SIKE in Hardware

Reza Azarderakhsh
Florida Atlantic University
CHES 2019

Atlanta, GA, USA

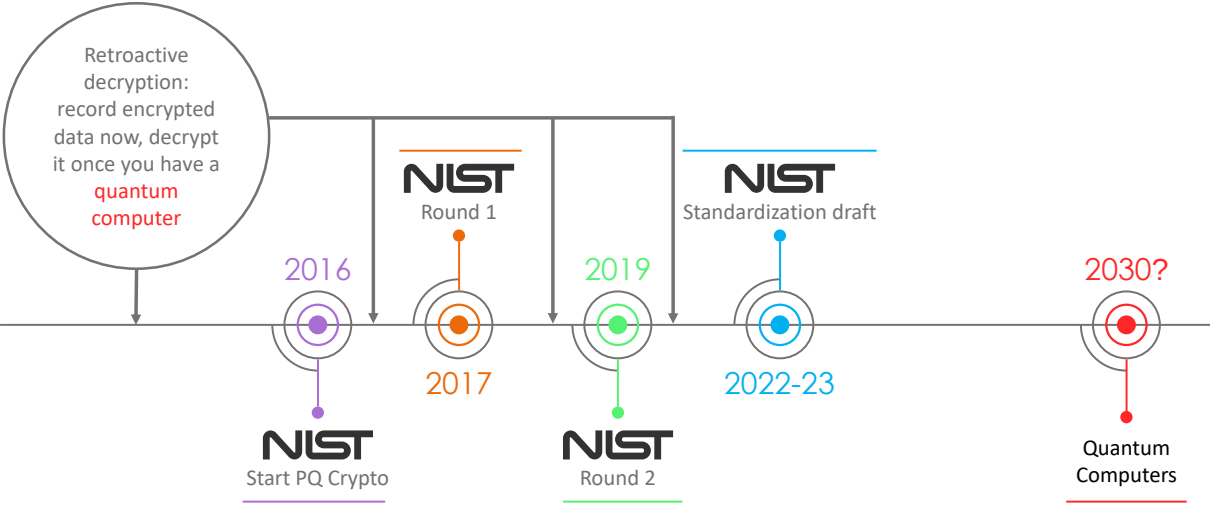
Quantum Threat to Information Security

Large-scale quantum computers could break some encryption schemes

Need to migrate encryption to quantum-resistant algorithms

When we should start the process?

Timeline


Retroactive decryption: record encrypted data now, decrypt it once you have a quantum computer
2016
NIST
Start PQ Crypto
NIST
Round 1
2017
2019
NIST
Round 2
NIST
Standardization draft
2022-23
2030?
Quantum Computers

Post-Quantum Key-Exchange

Lattice-based

Code-based

Isogeny-based

Post-Quantum Signatures

Lattice-based

Hash-based

Multivariate-based

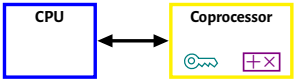
Zero-Knowledge based

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

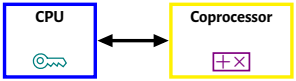
Architecture Selection for Cryptographic Design

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

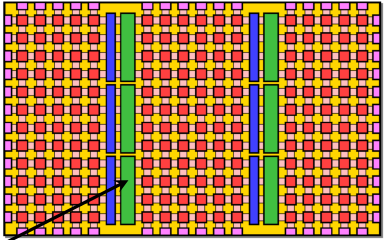
SW only

+ Limited HW costs (code/data storage)
+ Highest flexibility
+ Minimal HW design effort/eases handling of complexity (programming)
− Not optimized (energy, consumption, performance)

FPGAs: Field Programmable Gate Arrays

FPGAs are composed of:

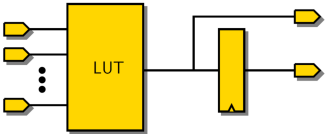
- Programmable logic cells
- A configurable routing matrix
- Configurable input/output cells
- Embedded memory blocks
- Small embedded multipliers
- etc.



18-bit×18-bit multiplier blocks

Inside a logic cell:

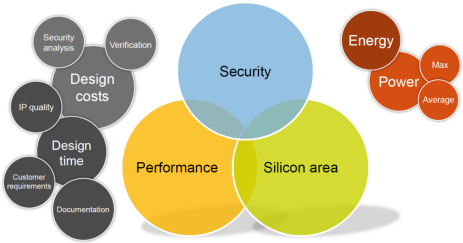
- Connections to the routing matrix
- Programmable lookup-tables
  - 4 inputs, 1 output
  - 6 inputs, 1 output
  - 6 inputs, 2 outputs
- Optional registers
  - Free pipelining
- More logic for fast carry-propagation



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



SIKE Team

UNIVERSITY OF WATERLOO

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

FAU FLORIDA ATLANTIC UNIVERSITY

Radboud Universiteit

UNIVERSITY OF TORONTO

Microsoft Research

amazon

LinkedIn

TEXAS INSTRUMENTS

INFOSEC GLOBAL

PQSecure

evolution

A brief history of public key cryptography

Cryptosystem	Hard Problem
Diffie-Hellman(1976) Elliptic curve cryptography (1986) Pairing-based cryptography (2000)	Discrete logarithms
RSA (1977) Rabin (1978) Composite residues (1985)	Factoring integers
Code-based cryptography (1979)	Decoding linear codes
Lattice-based / NTRU (1996) Isogeny based / CRS (1996) **SIDH / SIKE (2011)**	Finding Short lattice vectors Computing isogenies

History of Supersingular Isogeny-based cryptography

- [2006]: Birth of Supersingular isogeny-based cryptosystem
  - Charles-Goren-Lauter
  - built hash function from supersingular isogeny graph
- [2011]: Supersingular isogeny Diffie-Hellman key exhange (SIDH)
  - Jao-De Feo
- [2017]: Supersingular isogeny key encapsulation (SIKE)
  - SIKE Team (https://sike.org)

SIDH/SIKE

Supersingular Isogeny Diffie-Hellman (Jao and De Feo, 2011):

- A key-exchange protocol, similar to Diffie-Hellman, using isogenies between supersingular elliptic curves

Why isogenies?

- Because they seem to be quantum-resistant

Why supersingular elliptic curves?

- There is a quantum subexponential attack for ordinary (i.e. non-supersingular) curves (Childs, Jao, and Soukharev 2014)

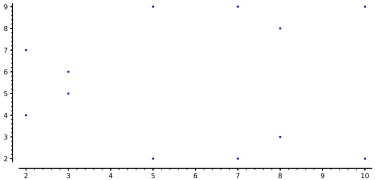
Supersingular Isogeny Key Encapsulation

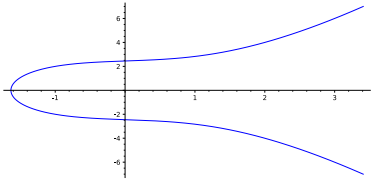
- A more secure (and slower) version of SIDH, with random padding and protection against active attacks.

