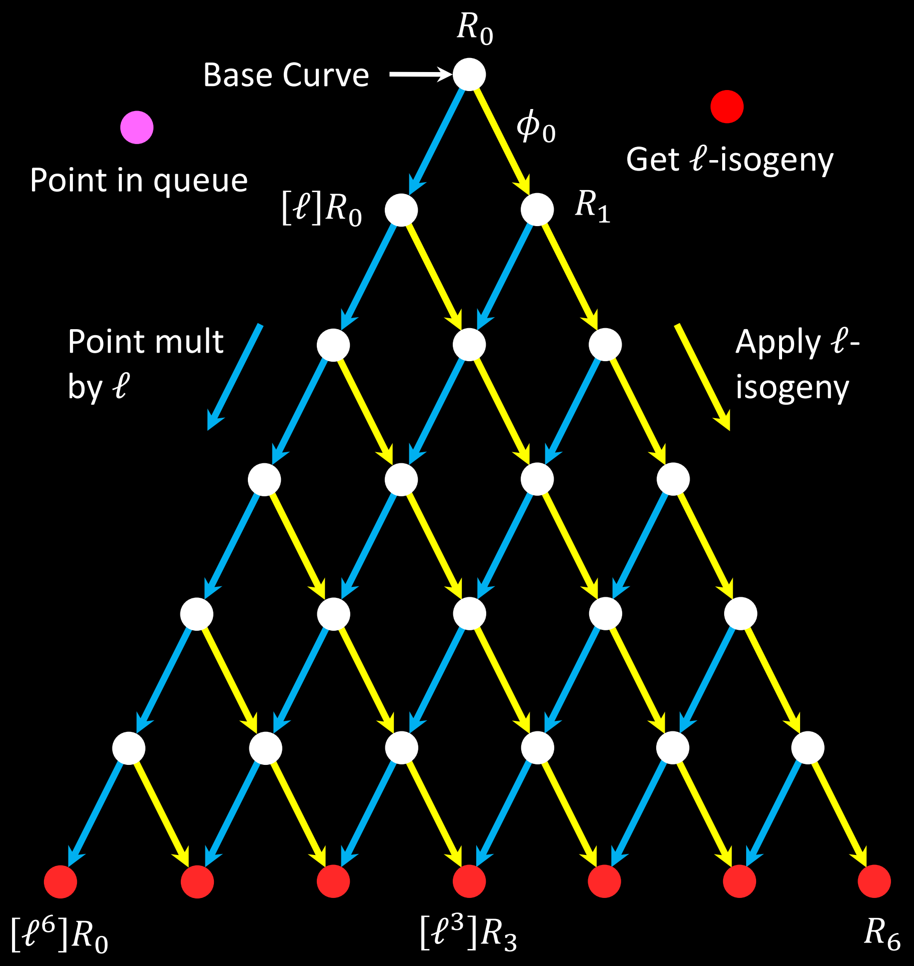
$\phi_0 \coloneqq E_0/\langle[\ell^6]R_0\rangle$

$E_1 = \phi_0(E_0)$



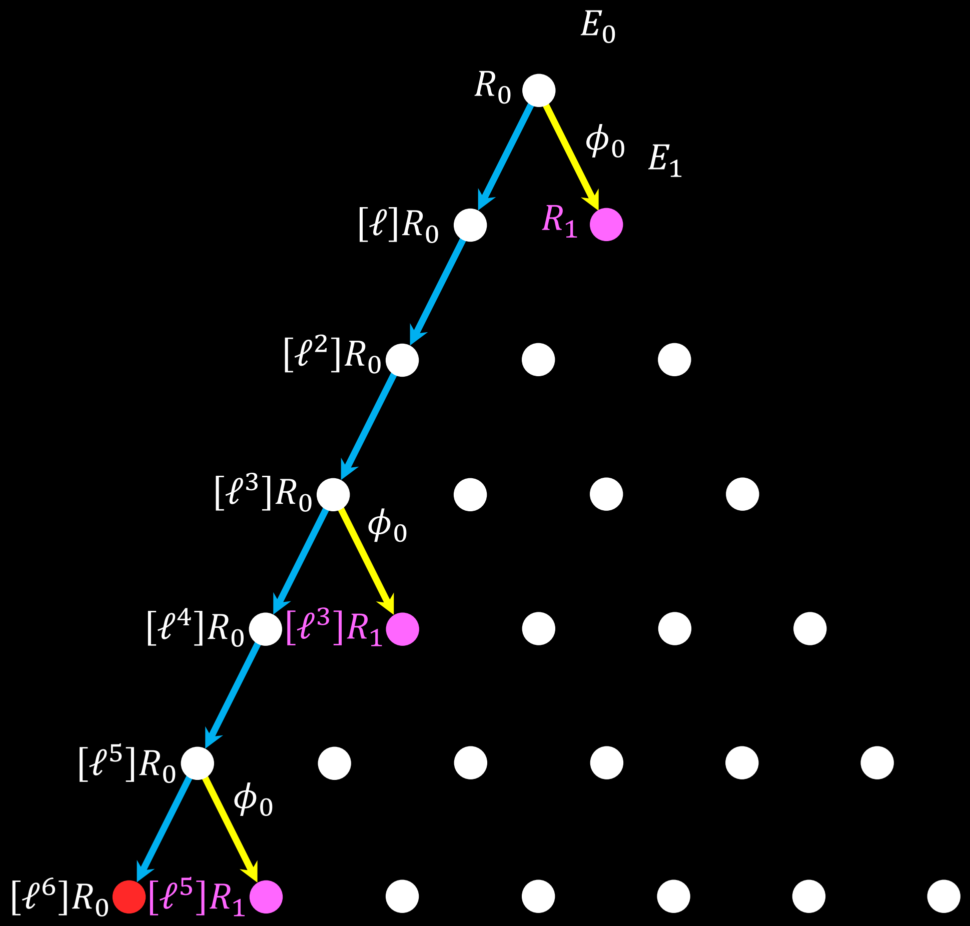
Large degree isogeny computations

$e = 7$

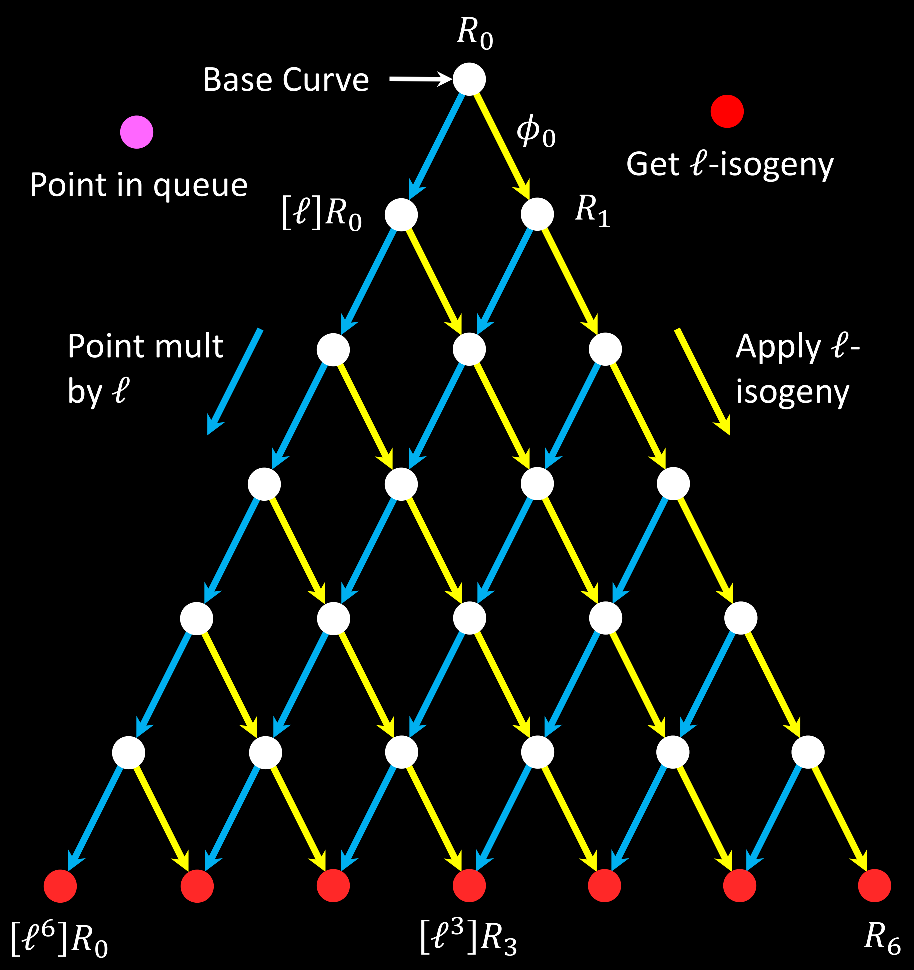


$R_1 = \phi_0(R_0)$

Order of $[\ell^5]R_1$ is ℓ



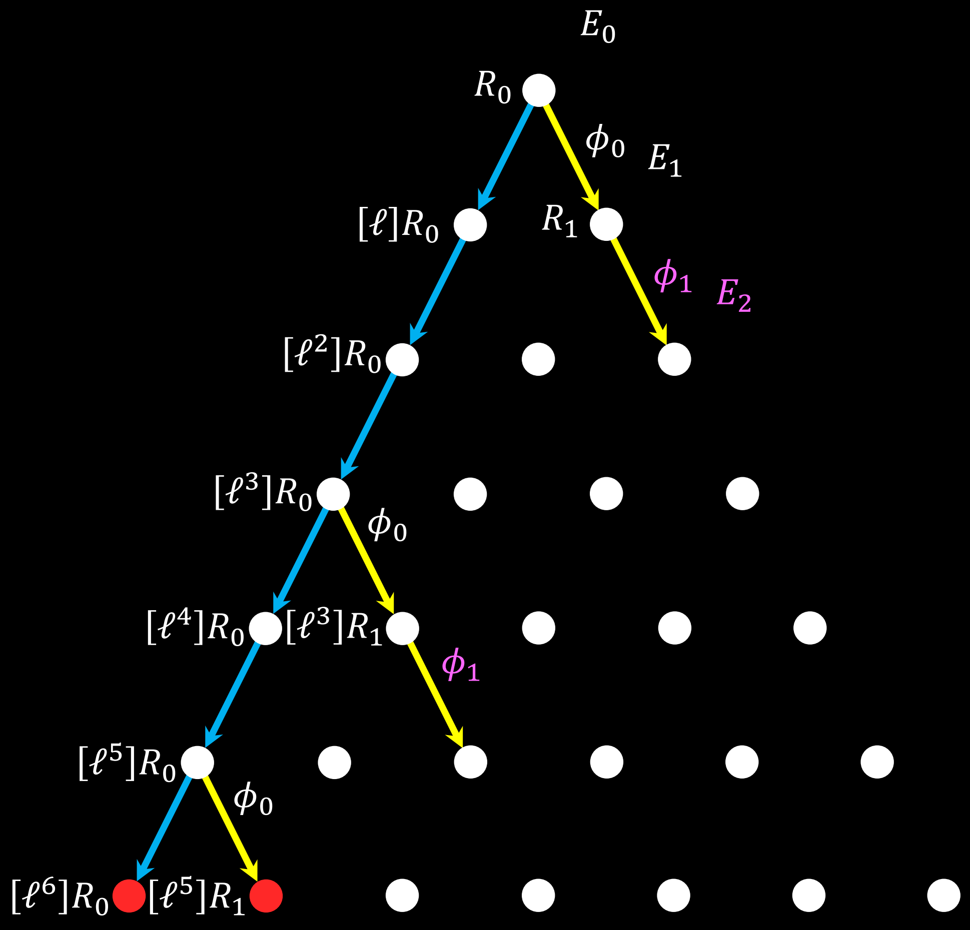
Large degree isogeny computations



$e = 7$

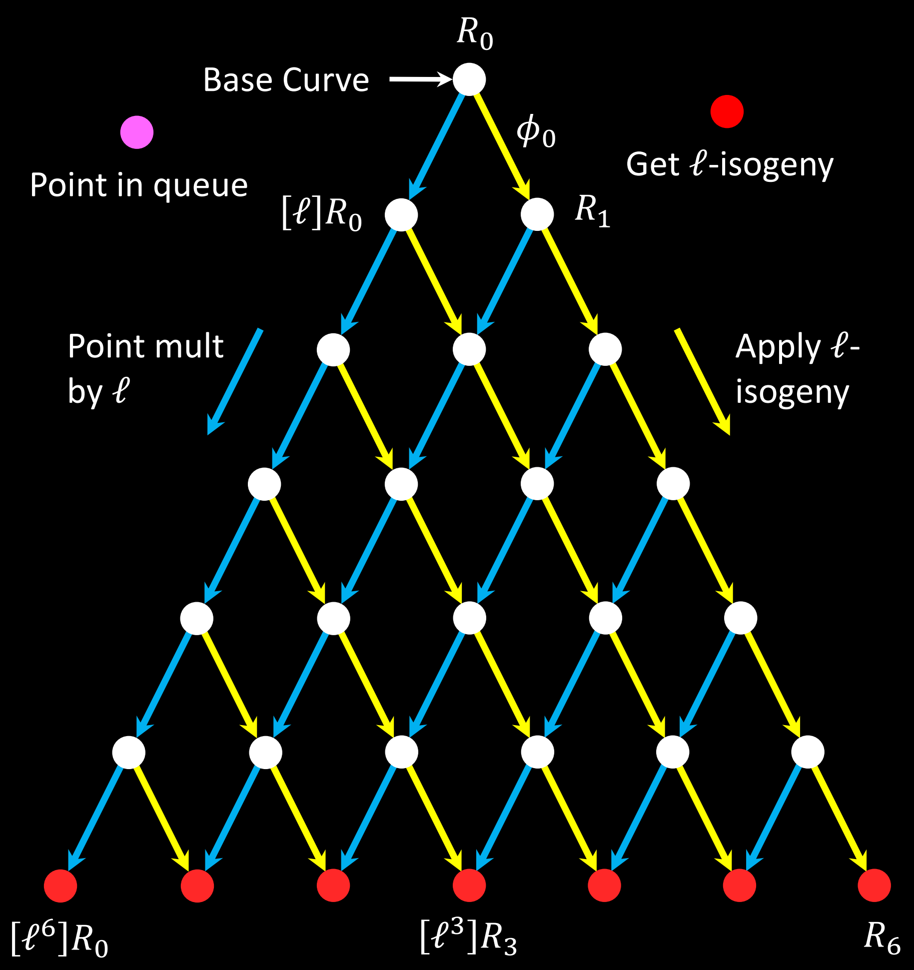
$\phi_1 \coloneqq E_1/\langle[\ell^5]R_1\rangle$

$E_2 = \phi_1(E_1)$



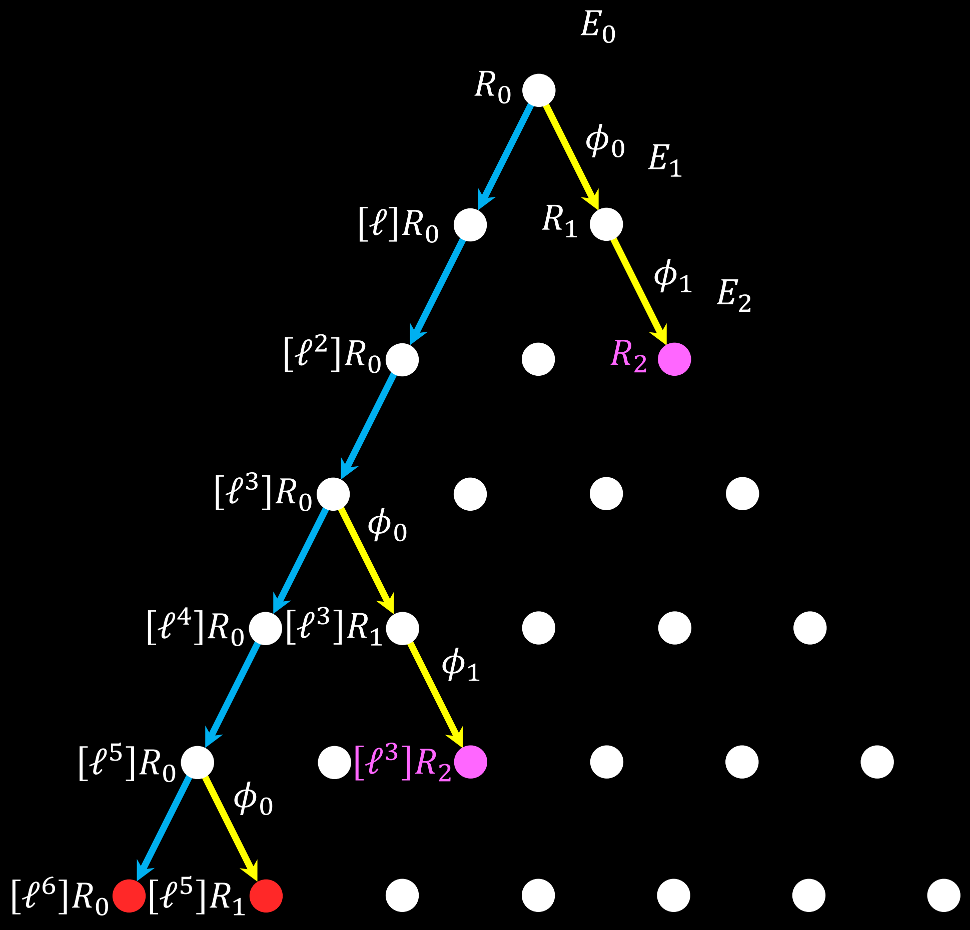
Large degree isogeny computations

$e = 7$

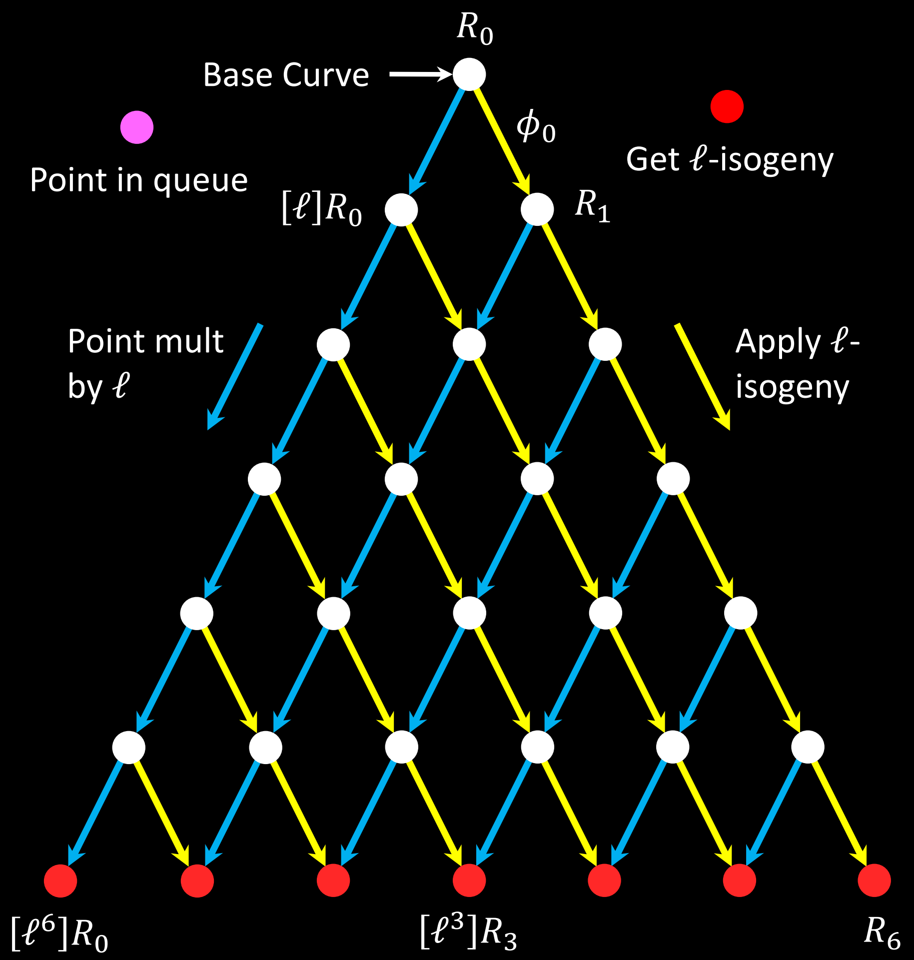


$R_2 = \phi_1(R_1)$

Order of $[\ell^3]R_2$ is ℓ^2

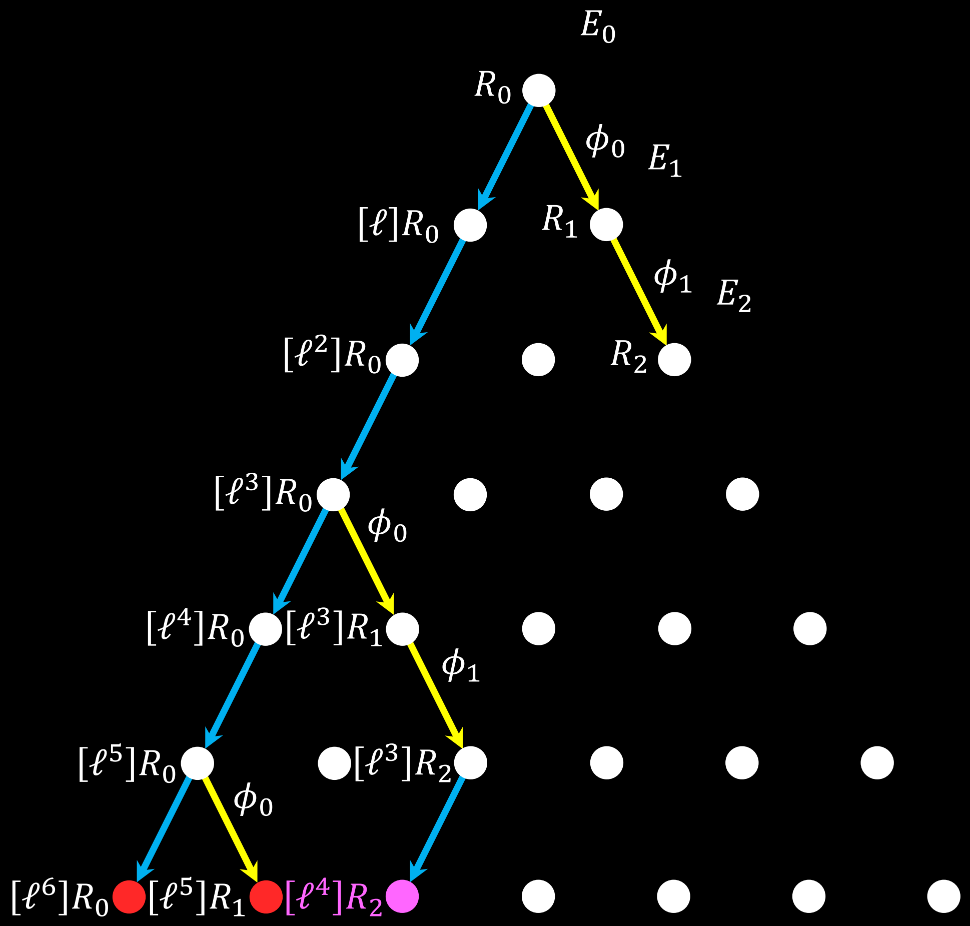


Large degree isogeny computations



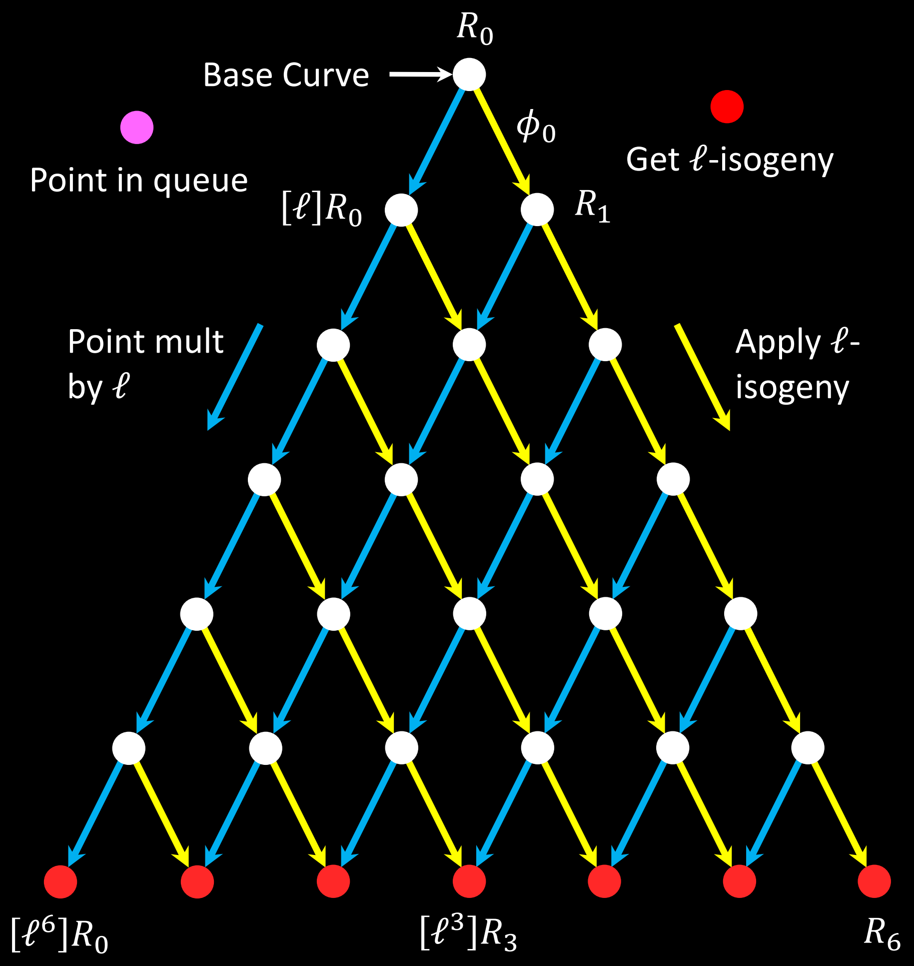
$e = 7$

Order of $[\ell^4]R_2$ is ℓ



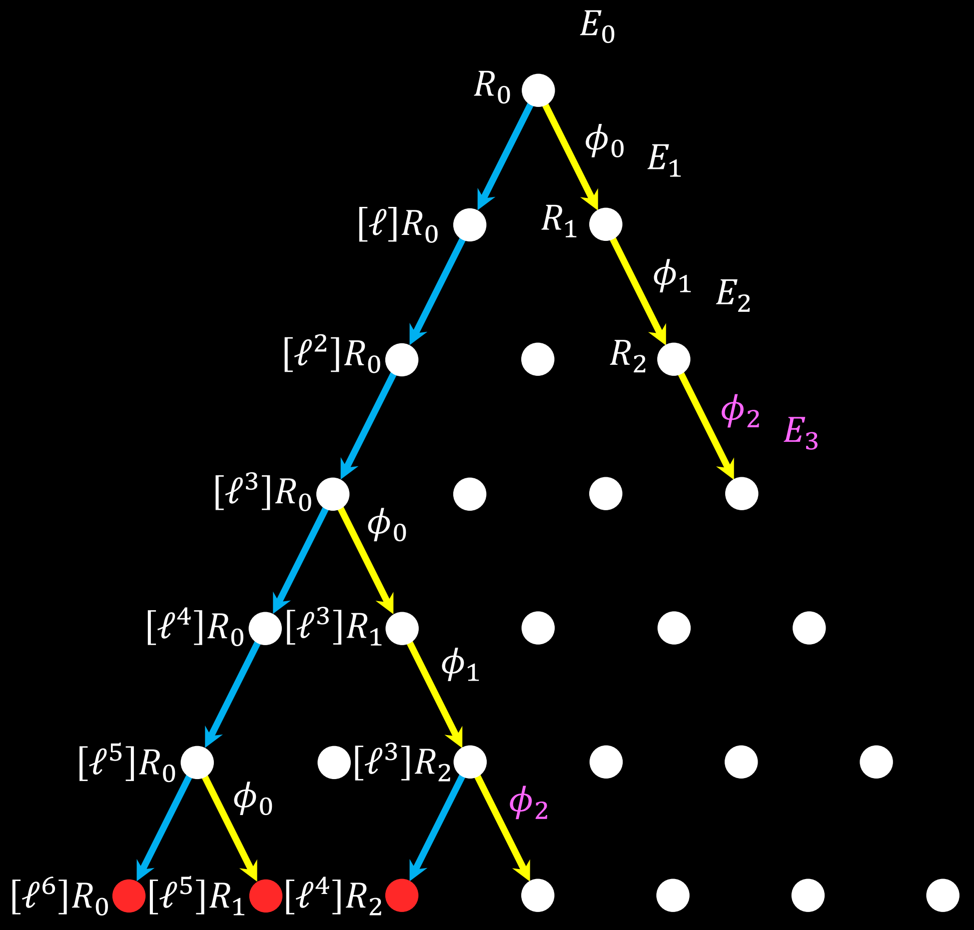
Large degree isogeny computations

$e = 7$



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

$E_3 = \phi_2(E_2)$

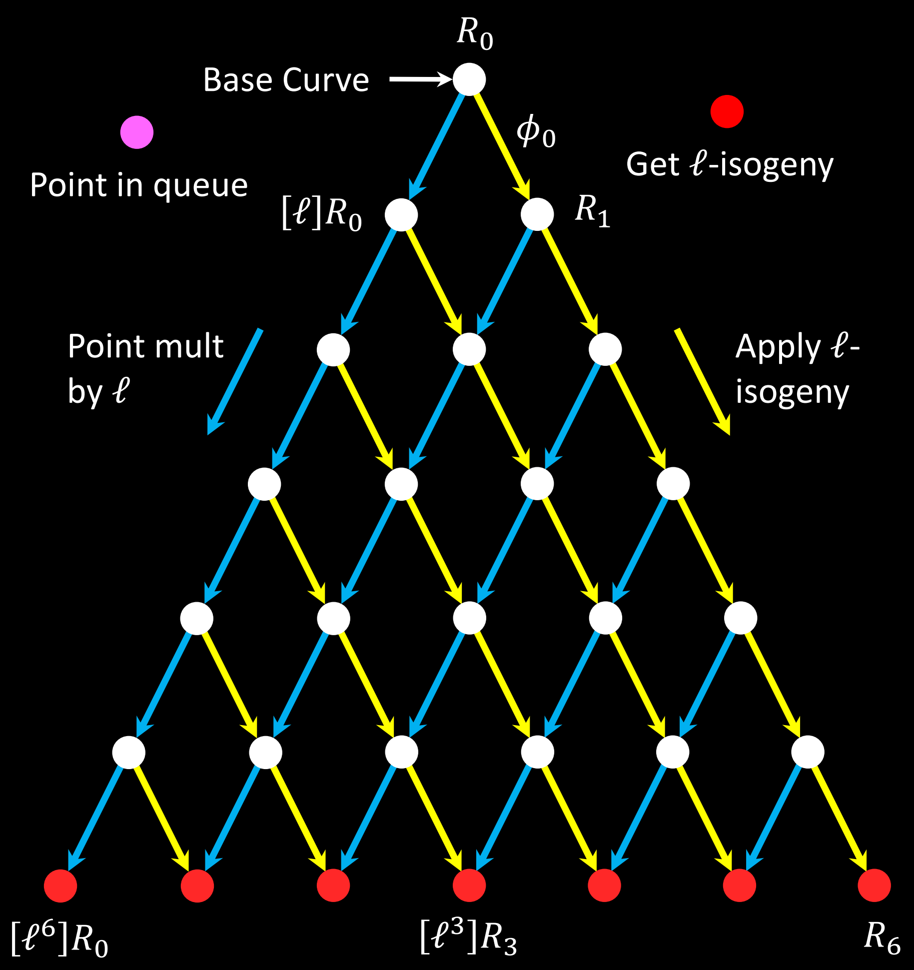


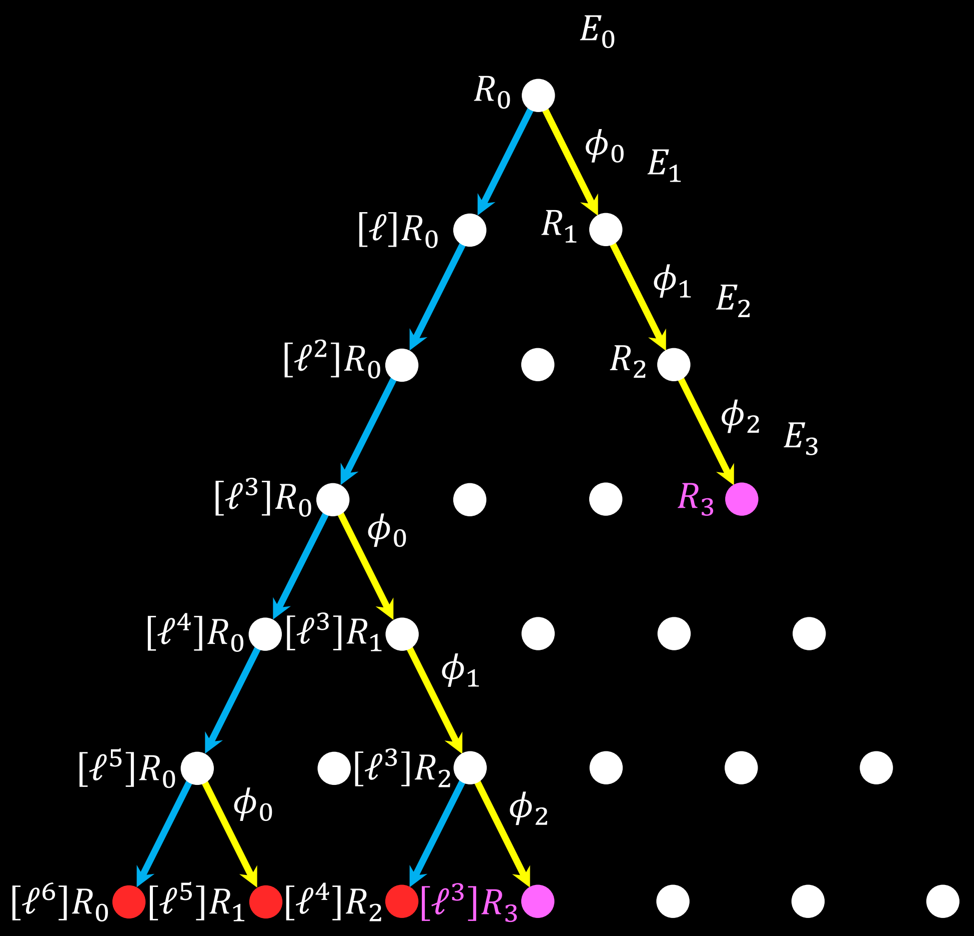
Large degree isogeny computations

$e = 7$

$R_3 = \phi_2(R_2)$