(Ordinary) Elliptic Curves

- An elliptic curve over prime field: $E_W/\mathbb{F}_p: \ y^2 = x^3 + ax + b$.
- To avoid infinite sets, we choose $x, y, a, b \in \mathbb{F}_p$.
- Example: SAGE: `E=EllipticCurve(GF(11),[1,6])`
  - Elliptic Curve defined by $y^2 = x^3 + x + 6$ over Finite Field of size 11
- SAGE: `E.points()`
- [(0 : 1 : 0), (2 : 4 : 1), (2 : 7 : 1), (3 : 5 : 1), (3 : 6 : 1), (5 : 2 : 1), (5 : 9 : 1), (7 : 2 : 1), (7 : 9 : 1), (8 : 3 : 1), (8 : 8 : 1), (10 : 2 : 1), (10 : 9 : 1)]

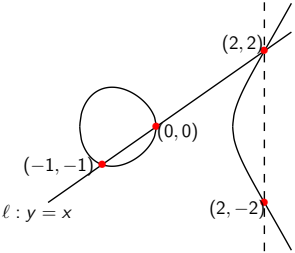
- SAGE: `E.cardinality()` or `E.order()=13`
- `E.is_ordinary()`: True
- `E.is_supersingular()`: False





Point Addition

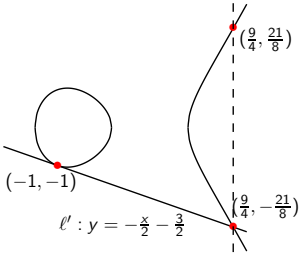
- Let E be an elliptic curve.
- Suppose $P, Q \in E$.
- We want to add P and Q.
- Draw a line through P and Q.
- Find where this line crosses E.
- Reflect around the x-axis.
- The reflected point is $P + Q$.



$E/\mathbb{R} : y^2 = x^3 - 2x$: addition.

Point Doubling

- Let E be an elliptic curve.
- Suppose $P, Q \in E$.
- To compute $P + Q$ when $P = Q$:
- Draw the tangent line through P.
- Find where this line crosses E.
- Reflect around the x-axis.
- The reflected point is $P + P$.



$E/\mathbb{R} : y^2 = x^3 - 2$: doubling.

Montgomery Curves

- 1987 Montgomery: All Montgomery curves are elliptic curves.
- Not all elliptic curves can be written in Montgomery form.

$$by^2 = x^3 + ax^2 + x$$

- It has been observed that the x-coordinate of $R = P + Q$ depends only on the x-coordinates of P, Q, and $P - Q$.

$$(x_3 + y_3) - (x_2, y_2) = (x_1, y_1)$$
$$(x_3 + y_3) + (x_2, y_2) = (x_5, y_5)$$
$$\Rightarrow x_5 = \frac{(x_2 x_3 - 1)^2}{x_1 (x_2 - x_3)^2}$$

- Similarly when $P = Q$ it is true for doubling: x-only doubling.

$$2(x_2 + y_2) = (x_4, y_4) \Rightarrow x_4 = \frac{(x_2^2 - 1)^2}{4x_2(x_2^2 + ax_2 + 1)}$$

- Can compute $P + Q$ from $\{P, Q, P - Q\}$ without y-coordinates.
- Use projective coordinates: points $(X : Z)$ with $x = X/Z$
- Cheap differential addition $4M + 2S$ and doubling $(2M + 2S)$

Real-world usage: Curve25519

- Proposed by Dan Bernestien 2006:

$$E_M/F_p : y^2 = x^3 + 486662x^2 + x$$

- Over prime number $p = 2^{255} - 19$
- Used for DH key exchange

Montgomery Ladder

- Point multiplication: $Q = k \cdot P$

```
function scalar-mult(k, P):
    T_0 ← O
    T_1 ← P
    for i ← n − 1 downto 0:
        if k_i = 1:
            T_0 ← T_0 + T_1
            T_1 ← 2T_1
        else:
            T_1 ← T_0 + T_1
            T_0 ← 2T_0
    return T_0
```

- Properties:
  - perform one addition and one doubling at each step
  - ensure that both results are used in the next step
- Example: $k = 26 = (11010)_2$

Supersingular Elliptic Curves

- Let $E/\mathbb{F}_q$ be an elliptic curve with $q = p^n$
- E is supersingular if $p \mid (q + 1 - \#E(\mathbb{F}_q))$. Otherwise, it is ordinary.
- Special cases:
  - When $E/\mathbb{F}_p$ supersingular and $\#E(\mathbb{F}_p) = P + 1$
  - When $E/\mathbb{F}_{p^2}$ supersingular and $\#E(\mathbb{F}_{p^2}) = (P + 1)^2$

- Example: EllipticCurve(GF(11),[1,0]): $y^2 = x^3 + x$
- `E.is_supersingular()`
  - True, `E.order()=12`
- EllipticCurve(GF(11^2),[1,0]): $y^2 = x^3 + x$ is also supersingular.
  - `E.order()=144`

Supersingular Elliptic Curves

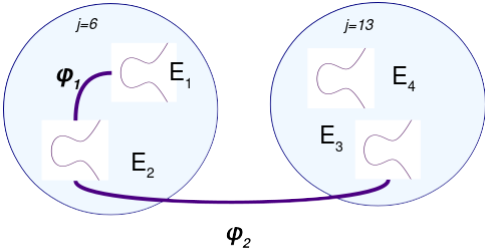
- There are only a finite number of supersingular elliptic curves.
- All supersingular curves can be defined over $\mathbb{F}_{p^2}$
- Over an algebraically closed field an elliptic curve is determined by its j-invariant.
  - it can be viewed as a way to group multiple elliptic curves into disjoint sets.
- Example: EllipticCurve(GF(11),[1,0]):$y^2 = x^3 + x$
  - $j(E) = j(a, b) = 1728\frac{4a^3}{4a^3+27b^2}$
  - `E.j_invariant=1`
- The j-invariant determines isomorphisim class over the field.
- $E1/\mathbb{F}_{13} : y^2 = x^3 + 9x + 8$, `E1=EllipticCurve(GF(13),[9,8])`
- $E2/\mathbb{F}_{13} : y^2 = x^3 + 3x + 5$, `E2=EllipticCurve(GF(13),[3,5])`
  - `E1.j_invariant()=E2.j_invariant()=3`
  - `E1.is_isomorphic(E2)`: True

Isomorphisms and Isogenies

- E_1 and E_2 isomorphic iff $j(E_1) = j(E_2)$.
- E_1 and E_2 isogenous iff $\#E_1 = \#E_2$.
- $\#E(\mathbb{F}_q) \leq q + 1 \pm 2\sqrt{q}$ (Hasse theorem)

So

- isogeny classes: $O(\sqrt{q})$
- isomorphism classes: $O(q)$



Isogenies of elliptic curves

Definition

An isogeny of elliptic curves over k is a non-zero morphism $E \to E'$ with finite kernel.

Definition

Let $E, E'/\mathbb{F}_q$ be elliptic curves and let $\ell \in \mathbb{Z}_{>0}$ be coprime to q.
An ℓ-isogeny $f : E \to E'$ is an isogeny with $\#\ker(f) = \ell$.

Fact

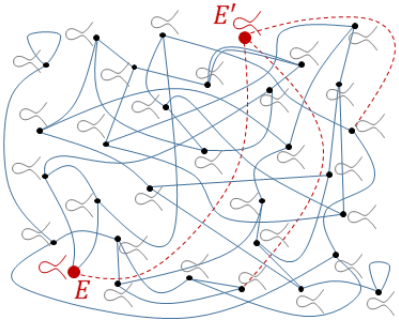
An isogeny is uniquely determined by its kernel.