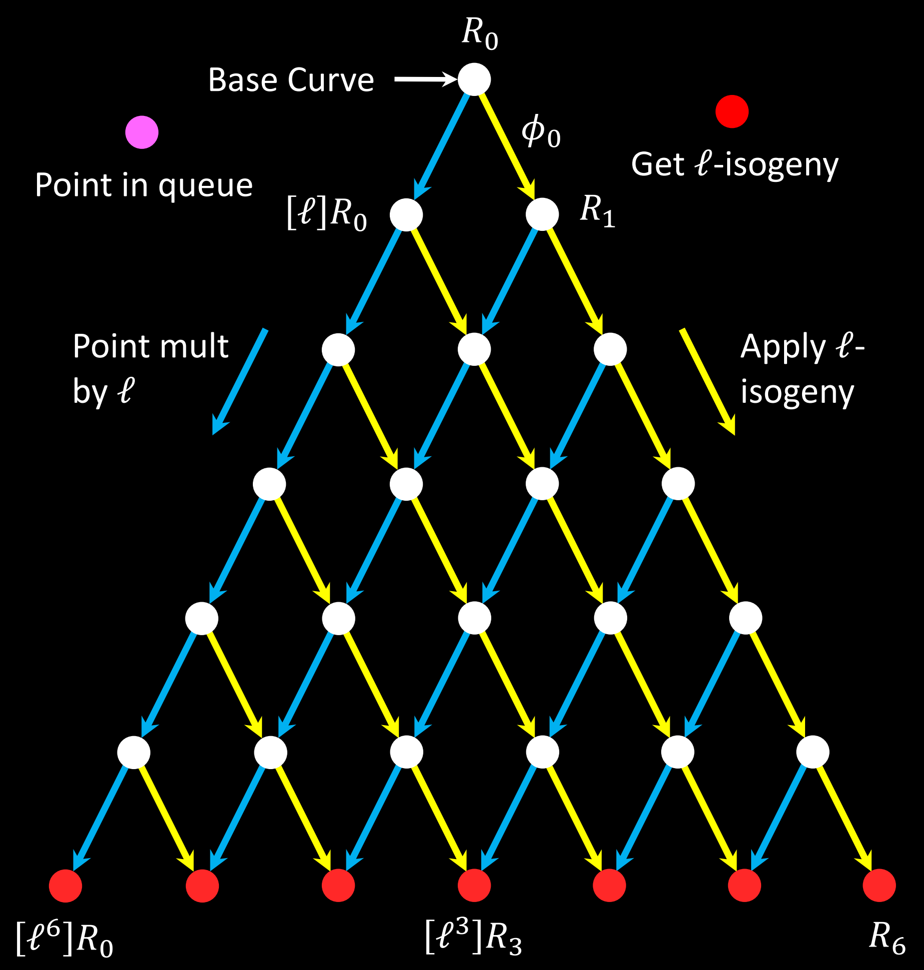
Order of $[\ell^3]R_3$ is ℓ





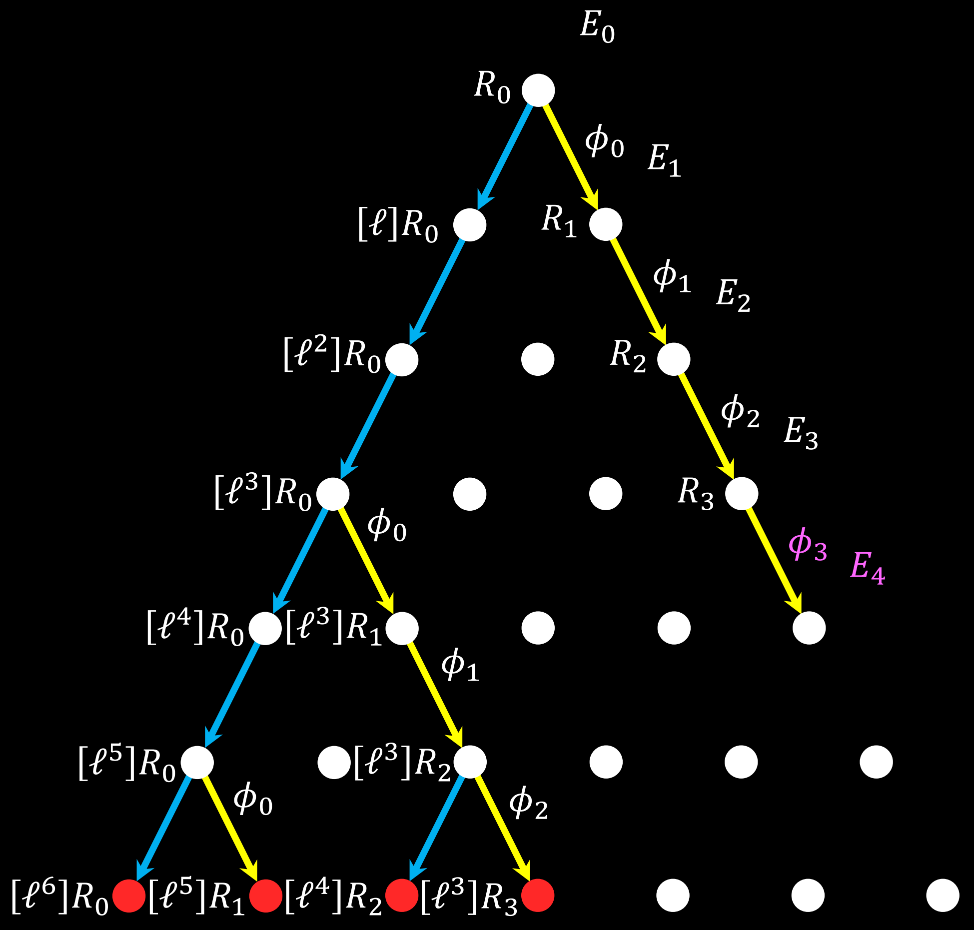
Large degree isogeny computations

$e = 7$

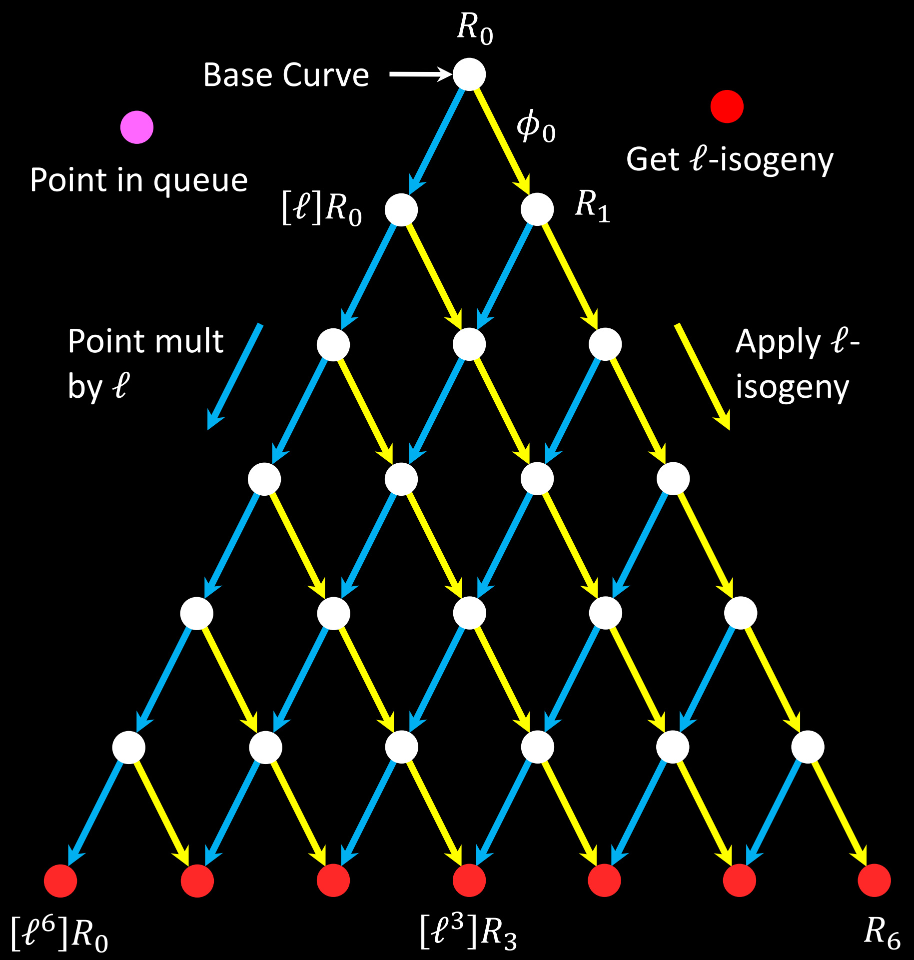


$\phi_3 \coloneqq E_3/\langle[\ell^3]R_3\rangle$

$E_4 = \phi_3(E_3)$



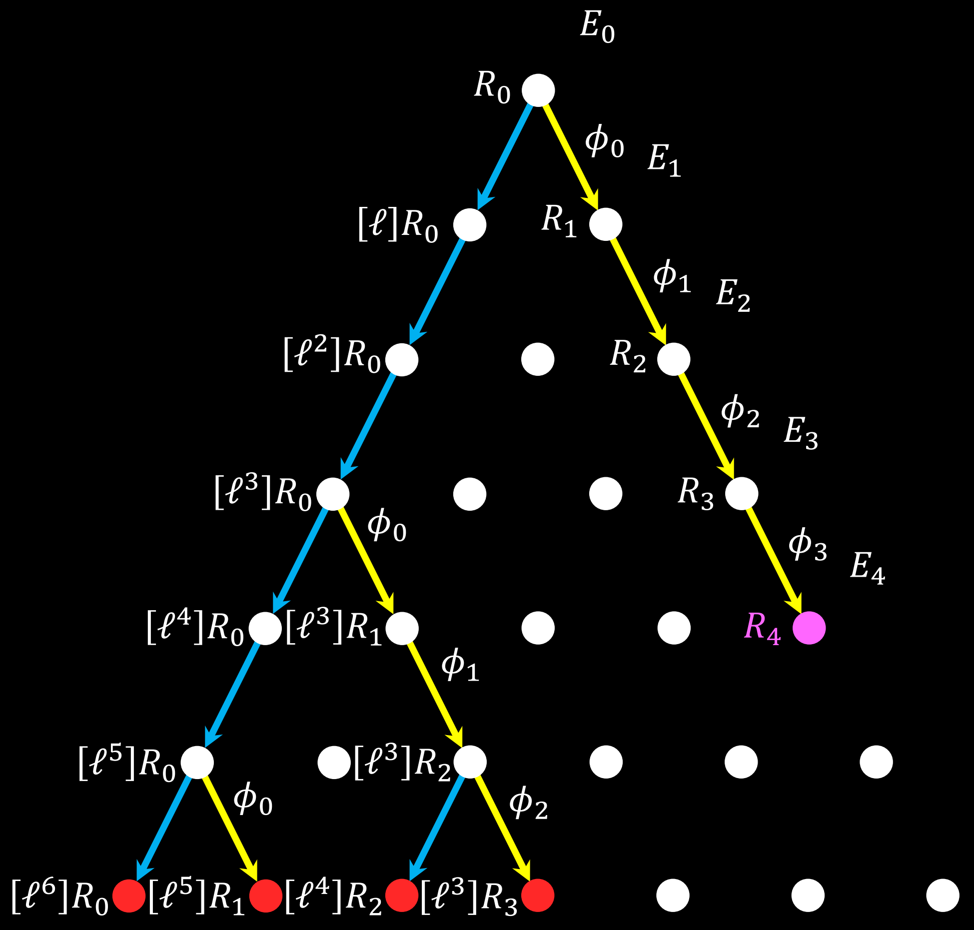
Large degree isogeny computations



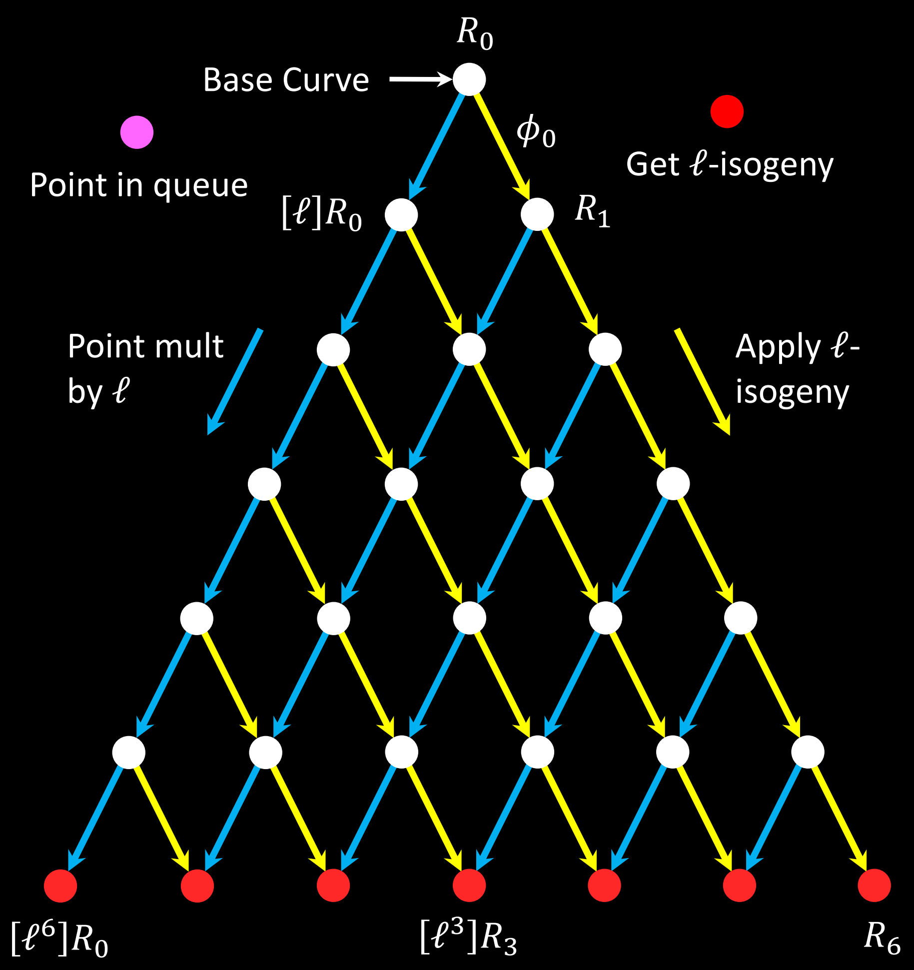
$e = 7$

$R_4 = \phi_3(R_3)$

Order of R_4 is ℓ^3

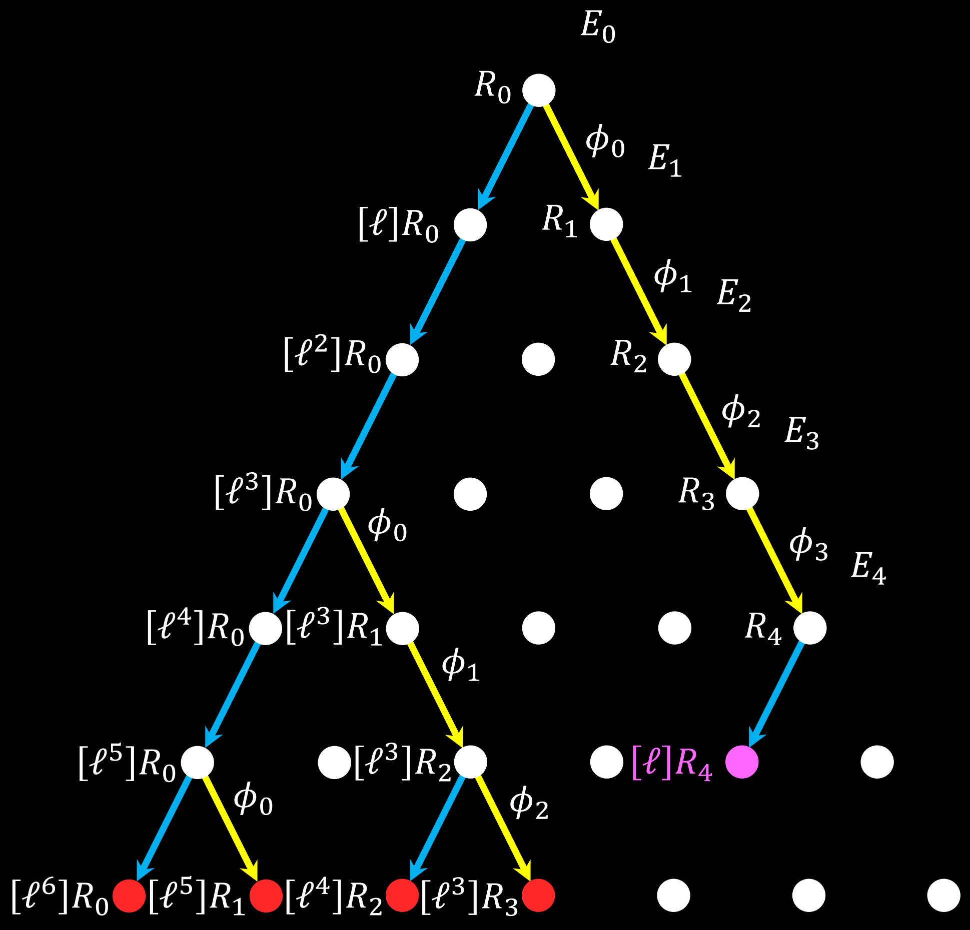


Large degree isogeny computations



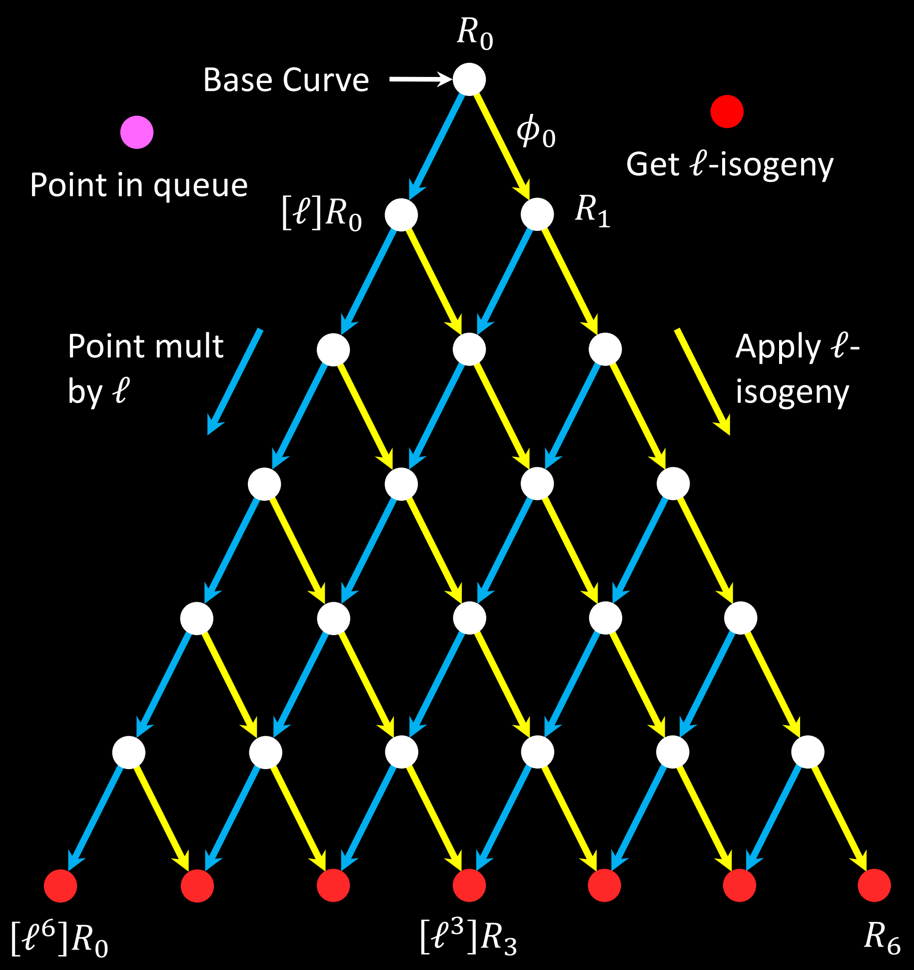
$e = 7$

Order of $[\ell]R_4$ is ℓ^2

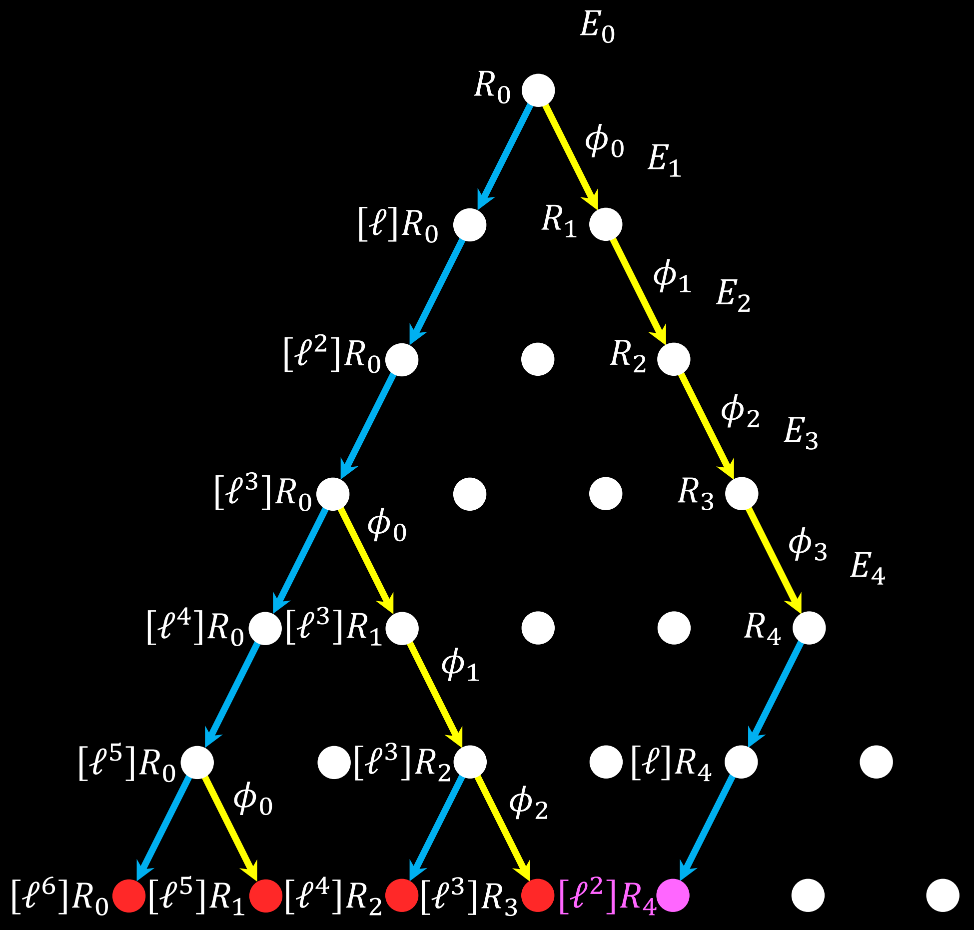


Large degree isogeny computations

$e = 7$

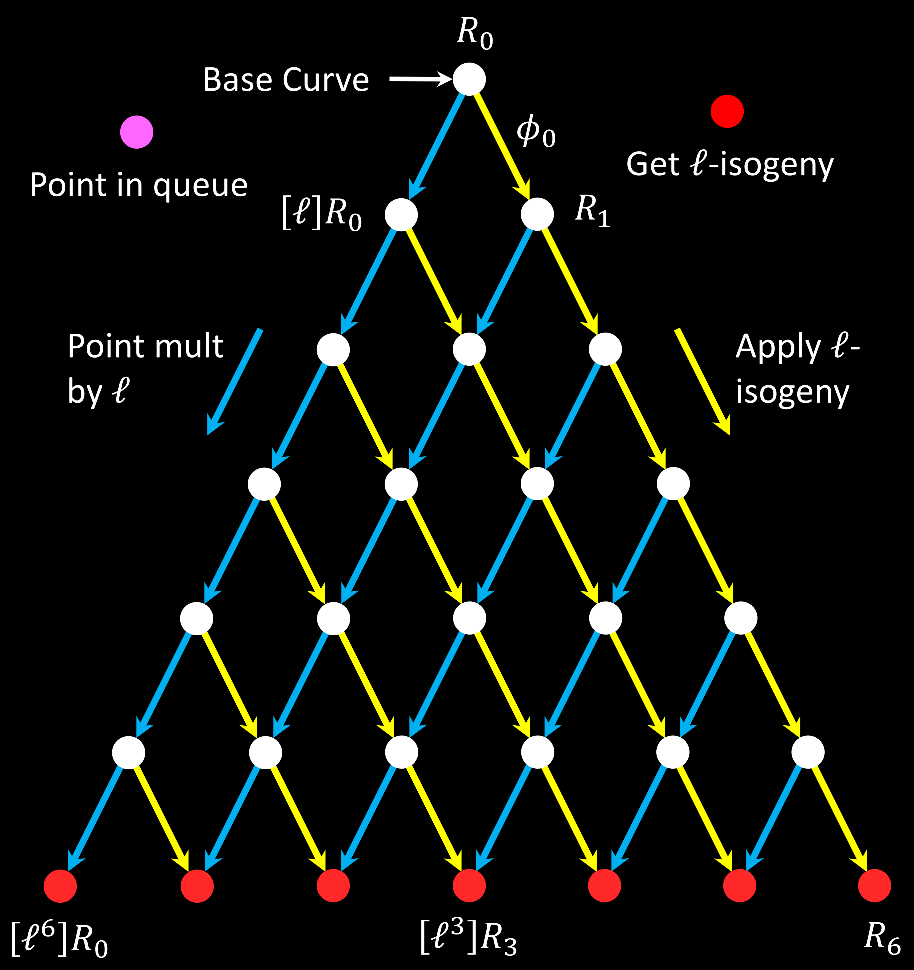


Order of $[\ell^2]R_4$ is ℓ



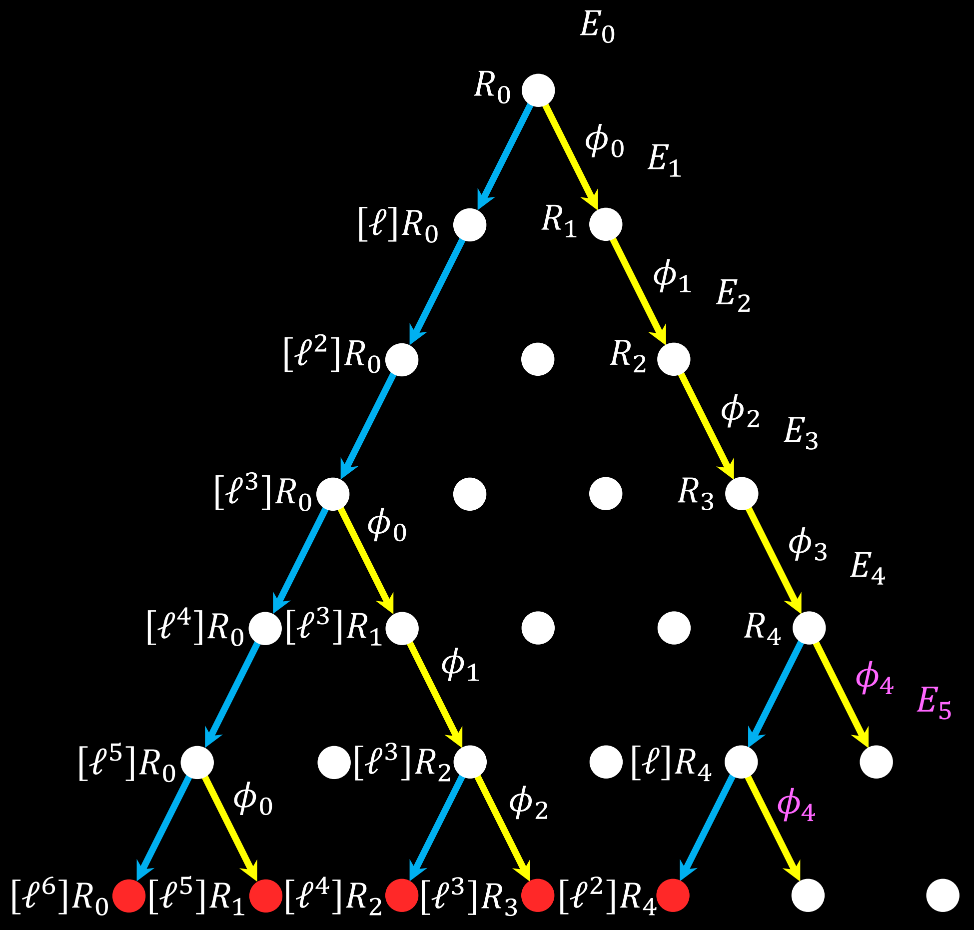
Large degree isogeny computations

$e = 7$

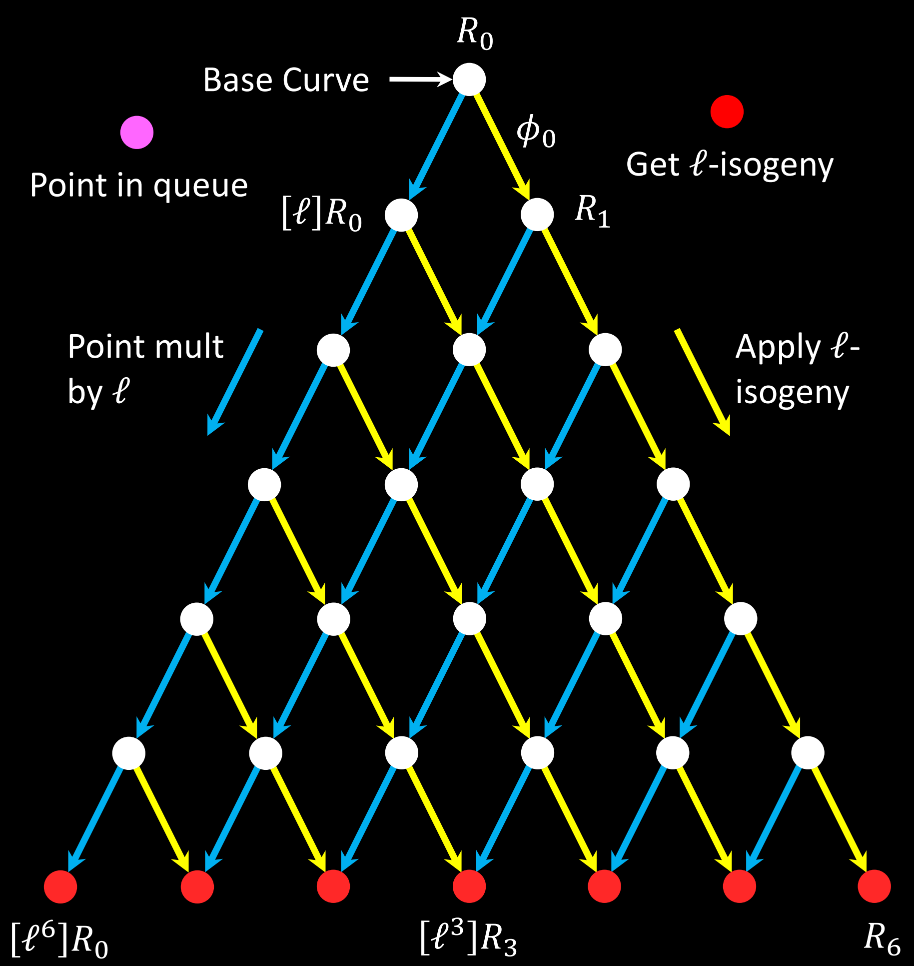


$\phi_4 \coloneqq E_4/\langle[\ell^2]R_2\rangle$

$E_5 = \phi_4(E_4)$



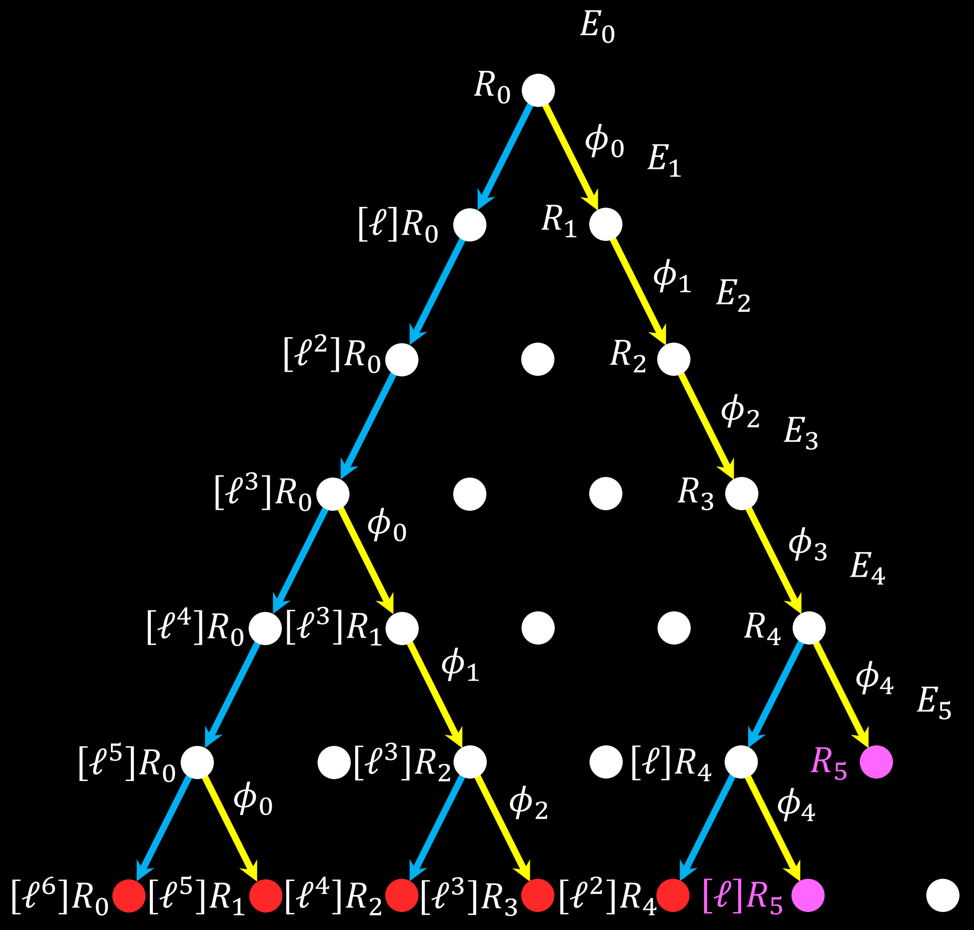
Large degree isogeny computations



$e = 7$

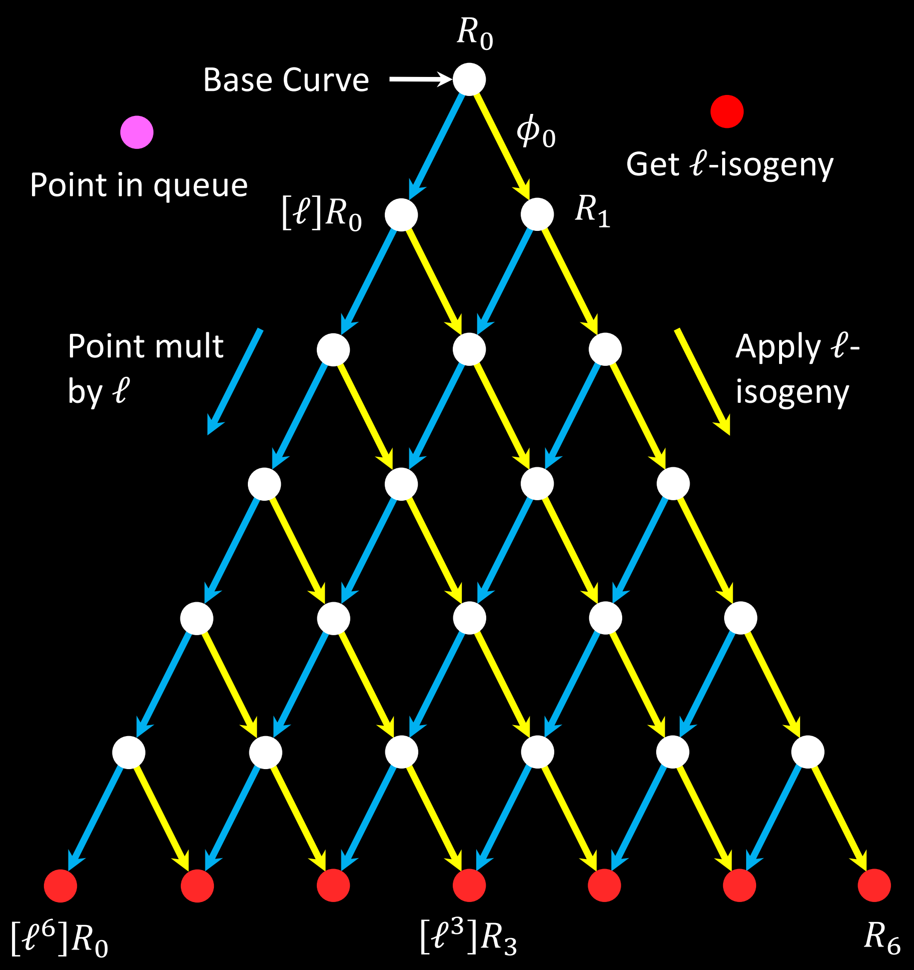
$R_5 = \phi_4(R_4)$

Order of $[\ell]R_5$ is ℓ



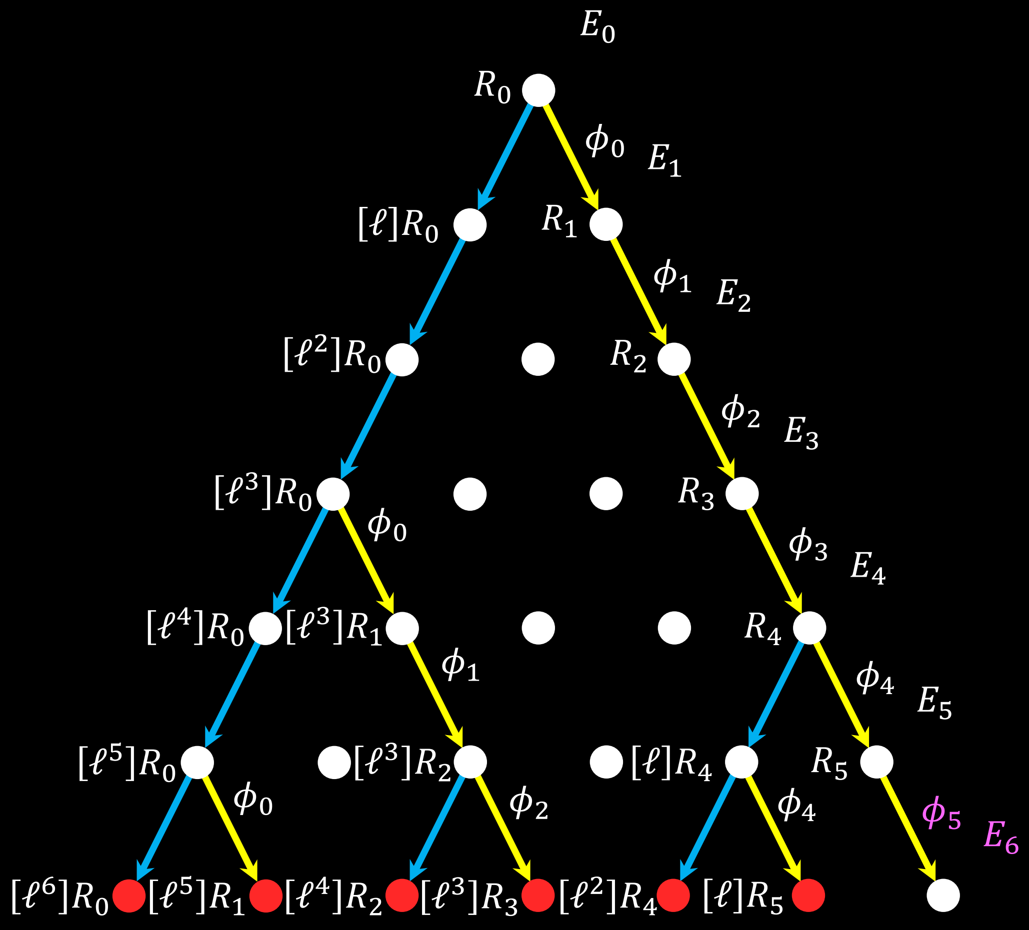
Large degree isogeny computations

$e = 7$



$\phi_5 \coloneqq E_5/\langle[\ell]R_5\rangle$

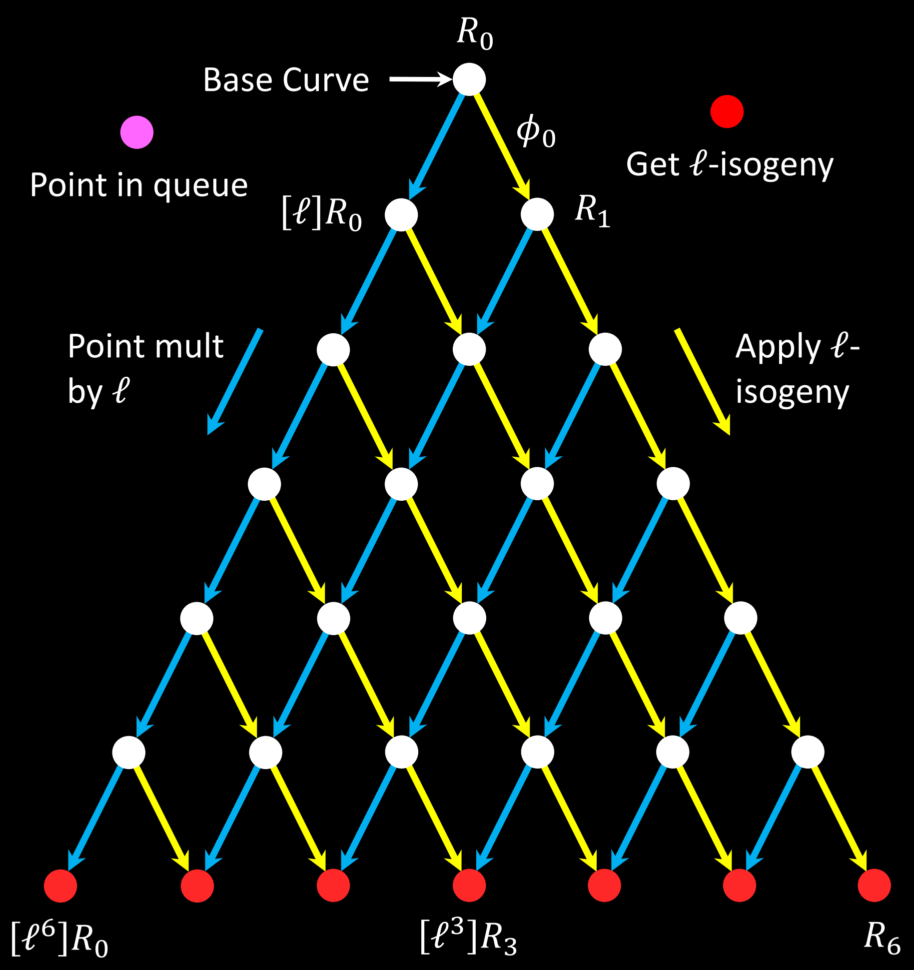
$E_6 = \phi_5(E_5)$

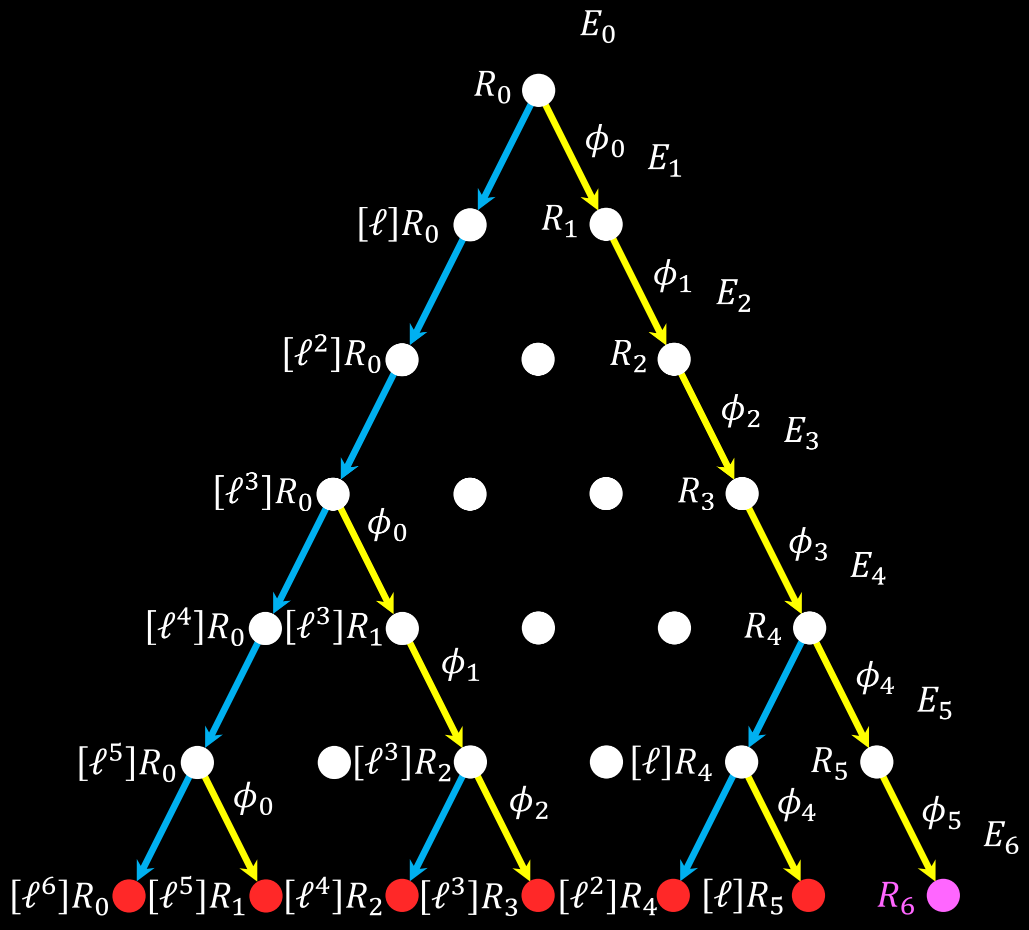