- *Write $\phi_G : E \to E/G$ for the isogeny from E with kernel G.*
- *Vélu's fomulas [1971] compute the ℓ-isogney from its kernel in time $O(\ell)$.*

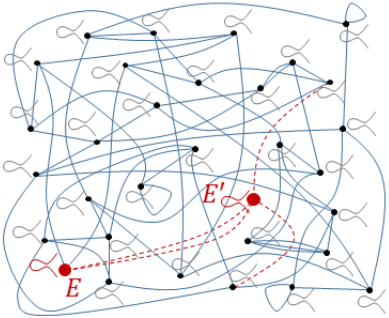
Isogenies of elliptic curves

- We call an isogeny cyclic if its kernel is cyclic.
- The kernel of a cyclic ℓ-isogeny is generated by an ℓ-torsion point.
  - An ℓ-torsion point is a point $P \in E(k)$ such that $[\ell]P = P_\infty$.
- We will work on isogenies with big kernels.

Isogeny graph: Wouter Castryck



2-isogenies



3-isogenies

SIDH overview

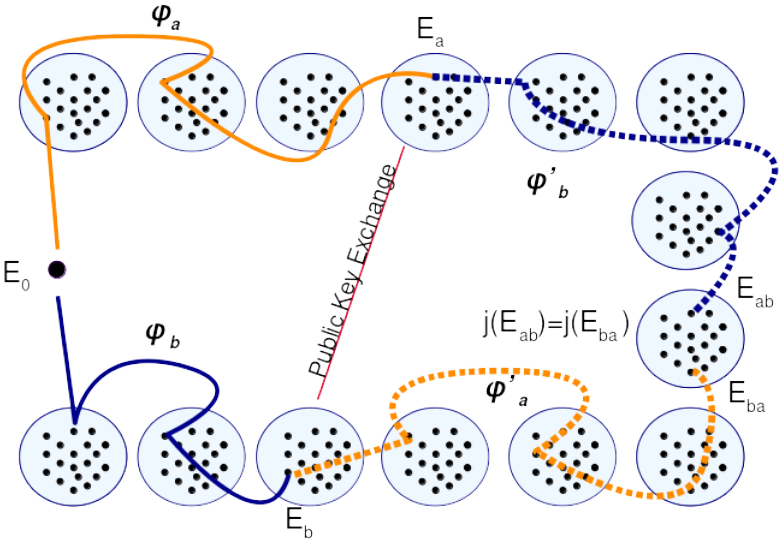
1. Public parameters: Supersingular elliptic curve E over $\mathbb{F}_{p^2}$.
2. Alice chooses a kernel $A \subset E(\mathbb{F}_{p^2})$ and sends E/A to Bob.
3. Bob chooses a kernel $B \subset E(\mathbb{F}_{p^2}$ and sends E/B to Alice.
4. The shared secret is

$$E/\langle A, B\rangle = (E/A)/\phi_A(B) = (E/B)/\phi_B(A).$$

$$\begin{tikzcd}
E \arrow[r, "\phi_A"] \arrow[d, "\phi_B"'] & E/A \arrow[d] \\
E/B \arrow[r] & E/\langle A, B\rangle
\end{tikzcd}$$

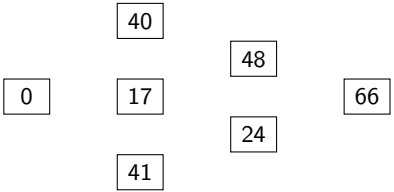
The core operation in SIDH is to compute $\phi_A : E \to E/A$ given A.

SIDH: from Cloudflare



SIDH Overview

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



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

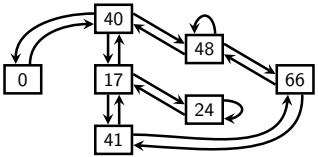
SIDH Parameter Selection

- Finite extension field formed as $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$ with $i^2 + 1 = 0$ (Efficiency)
- Starting curve selection:
  - Montgomery curve: $E_0/\mathbb{F}_{p^2}: \; y^2 = x^3 + x$ (where $j(E_0) = 1728$)
  - SIKE round 2: $E_0/\mathbb{F}_{p^2}: \; y^2 = x^3 + 6x^2 + x$ (where $j(E_0) = 287496$) (Security)
  - Torsion points in $E_0[2^{e_A}]$ and $E_0[3^{e_B}]$

Isogeny Graphs

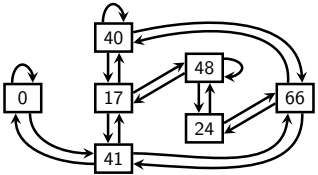
Vertices: All isogenous eliptic curves over $\mathbb{F}_{p^2}$.
Edges: Isogenies of degree ℓ
With Isogeny f degree ℓ, we get a connected $(\ell+1)$-regular graph.

Alice



2-isogeny graph

Bob

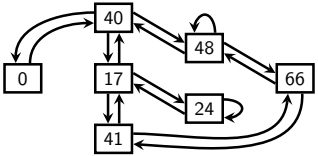


3-isogeny graph

Public Parameters

$E_0/\mathbb{F}_{p^2}$
$\{P_A, Q_A\} \in E_0[2^{e_A}]$
$\{P_B, Q_B\} \in E_0[3^{e_B}]$

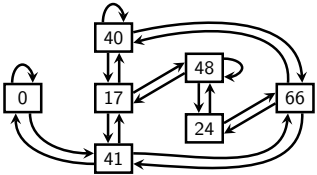
Alice



$P_A = (53, 55)$
$Q_A = (18, 27w + 44)$

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

Bob

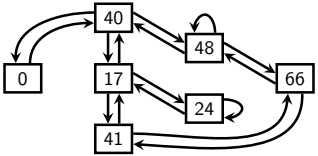


$P_A = (7w + 20, 31w + 50)$
$Q_A = (21w + 64, 38w + 13)$

Secret Key

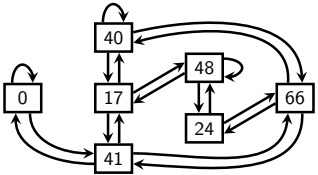
$s_A \in [0, 2^{e_A}]$
$s_B \in [0, 3^{e_B}]$

Alice



$s_A = 6$

Bob

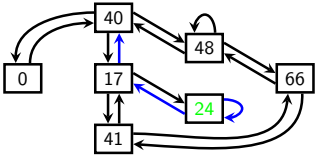


$s_B = 3$

Public Key Generation

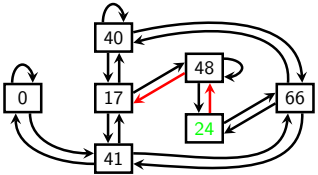
E_A

Alice



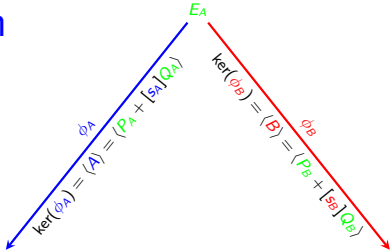
$E_0 : y^2 = x^3 + x$

Bob

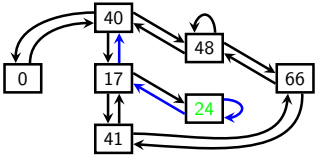


$E_0 : y^2 = x^3 + x$

Public Key Generation



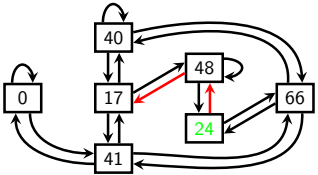
Alice



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

$\phi_A : E_0 \to E_A$

Bob



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

$\phi_B : E_0 \to E_B$

Public Key Generation

E_A

ϕ_A, $\ker(\phi_A) = \langle A \rangle = \langle P_A + [s_A]Q_A \rangle$

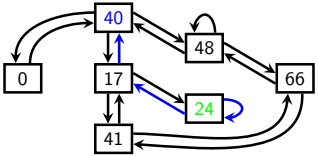
ϕ_B, $\ker(\phi_B) = \langle B \rangle = \langle P_B + [s_B]Q_B \rangle$