Large degree isogeny computations

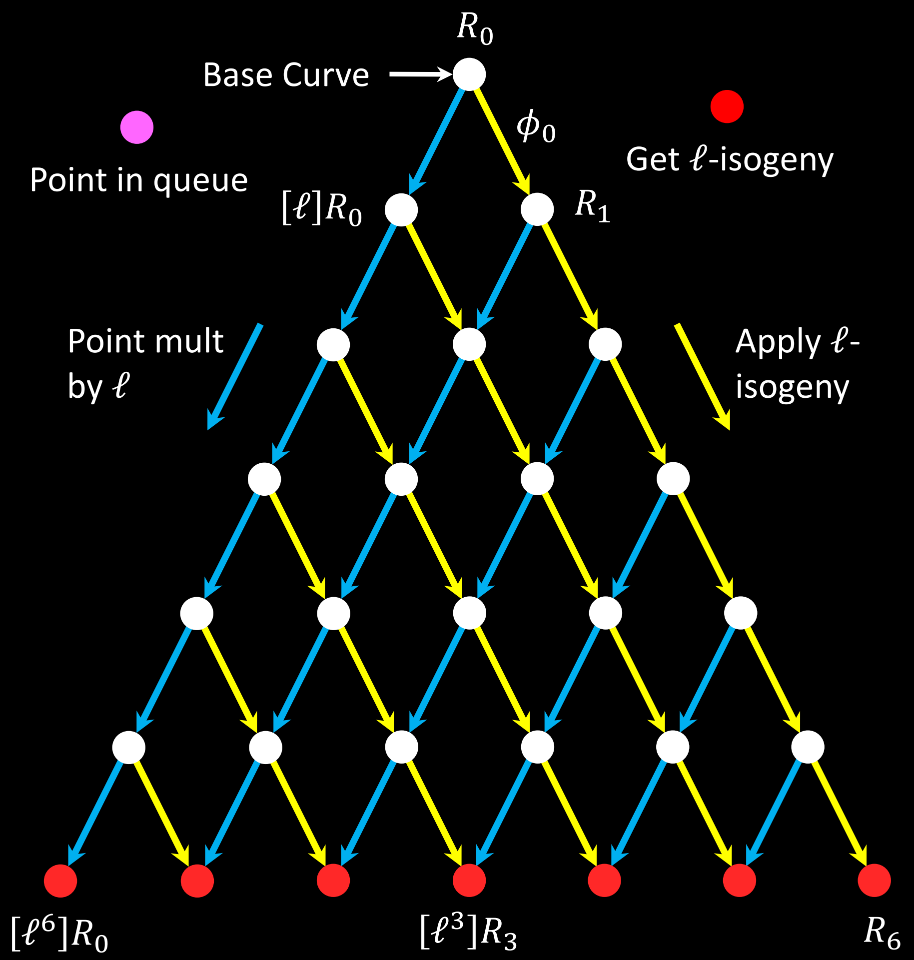
$e = 7$

Order of R_6 is ℓ





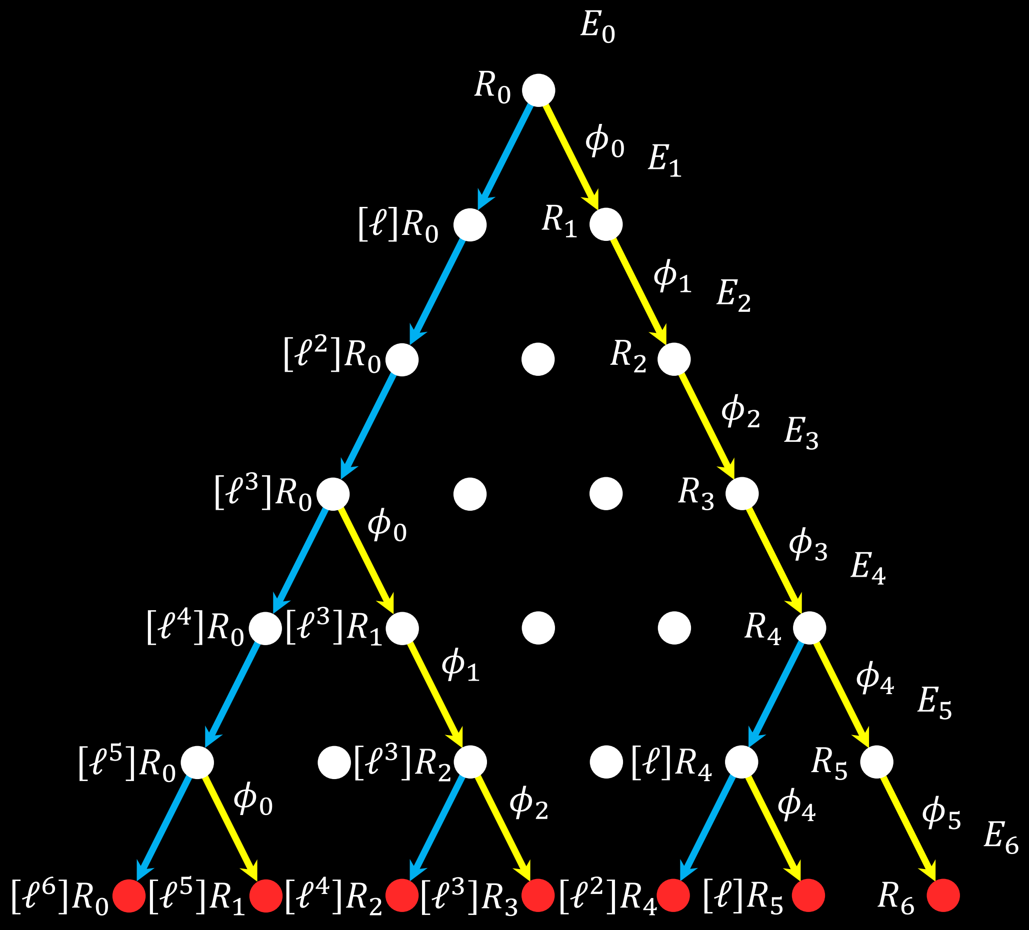
Large degree isogeny computations



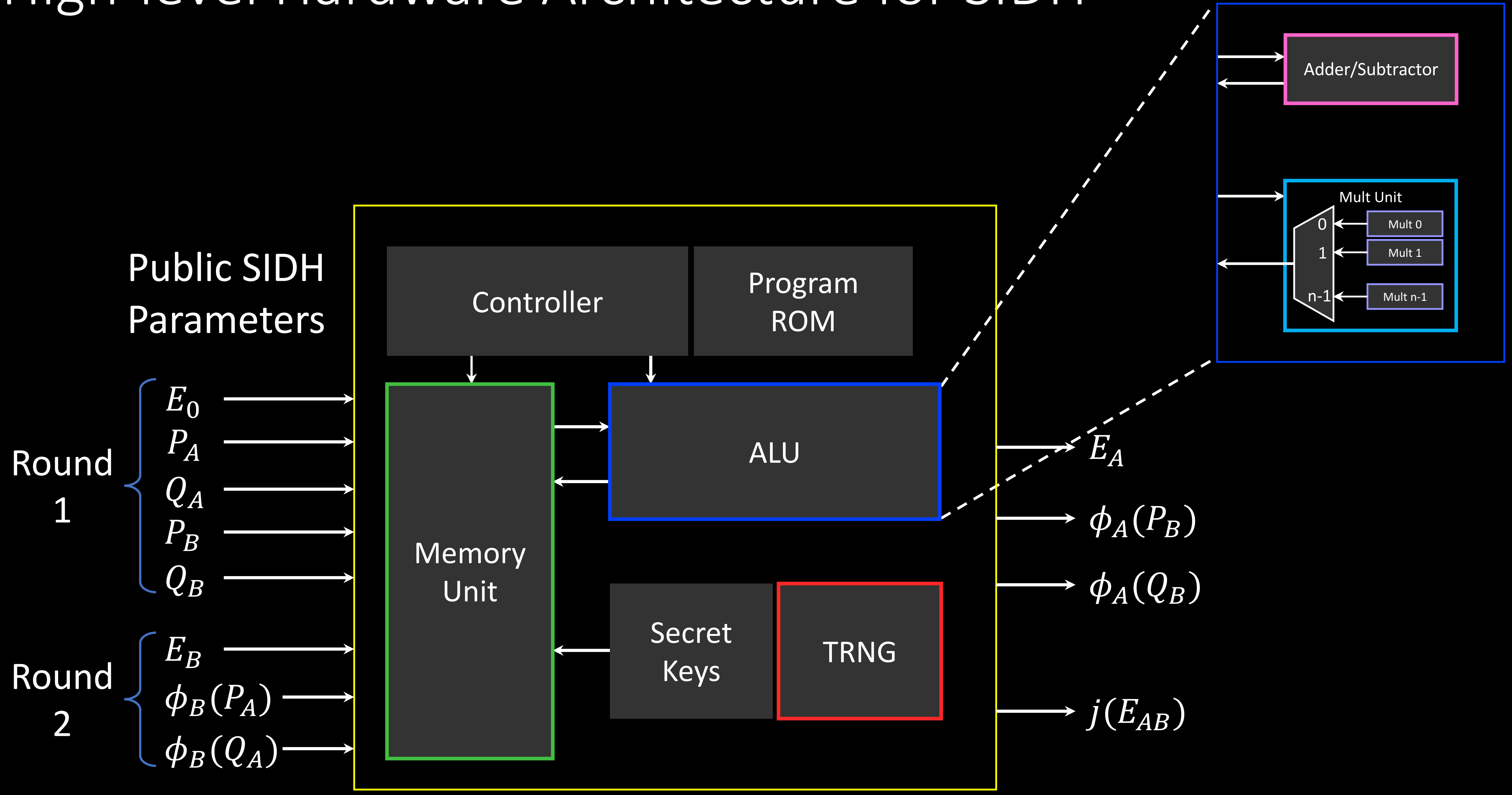
$e = 7$

$\phi_6 \coloneqq E_6/\langle R_6 \rangle$

$E_7 = \phi_6(E_6)$

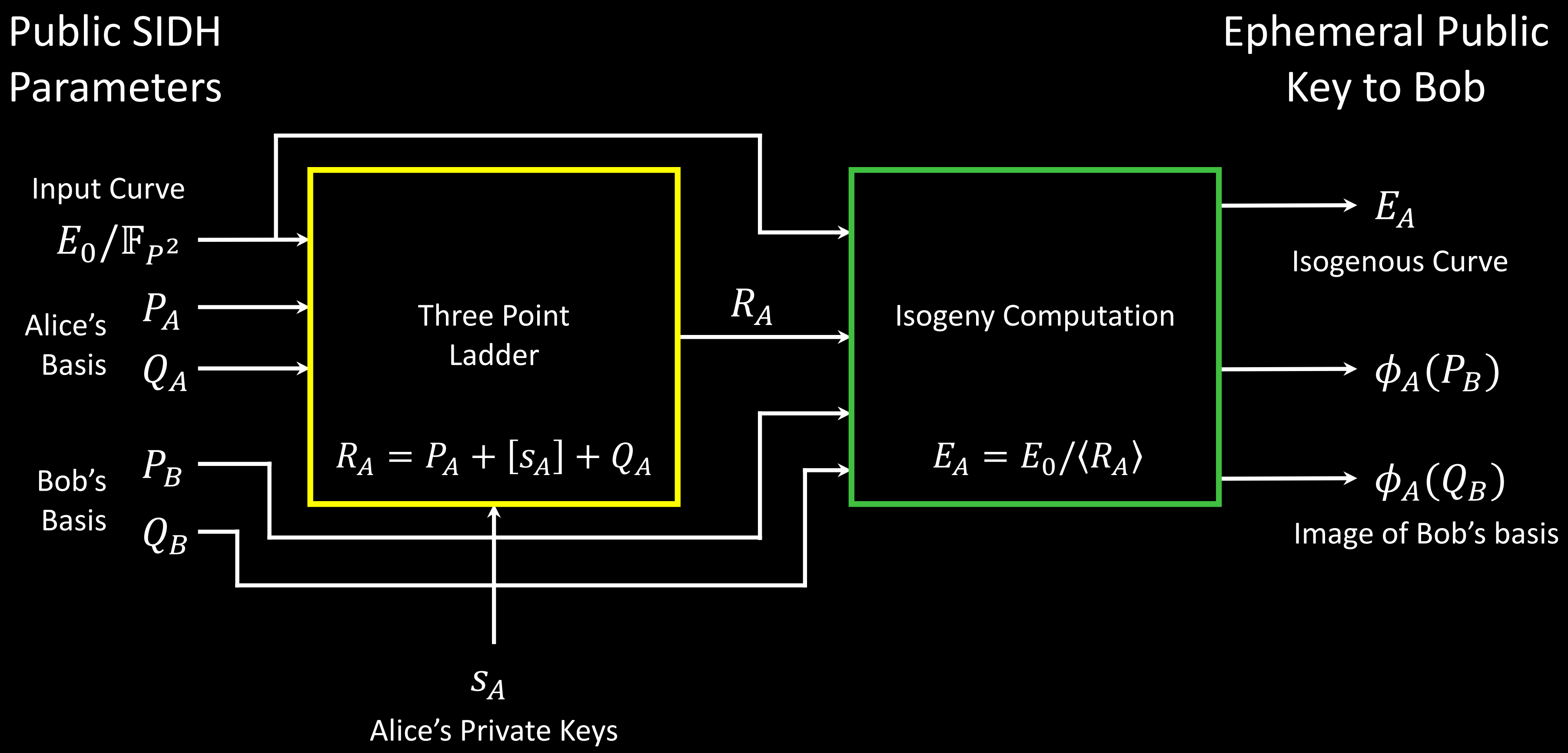


High-level Hardware Architecture for SIDH



Fast Kernel Computations

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



Field Multiplication

- 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 ($2^{e_A} \cdot 3^{e_B} - 1$), $p = 1 \ldots \underbrace{111 \ldots 111}_{e_A}$ and $p' = -p^{-1} = 1 \ (mod\ 2^w)$ where $w \leq e_A$