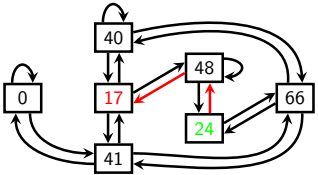
Alice

$$E_A = E_0/\langle A \rangle$$
$$\{R_A, S_A\} = \{\phi_A(P_B), \phi_A(Q_B)\}$$

Bob

$$E_B = E_0/\langle B \rangle$$
$$\{R_B, S_B\} = \{\phi_B(P_A), \phi_B(Q_A)\}$$





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

$\phi_A : E_0 \to E_A$

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

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

$\phi_B : E_0 \to E_B$

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

Key exchange

E_A

ϕ_A

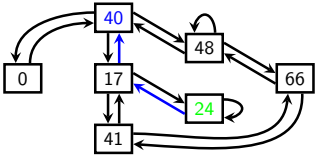
$\ker(\phi_A) = \langle A \rangle = \langle P_A + [s_A]Q_A \rangle$

$\ker(\phi_B) = \langle B \rangle = \langle P_B + [s_B]Q_B \rangle$

ϕ_B

Alice

$$E_A = E_0/\langle A \rangle$$
$$\{R_A, S_A\} = \{\phi_A(P_B), \phi_A(Q_B)\}$$



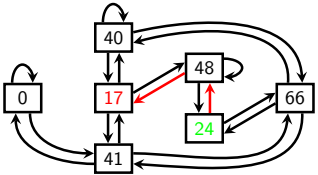
$E_0 : y^2 = x^3 + x$

$\phi_A : E_0 \to E_A$

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

Bob

$$E_B = E_0/\langle B \rangle$$
$$\{R_B, S_B\} = \{\phi_B(P_A), \phi_B(Q_A)\}$$



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

$\phi_B : E_0 \to E_B$

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

Shared Secret Generation

E_A

ϕ_A

$\ker(\phi_A) = \langle A \rangle = \langle P_A + [s_A]Q_A \rangle$

$\ker(\phi_B) = \langle B \rangle = \langle P_B + [s_B]Q_B \rangle$

ϕ_B

Alice

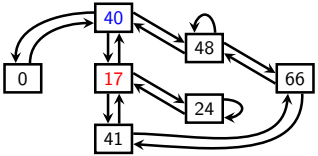
$E_A = E_0/\langle A \rangle$

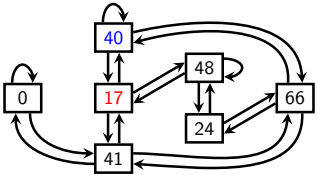
$\{R_A, S_A\} = \{\phi_A(P_B), \phi_A(Q_B)\}$

$E_B = E_0/\langle B \rangle$

$\{R_B, S_B\} = \{\phi_B(P_A), \phi_B(Q_A)\}$

Bob

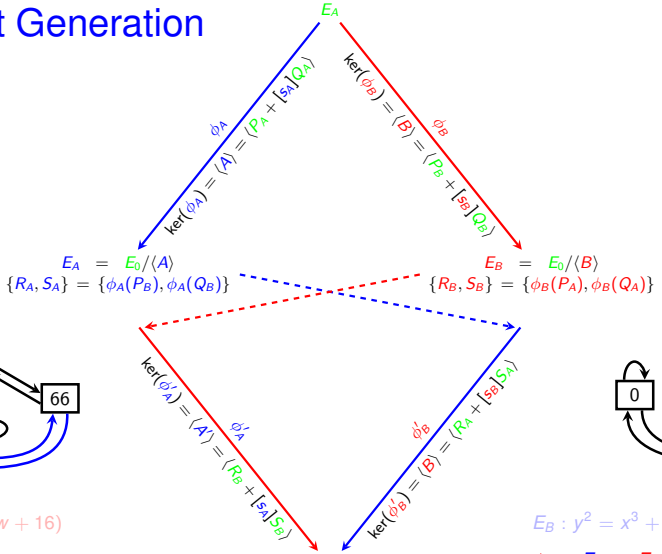




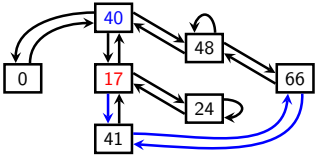
$E_B : y^2 = x^3 + 63x + (55w + 16)$

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

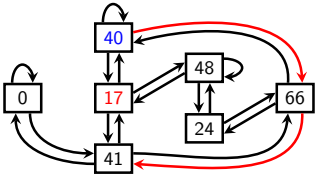
Shared Secret Generation



Alice



Bob



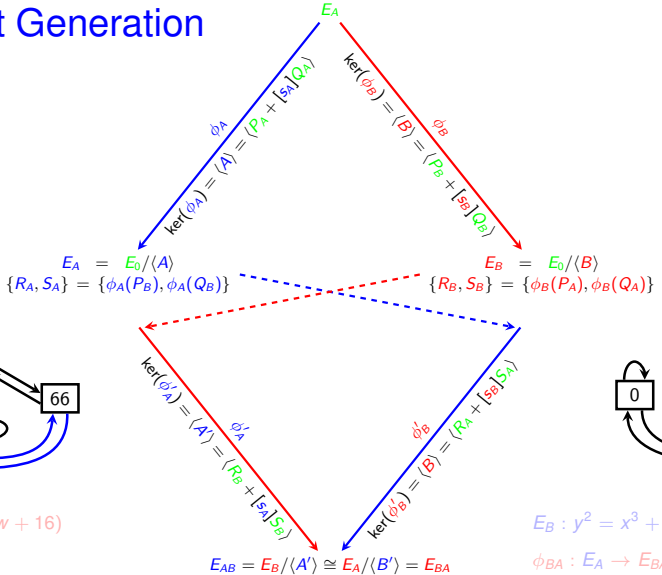
$E_B : y^2 = x^3 + 63x + (55w + 16)$

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

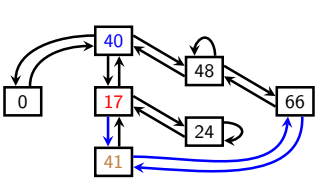
$E_B : y^2 = x^3 + 22x + 35$

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

Shared Secret Generation



Alice

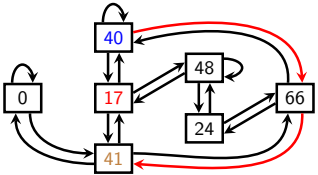


$E_B : y^2 = x^3 + 63x + (55w + 16)$

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

$E_B : y^2 = x^3 + (21w + 14)x + (57w + 21)$

Bob

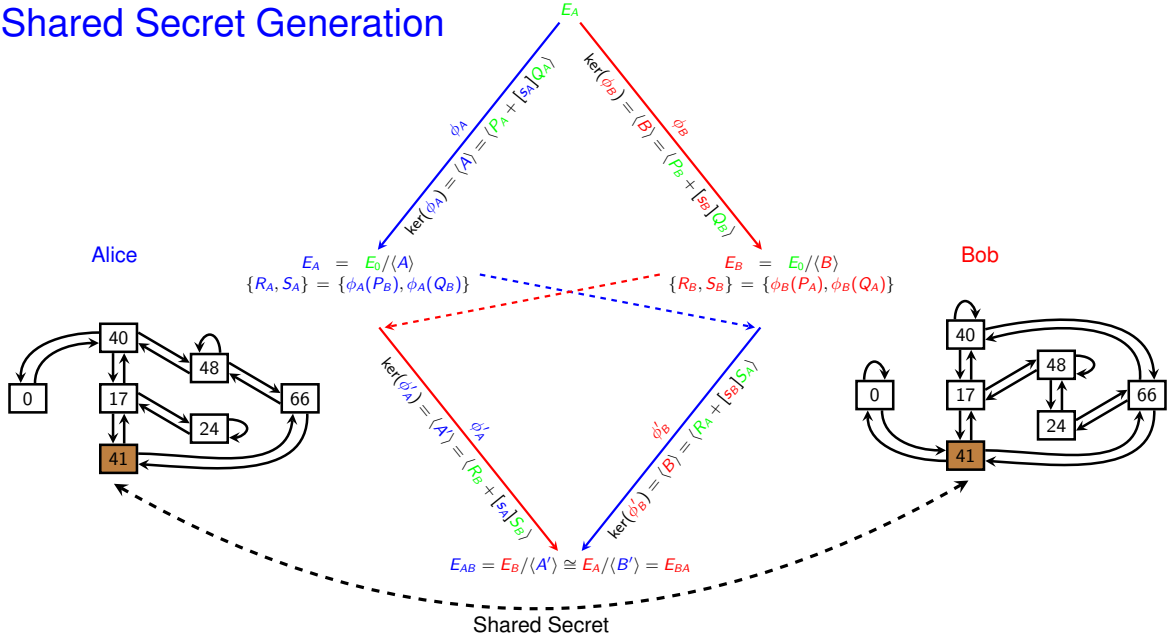


$E_B : y^2 = x^3 + 22x + 35$

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

$E_B : y^2 = x^3 + (21w + 14)x + (57w + 21)$

Shared Secret Generation



SIDH: Security

- Hard problem: Given $P, Q \in E$ and $\phi(P), \phi(Q) \in \phi(E) \Rightarrow$ compute ϕ.
- Best known attack: classical $O(p^{1/4})$ and quantum $O(p^{1/6})$.

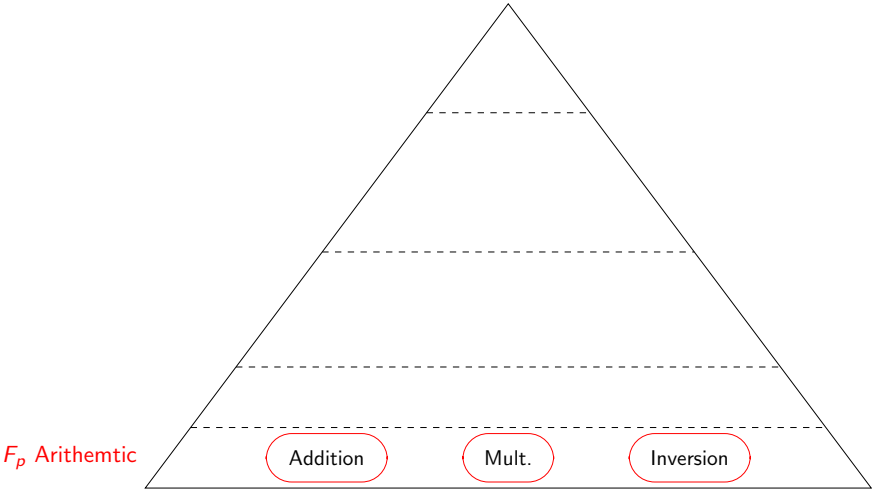
SIKE Round 2 Key sizes

NIST Level	Prime size (bits) Round 1	Prime size (bits) Round 2
1 (AES128)	503	434
2 (SHA256)	—	503
3 (AES192)	751	610
5 (AES256)	964	751