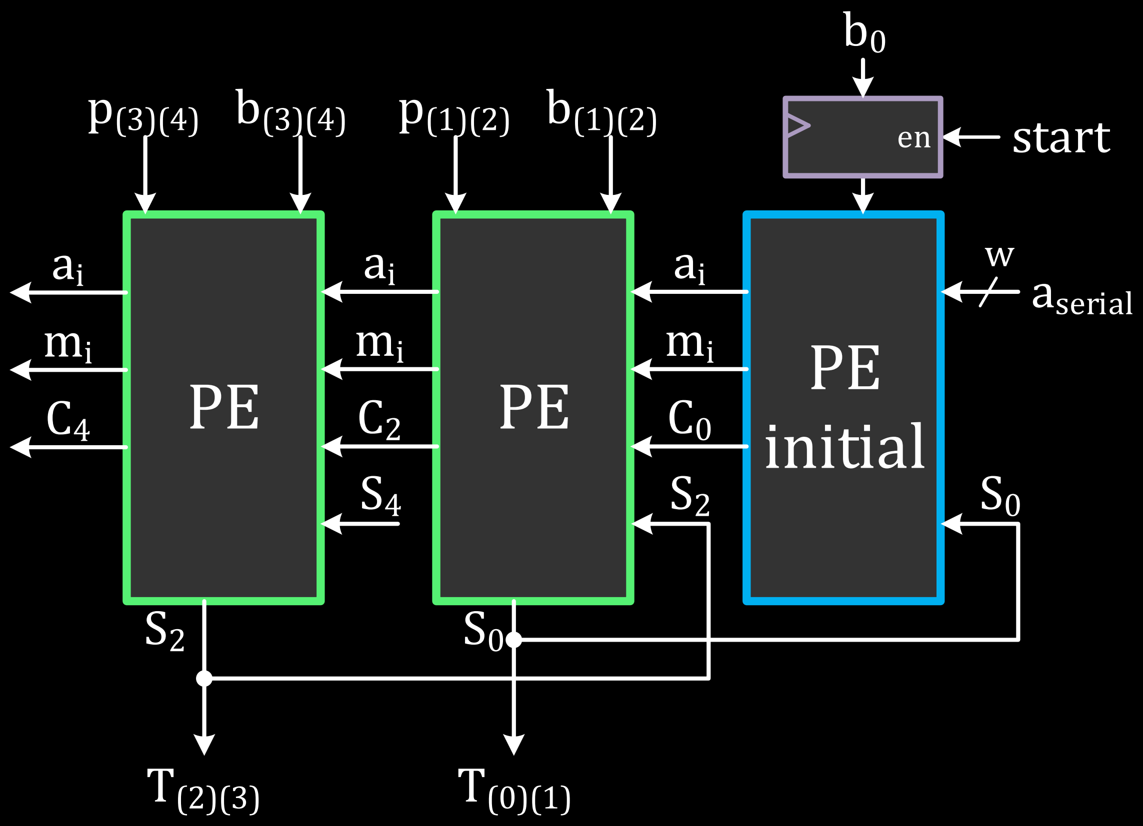
Coarsely Integrated Operand Scanning (CIOS):

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

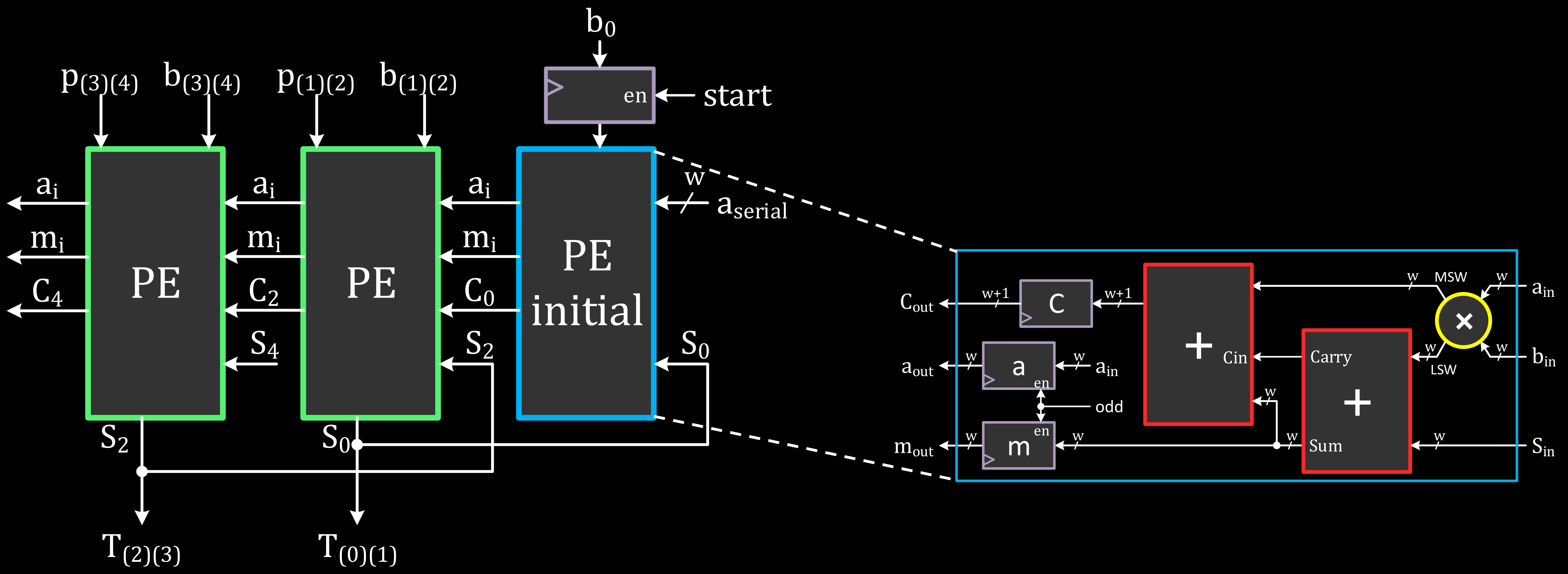
Finely Integrated Operand Scanning (FIOS):

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

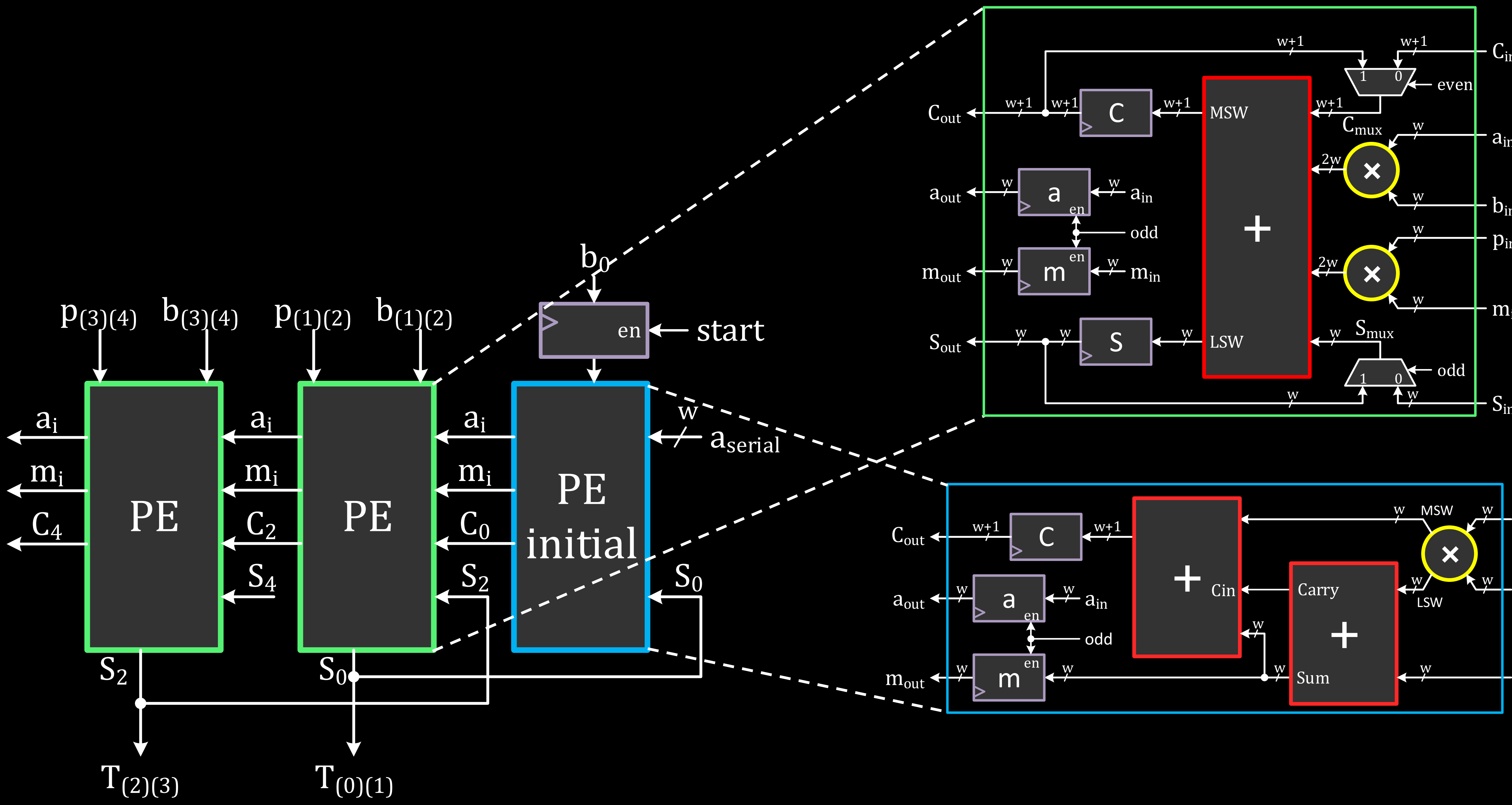
FIOS Design (Number of words = 4)



FIOS Design (Number of words = 4)



FIOS Design (Number of words = 4)



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)	$2A$
$a\times b=$	$(a_0.b_0-a_1.b_1, (a_0+a_1).(b_0+b_1)-a_0.b_0-a_1.b_1)$	$3M+5A$
$a^2=$	$((a_0+a_1)(a_0-a_1), 2a_0a_1)$	$2M+3A$
$a^{-1}=$	$(a_0(a_0^2+a_1^2)^{-1}, -a_1(a_0^2+a_1^2)^{-1})$	$4M+2A+1I$

SIKE Architecture

KEY GENERATION (Bob)

Bob's secret key s_B

Legend

Public Parameters
Alice's values
Bob's values

SIKE Architecture

KEY GENERATION (Bob)

Bob's secret key s_B $\longrightarrow$

Isogeny

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

Legend

Public Parameters
Alice's values
Bob's values

SIKE Architecture

KEY GENERATION (Bob)

Isogeny

Bob's secret key s_B → $E_B = E_0/\langle P_B + [s_B]Q_B \rangle$ → Bob's public key $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

Legend

Public Parameters
Alice's values
Bob's values

SIKE Architecture

KEY GENERATION (Bob)

Bob's secret key s_B → Isogeny: $E_B = E_0/\langle P_B + [s_B]Q_B \rangle$ → Bob's public key $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

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)

Isogeny

Bob's secret key s_B → $E_B = E_0/\langle P_B + [s_B]Q_B \rangle$ → Bob's public key $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

KEY ENCAPSULATION (Alice)

Hash

Alice's secret message m → $r = \text{Keccak}(m, pk_B)$

Bob's public key pk_B → $r = \text{Keccak}(m, pk_B)$

Legend

Public Parameters
Alice's values
Bob's values

SIKE Architecture

KEY GENERATION (Bob)

Bob's secret key s_B →

Isogeny

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

→

Bob's public key $pk_B =$
$\{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

Legend

Public Parameters
Alice's values
Bob's values

KEY ENCAPSULATION (Alice)

Alice's secret message m →

Hash

$r =$
$\mathrm{Keccak}(m, pk_B)$

→

Isogeny

$E_A =$
$E_0/\langle P_A + [r]Q_A \rangle$

Bob's public key pk_B →

$E_{AB} =$
$E_B/\langle \phi_B(P_A) + [r]\phi_B(Q_A) \rangle$

Isogeny

SIKE Architecture

KEY GENERATION (Bob)

Bob's secret key s_B →

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

Bob's public key $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

KEY ENCAPSULATION (Alice)

Alice's secret message m →

Hash: $r = \mathrm{Keccak}(m, pk_B)$ →

Isogeny: $E_A = E_0/\langle P_A + [r]Q_A \rangle$ →

Alice's public key $pk_A = \{E_A, \phi_A(P_B), \phi_A(Q_B)\}$

Bob's public key pk_B →

Isogeny: $E_{AB} = E_B/\langle \phi_B(P_A) + [r]\phi_B(Q_A) \rangle$ →

Hash: $c = \mathrm{Keccak}(j(E_{AB})) \oplus m$

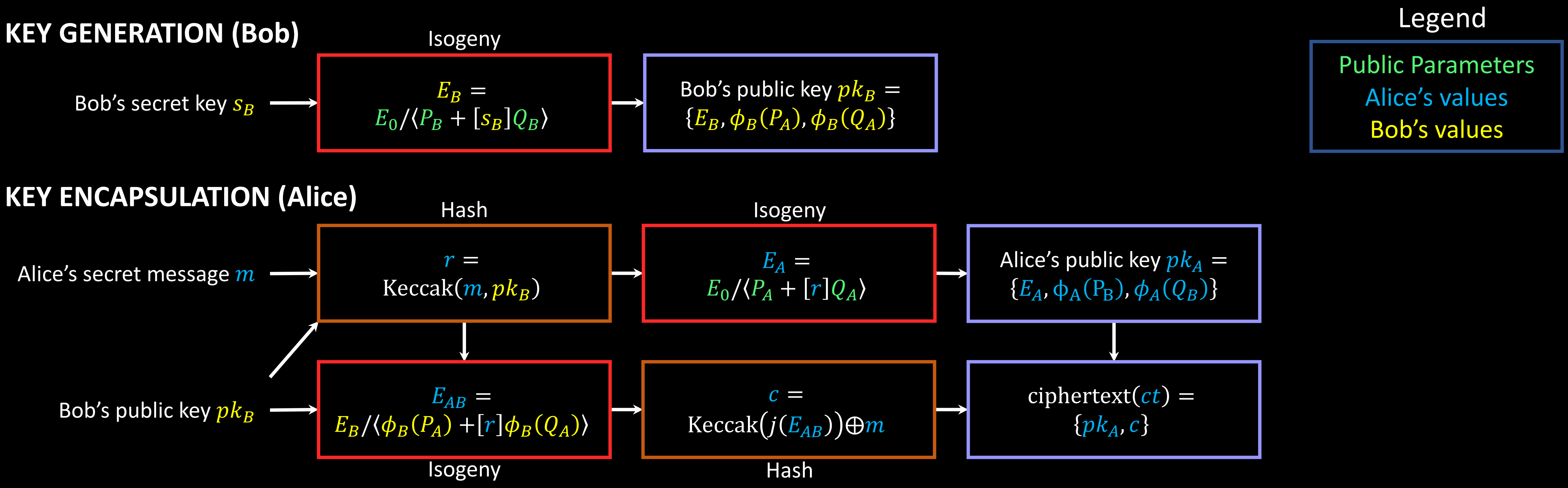
Legend

Public Parameters

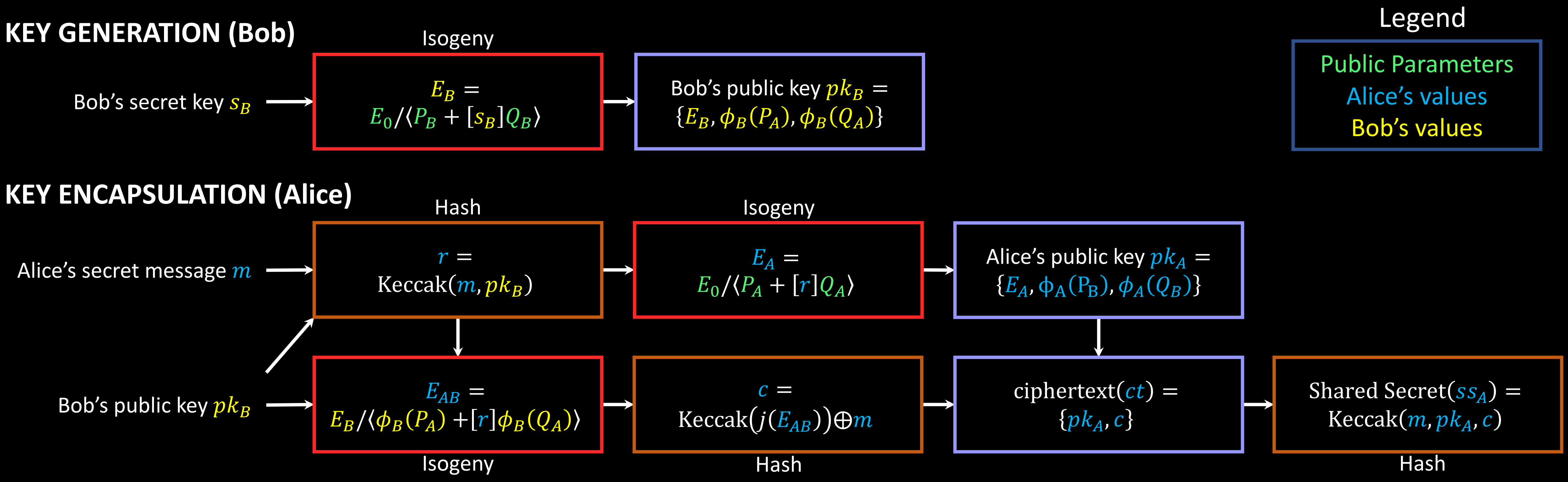
Alice's values

Bob's values

SIKE Architecture



SIKE Architecture



SIKE Architecture

KEY GENERATION (Bob)

Bob's secret key s_B → Isogeny: $E_B = E_0/\langle P_B + [s_B]Q_B \rangle$ → Bob's public key $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

Legend

Public Parameters

Alice's values

Bob's values

KEY ENCAPSULATION (Alice)