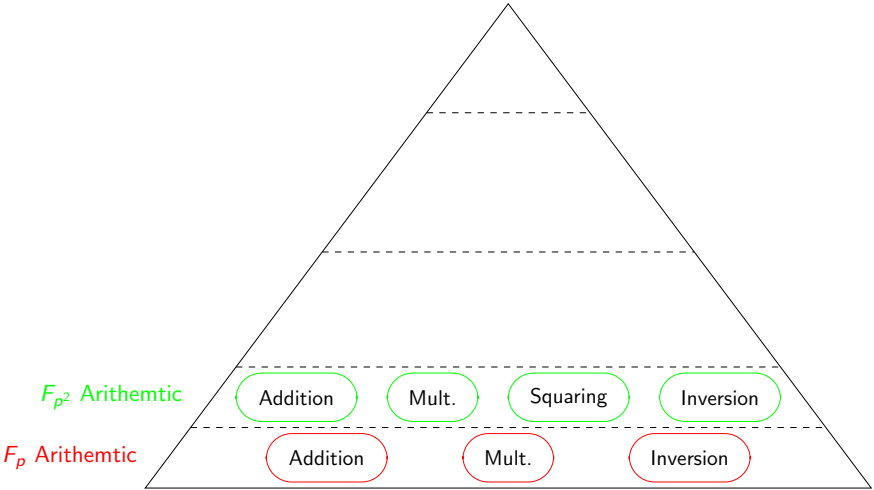
SIKE Round 2 Key sizes

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

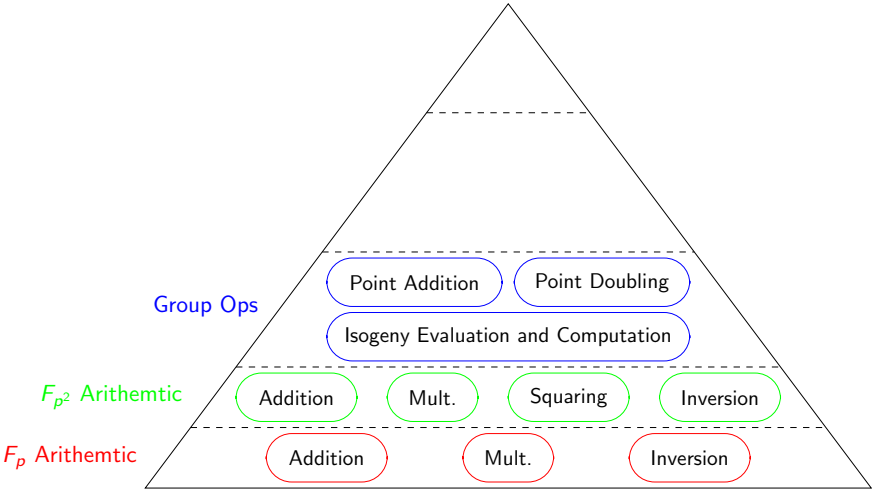
SIDH Computations



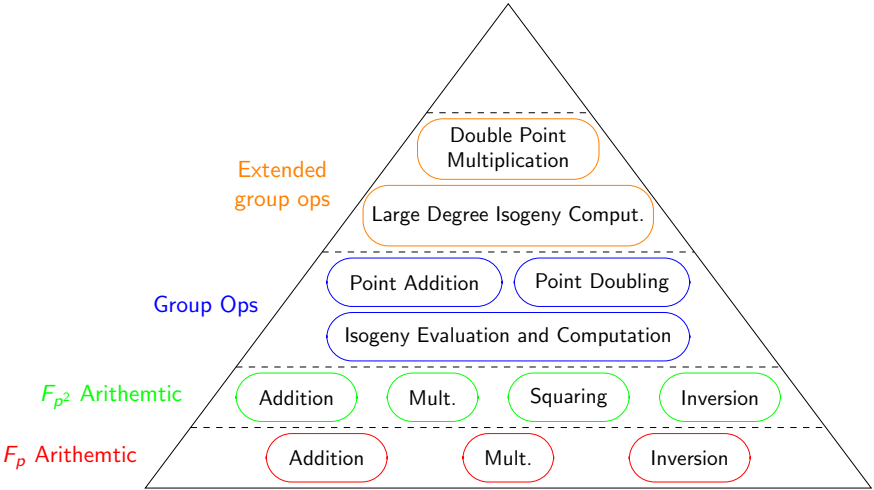
SIDH Computations



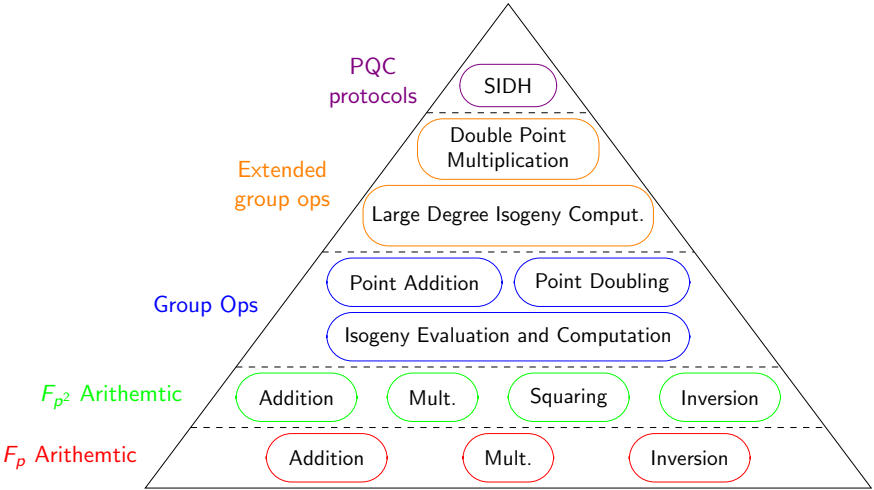
SIDH Computations



SIDH Computations

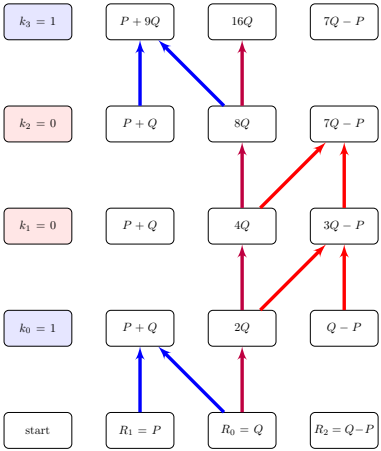


SIDH Computations



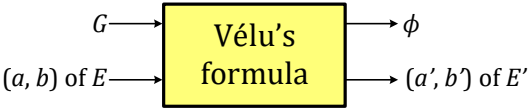
Three-Point Differential Ladder

- Compute $R = P + [k]Q$
- Jao et al. (2014) - Cost: 2PA + 1PD
- Hernandez et al. (2017) - Cost: 1PA + 1PD
- Example: $k = 9 = 1001_b$



Large Degree Isogeny Computations

- Vélu's fomulas is only suitable for small degree isogenies: $\phi_G : E \to E/G$

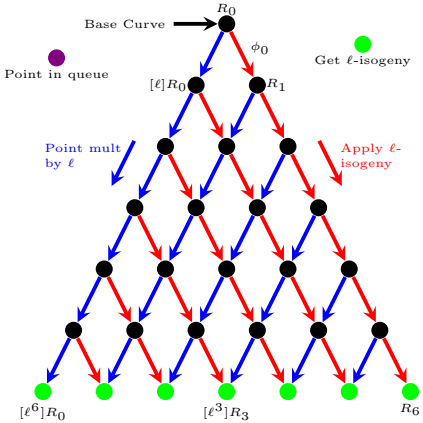


- Evaluate ϕ_G as a chain of small-degree isogenies
- Complexity: $O(e^2 \cdot \ell)$. Exponentially smaller than ℓ^e.
  - Jao and De Feo [2014]: Optimal strategy improves this to $O(e \log e \cdot \ell)$.
  - For SIDH we only use isogenies of degree ℓ^e for $\ell \in \{2, 3\}$.

Large Degree Isogeny Computations

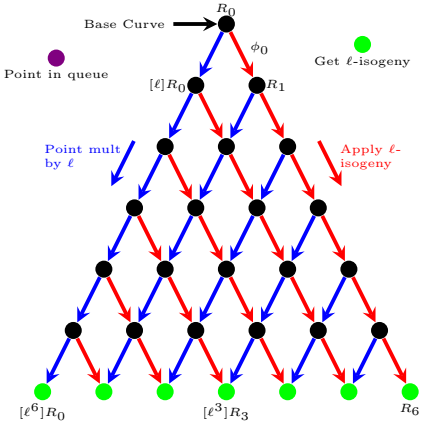
e.g., $\phi : E_0/\langle R_0 \rangle, ord(R_0) = \ell^7$

- 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 $R_{i+1} = \phi_i(R_i)$



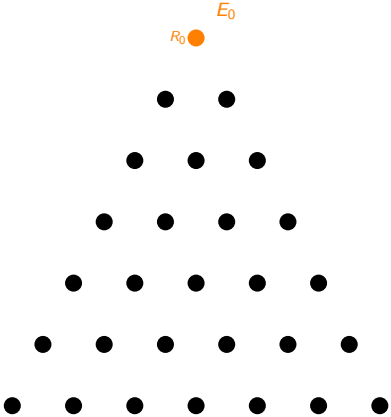
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$

Large Degree Isogeny Computations

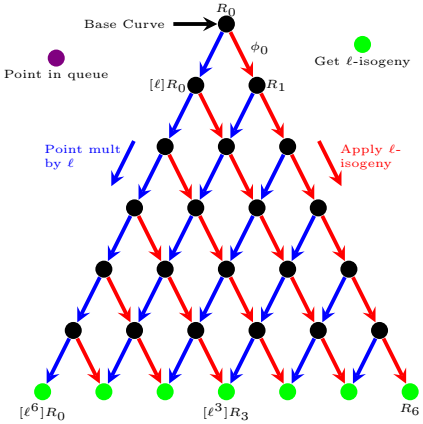


$e = 7$

Order of R_0 is ℓ^7

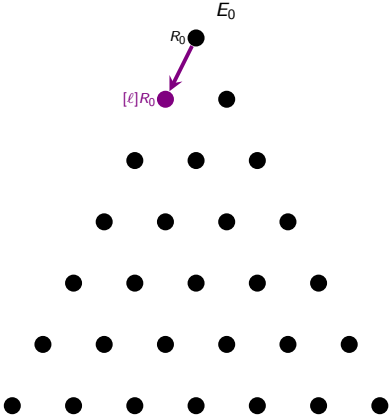


Large Degree Isogeny Computations

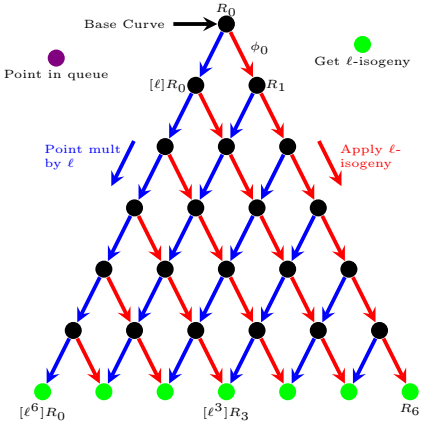


$e = 7$

Order of $[\ell]R_0$ is ℓ^6

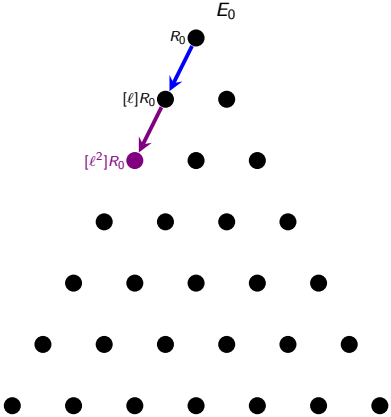


Large Degree Isogeny Computations

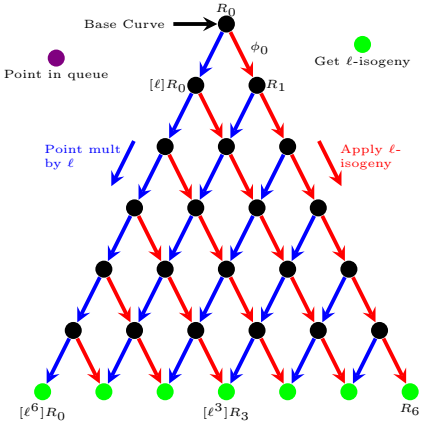


$e = 7$

Order of $[\ell^2]R_0$ is ℓ^5

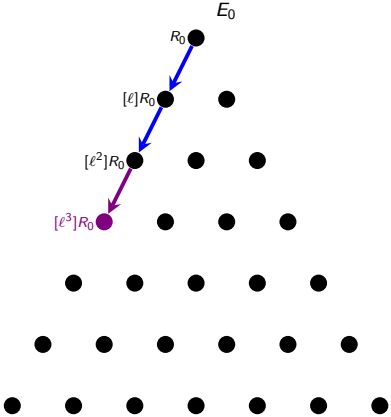


Large Degree Isogeny Computations

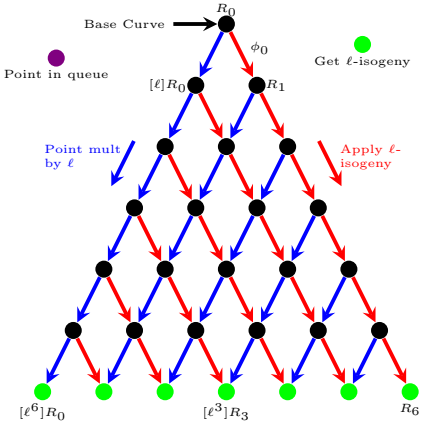


$e = 7$

Order of $[\ell^3]R_0$ is ℓ^4

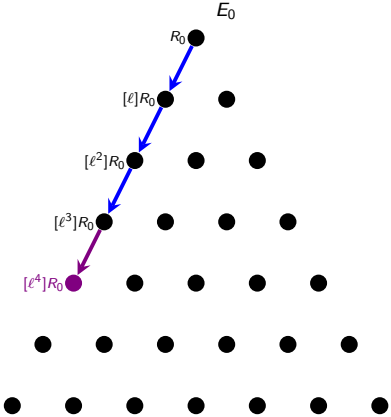


Large Degree Isogeny Computations

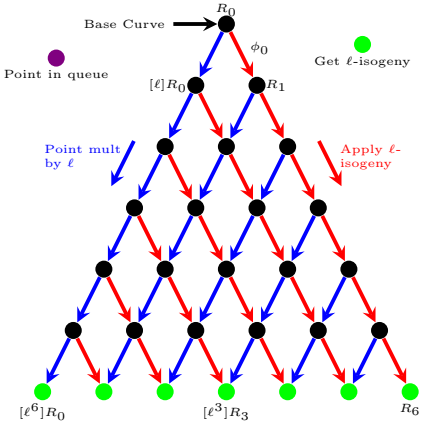


$e = 7$

Order of $[\ell^4]R_0$ is ℓ^3

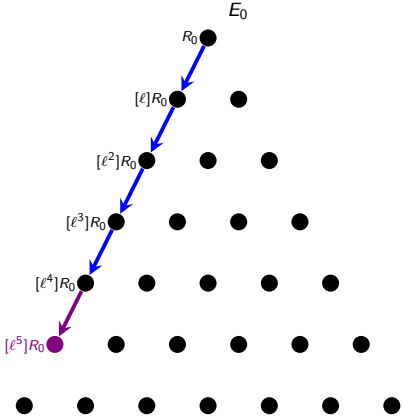


Large Degree Isogeny Computations

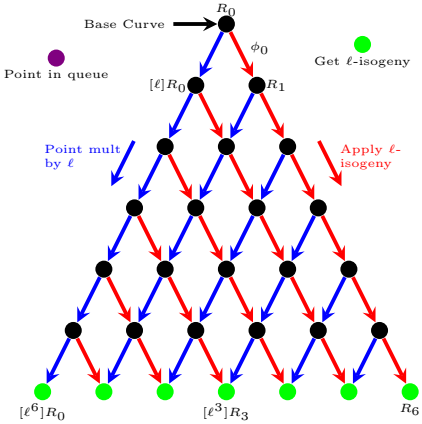


$e = 7$

Order of $[\ell^5]R_0$ is ℓ^2

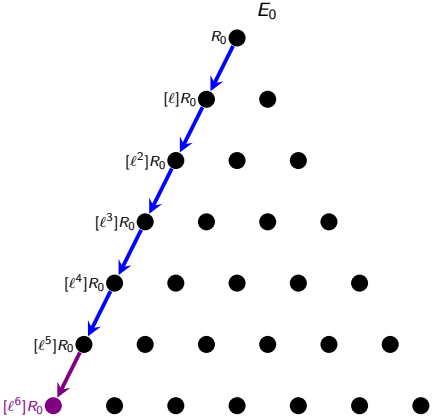


Large Degree Isogeny Computations