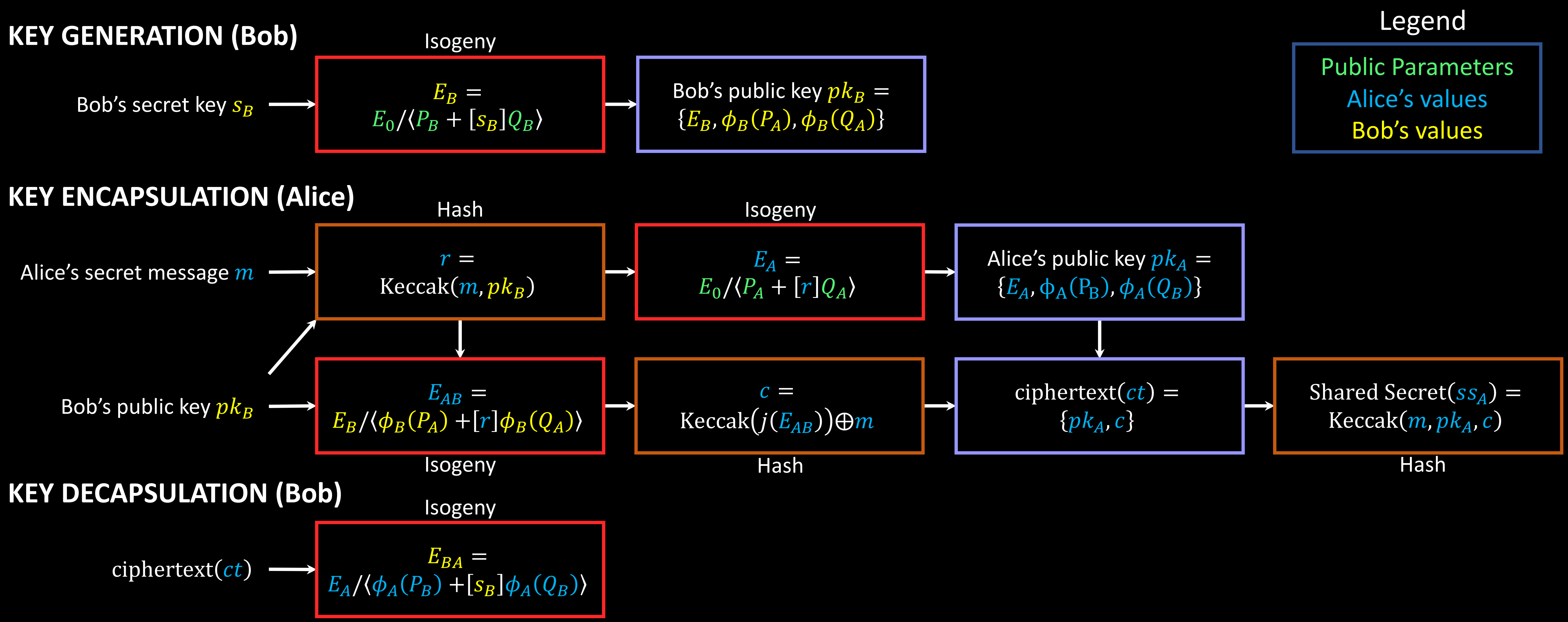
Alice's secret message m → Hash: $r = \text{Keccak}(m, pk_B)$ → Isogeny: $E_A = E_0/\langle P_A + [r]Q_A \rangle$ → Alice's public key $pk_A = \{E_A, \phi_A(P_B), \phi_A(Q_B)\}$

Bob's public key pk_B → Isogeny: $E_{AB} = E_B/\langle \phi_B(P_A) + [r]\phi_B(Q_A) \rangle$ → Hash: $c = \text{Keccak}(j(E_{AB})) \oplus m$ → $\text{ciphertext}(ct) = \{pk_A, c\}$ → Hash: $\text{Shared Secret}(ss_A) = \text{Keccak}(m, pk_A, c)$

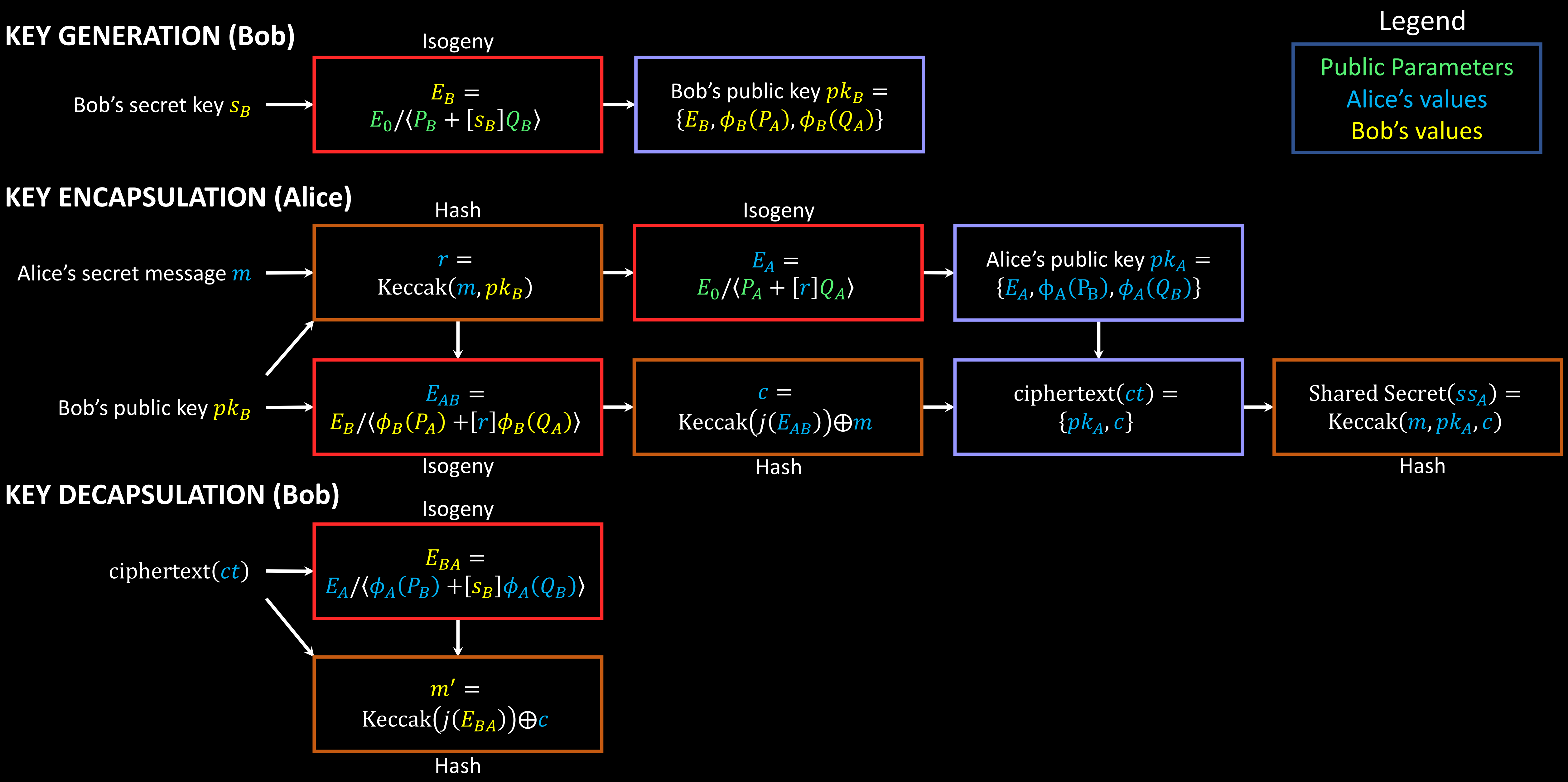
KEY DECAPSULATION (Bob)

$\text{ciphertext}(ct)$

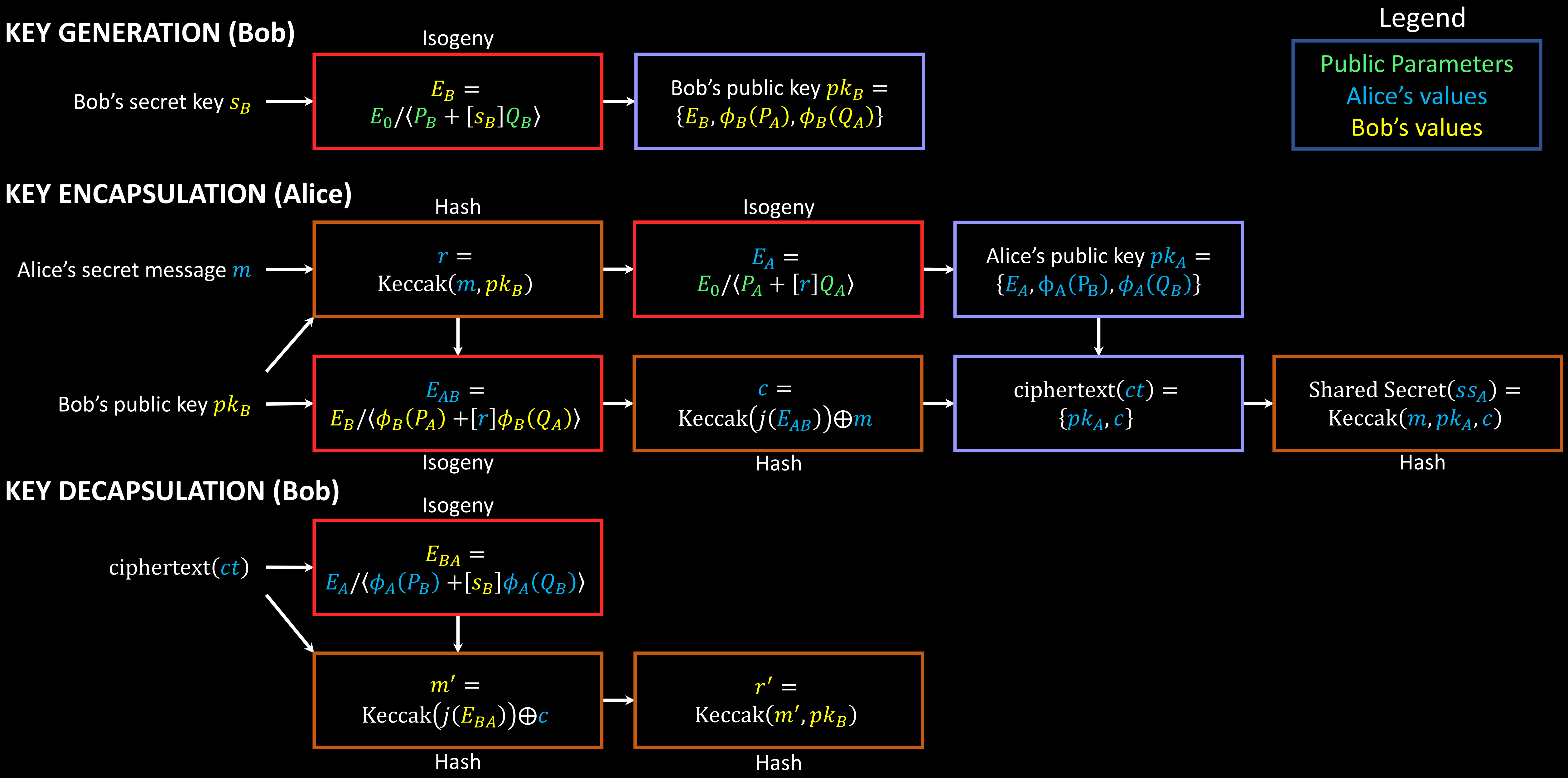
SIKE Architecture



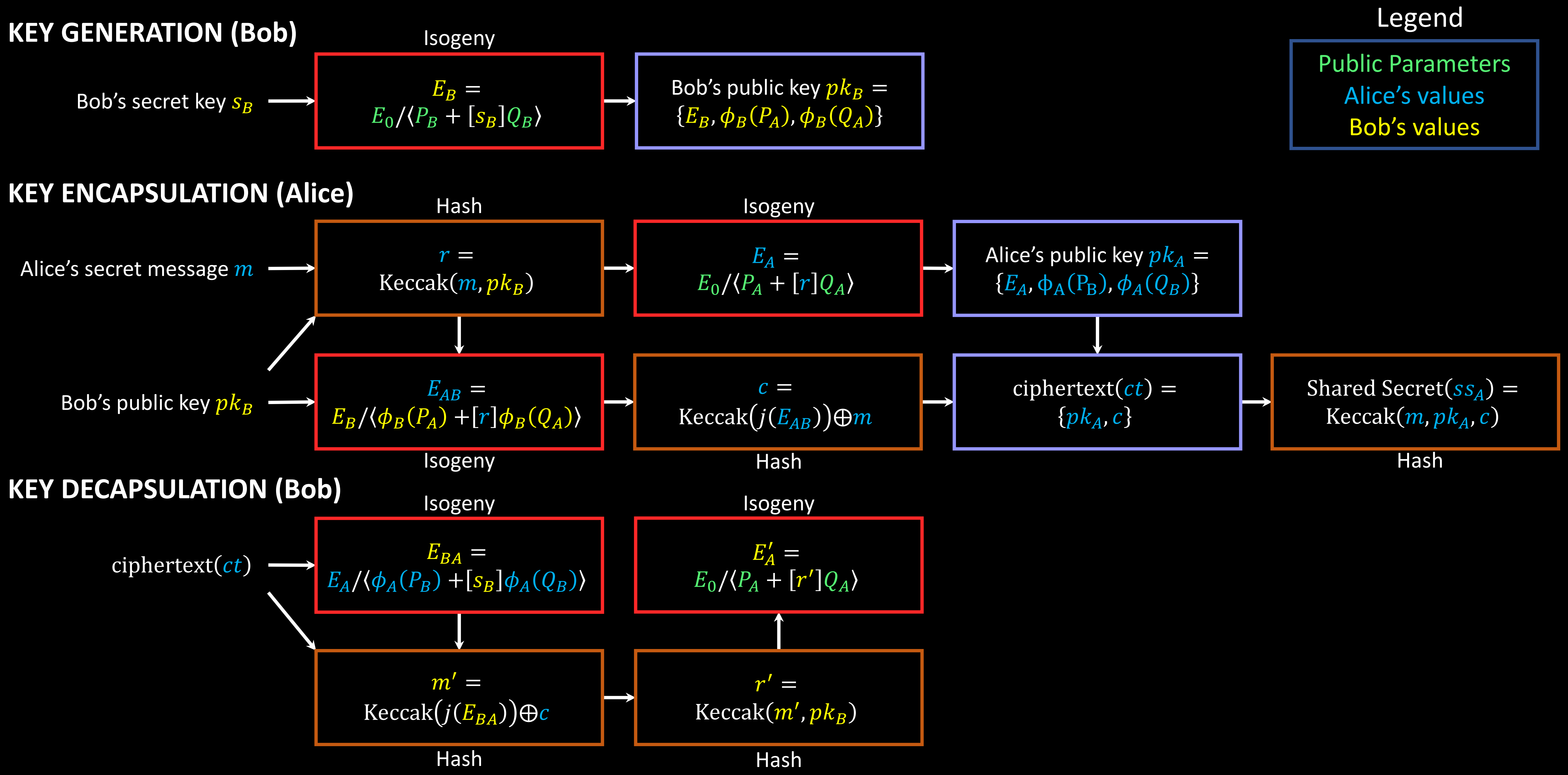
SIKE Architecture



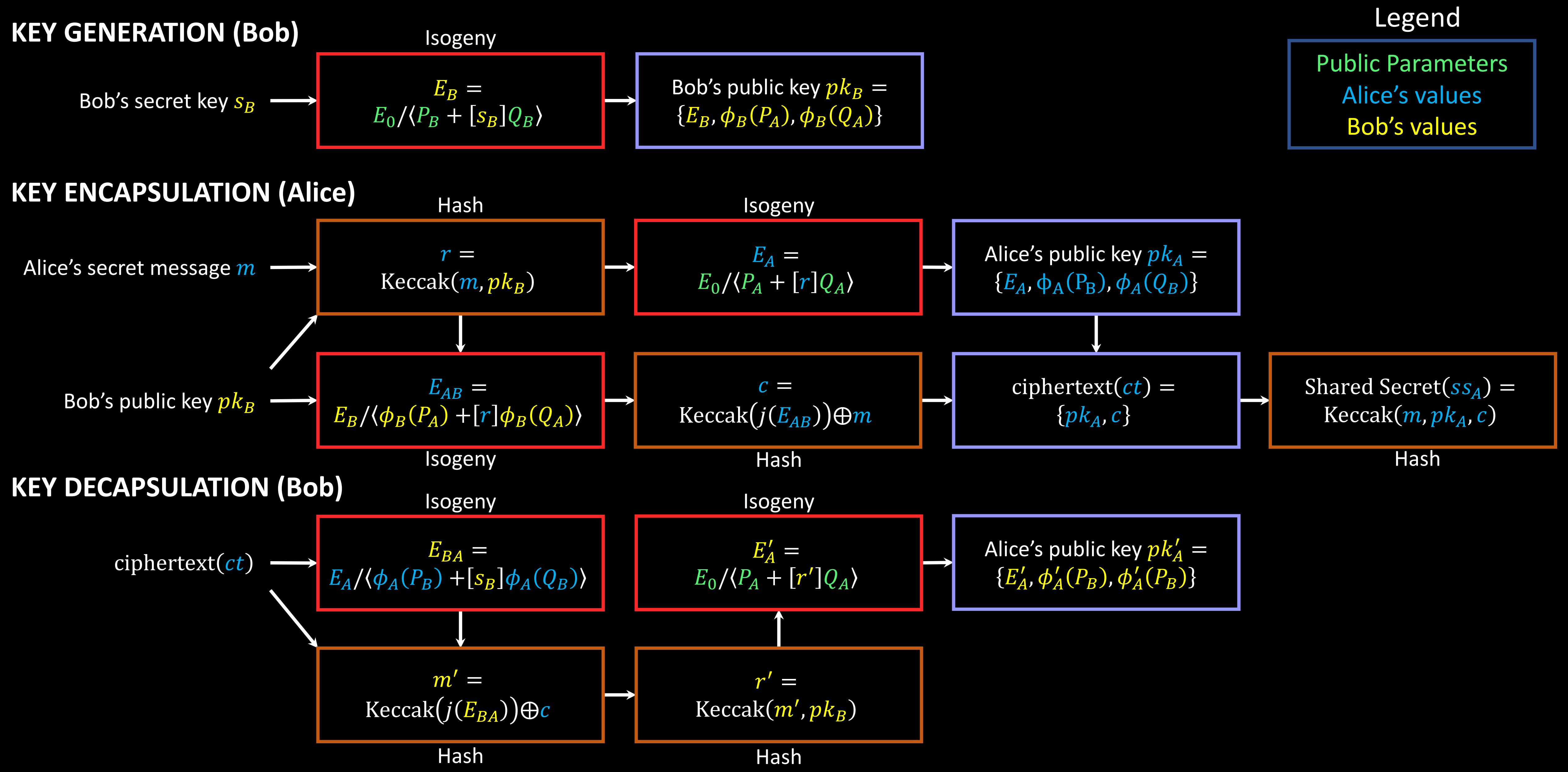
SIKE Architecture



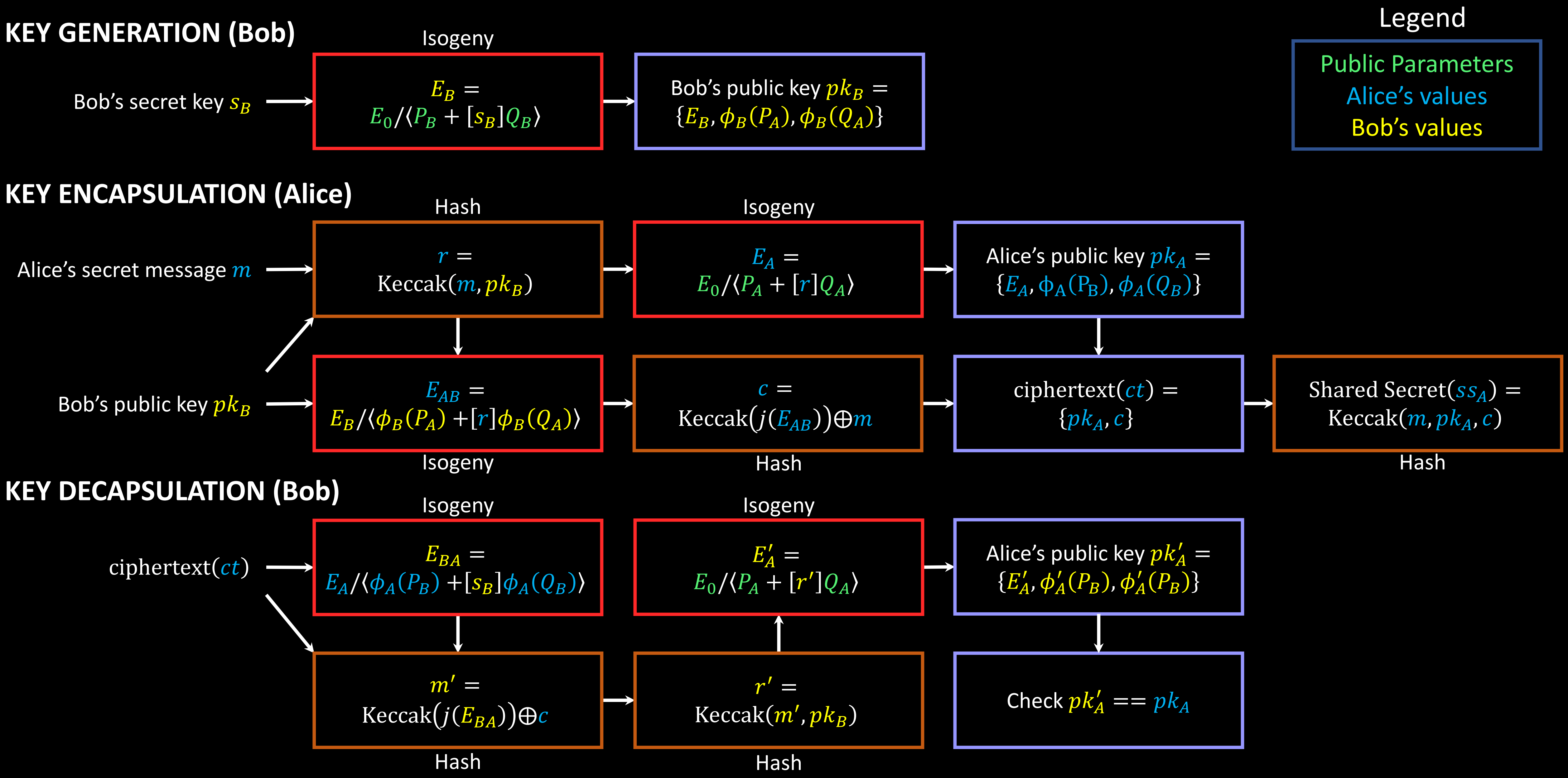
SIKE Architecture



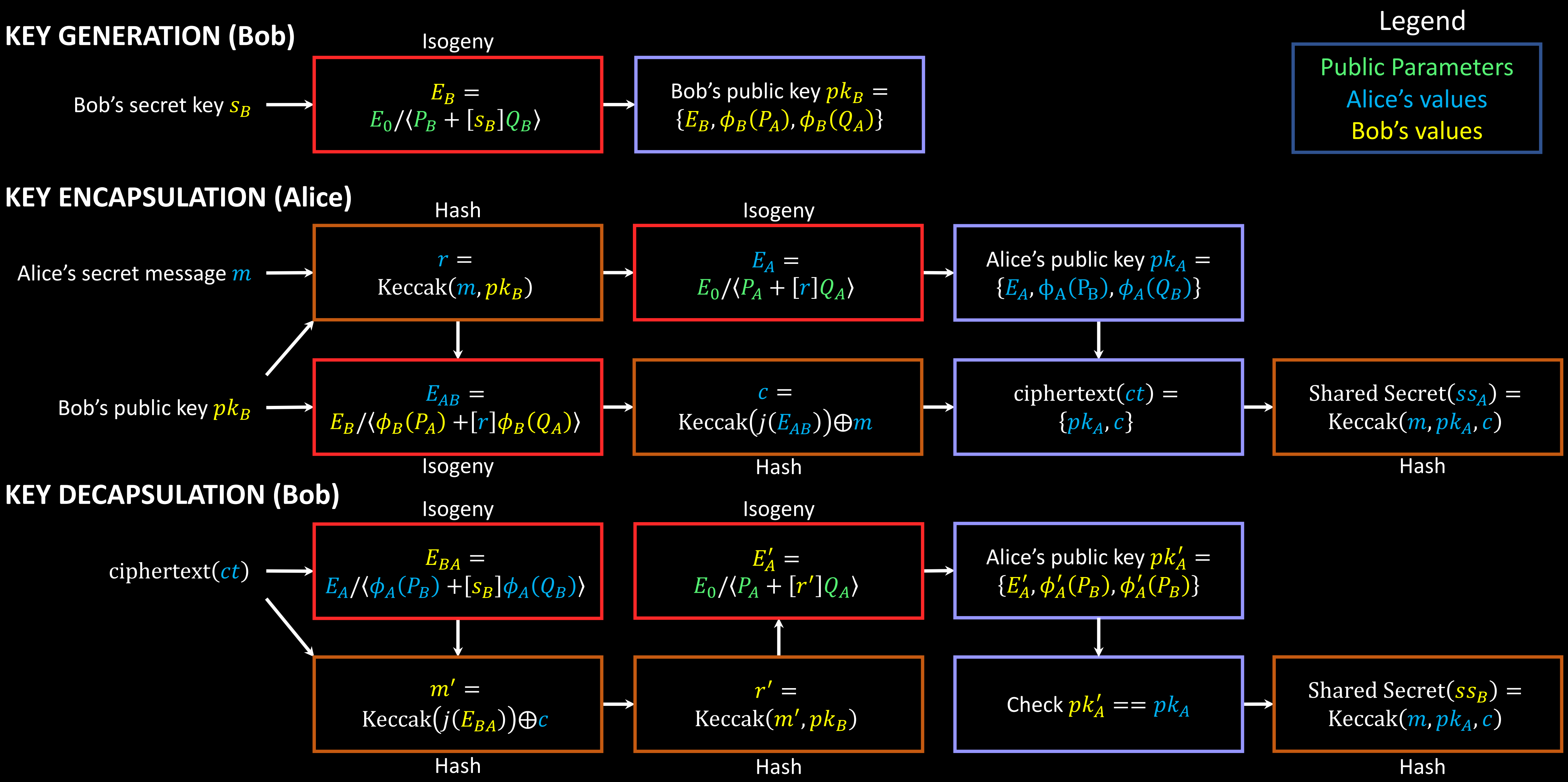
SIKE Architecture



SIKE Architecture

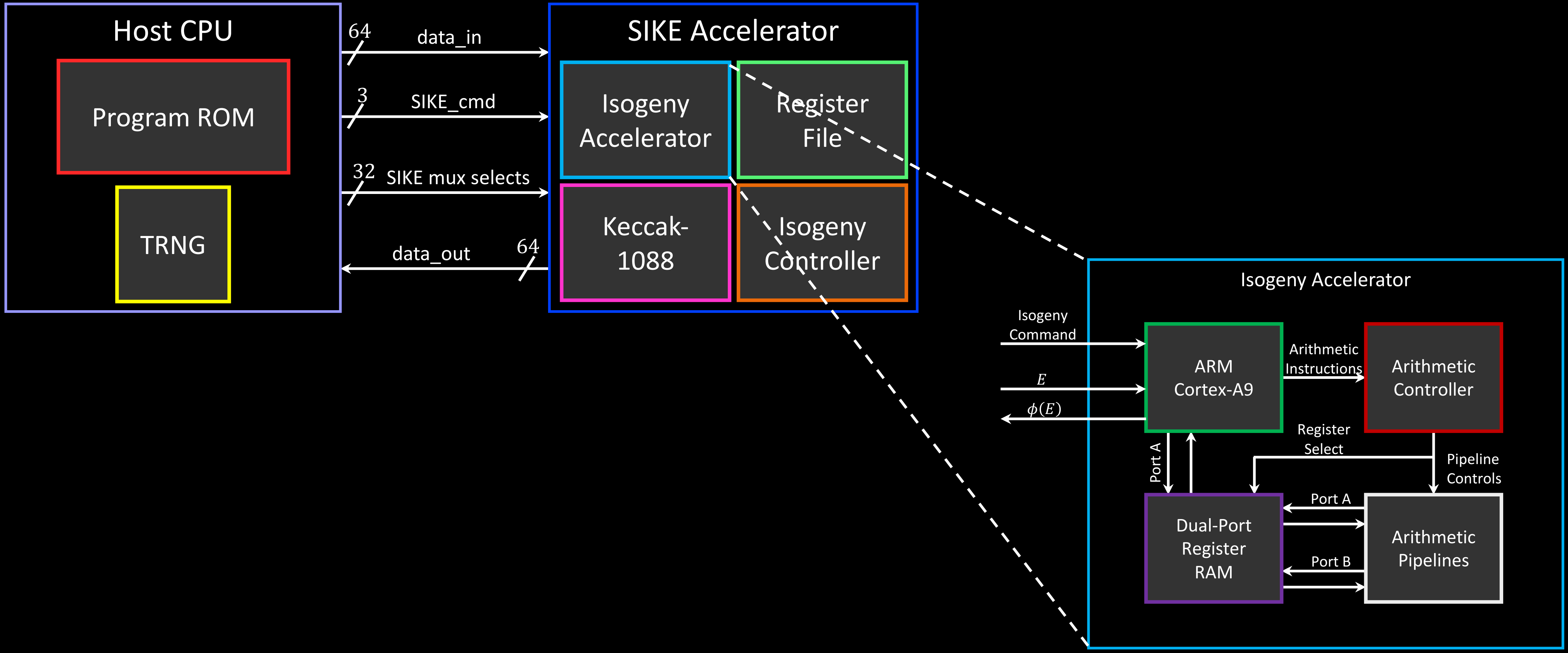


SIKE Architecture



SIKE in FPGA

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



SIKE Round 2 Key sizes

NIST Level	Prime (bits)	Public key size (bytes)	Compressed PK size (bytes)
1	434	330	196
2	503	378	224
3	610	462	273
5	751	564	331

Classical Key Size Comparison (in Bytes)

	Level 1	Level 3	Level 5
RSA	384	960	1920
ECC	32	48	64
SIKE	330	462	564
SIKE Compressed	196	273	331

- RSA and ECC are currently used but **NOT** quantum-safe
- Other Post-Quantum candidates have key lengths up to 10 times longer

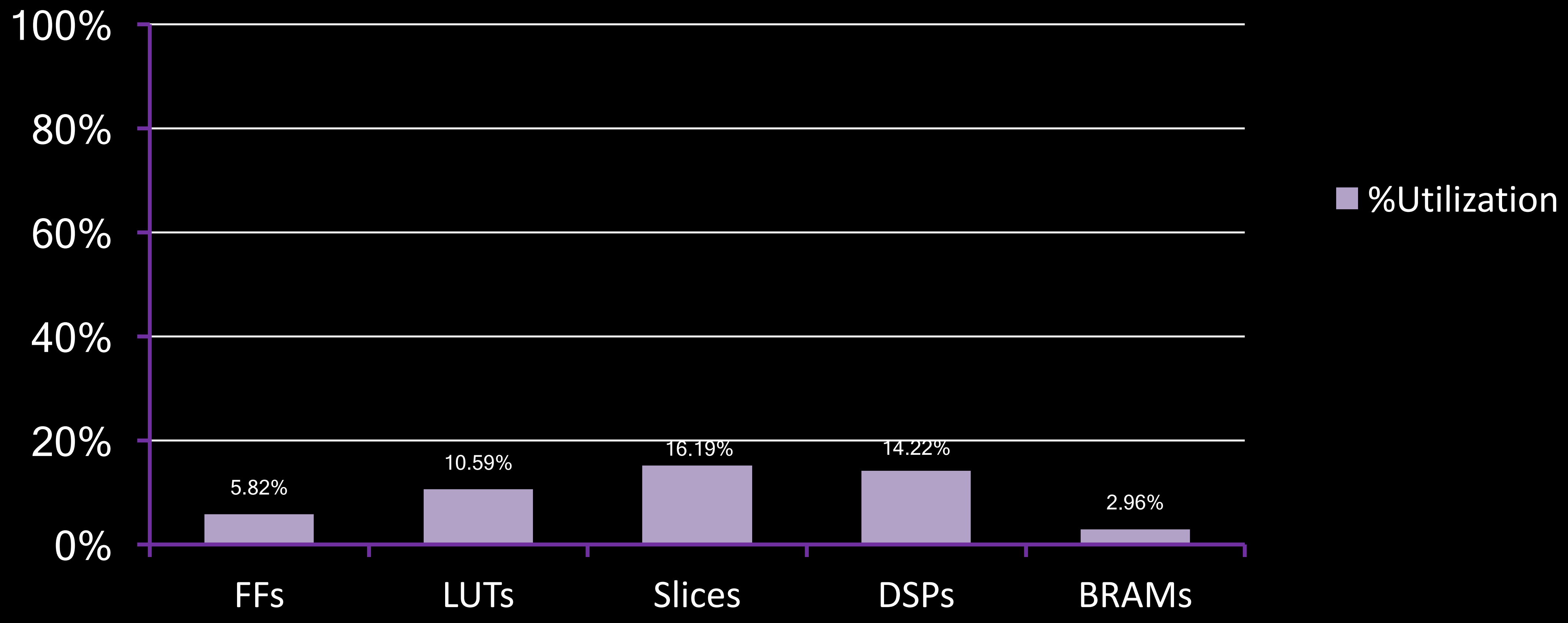
SIKE Operations

- 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

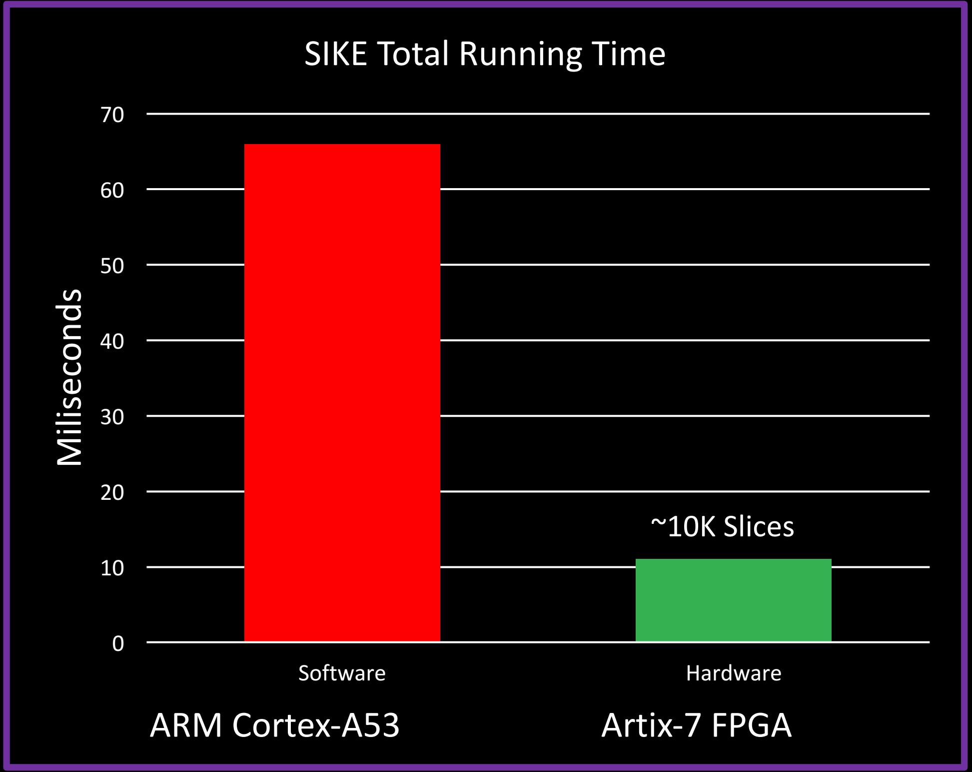
SIKE in FPGA Area Results

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

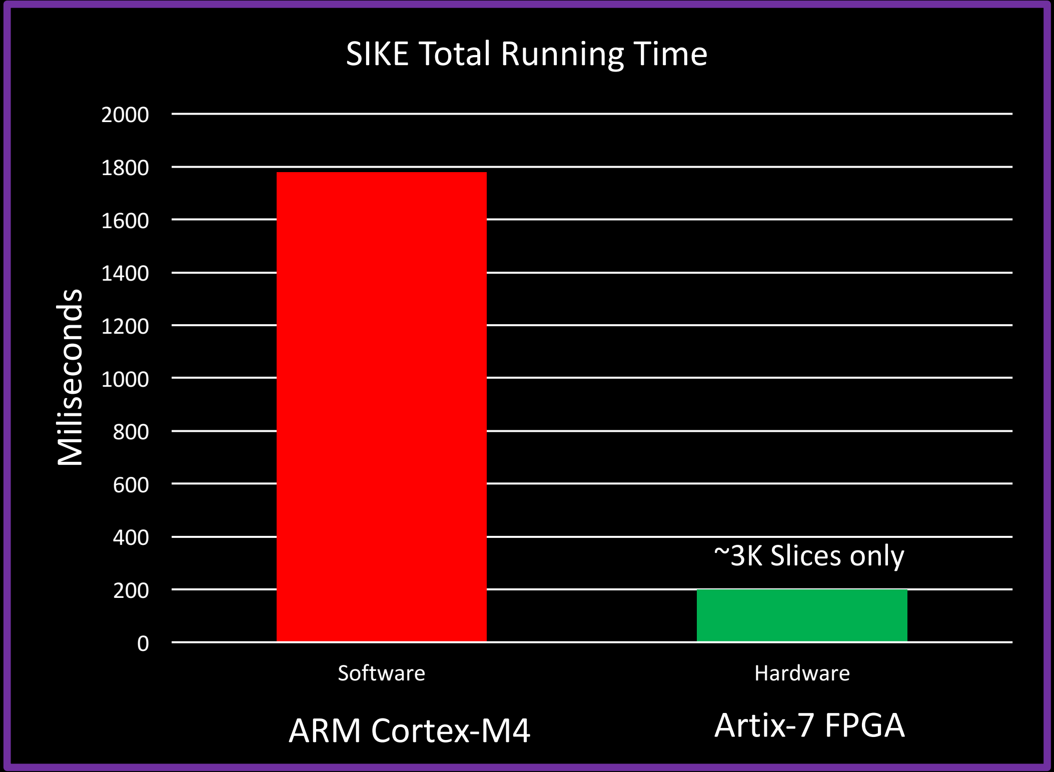


SIKE: Results for NIST level 1

Target: High Performance Edge



Target: Resource-constrained IoT



SIKE in FPGA

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	**33.35**

SIKE in FPGA Improved

CHES 2019: 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**

The case for SIKE

- 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: Communication 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.

- Questions?

Reza Azarderakhsh:

Email:

razarderakhsh@fau.edu