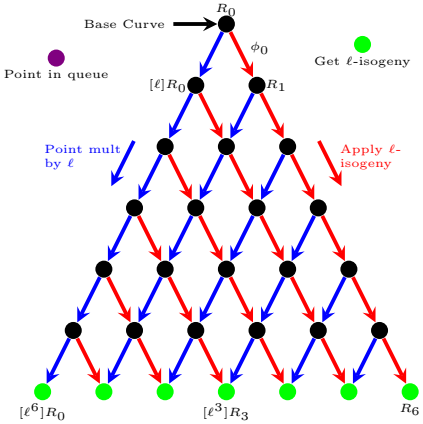


$e = 7$

Order of $[\ell^6]R_0$ is ℓ

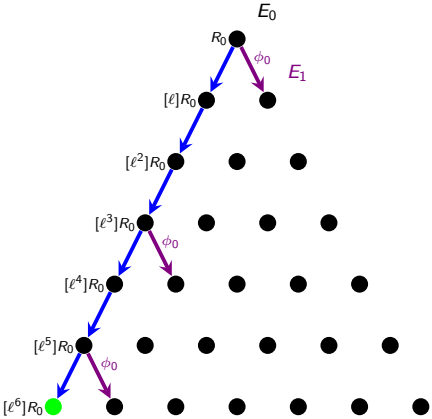


Large Degree Isogeny Computations

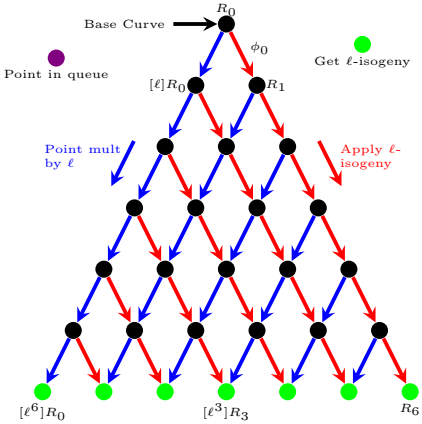


$e = 7$

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

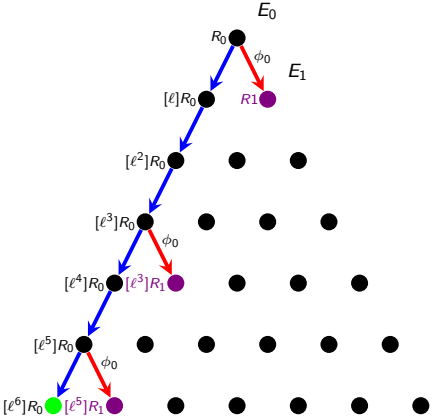


Large Degree Isogeny Computations

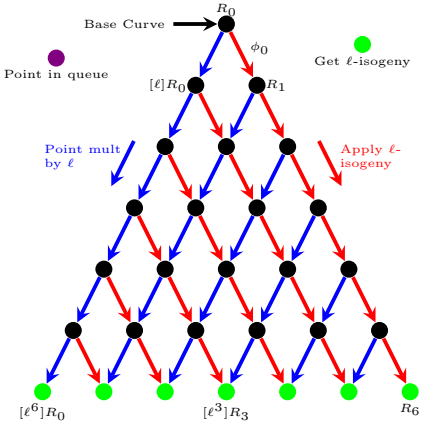


$e = 7$

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

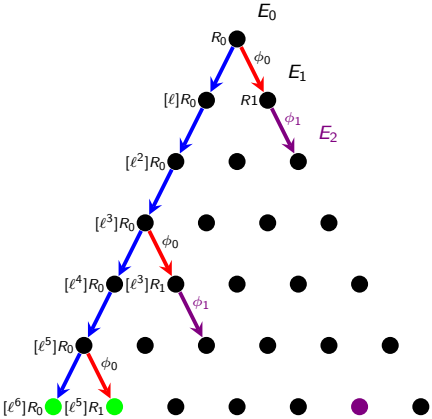


Large Degree Isogeny Computations

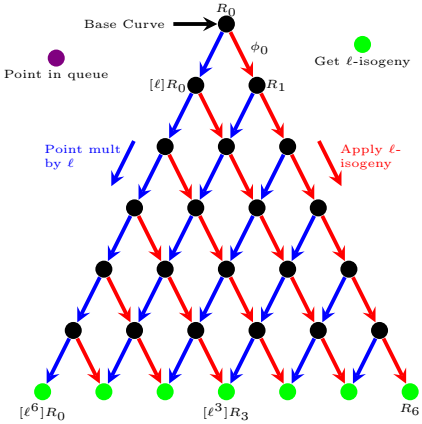


$e = 7$

$\phi_1 := E_1/\langle[\ell^5]R_1\rangle$
$E_2 = \phi_1(E_1)$

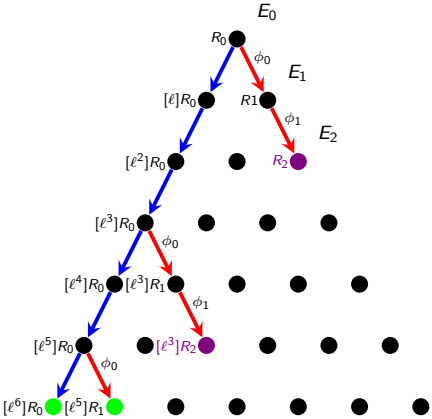


Large Degree Isogeny Computations

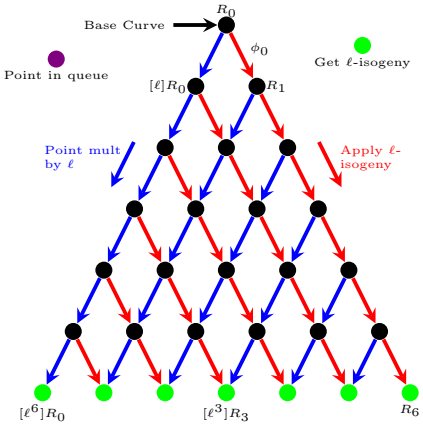


$e = 7$

$R_2 = \phi_1(R_1)$
Order of $[\ell^3]R_2$ is ℓ^2

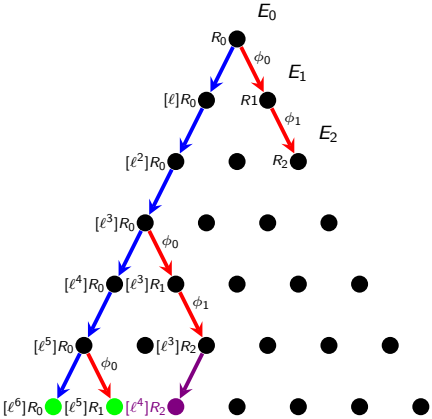


Large Degree Isogeny Computations

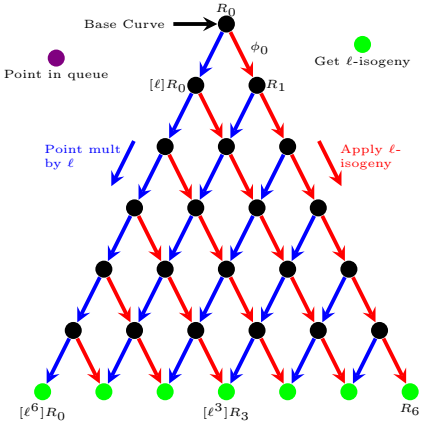


$e = 7$

Order of $[\ell^4]R_2$ is ℓ

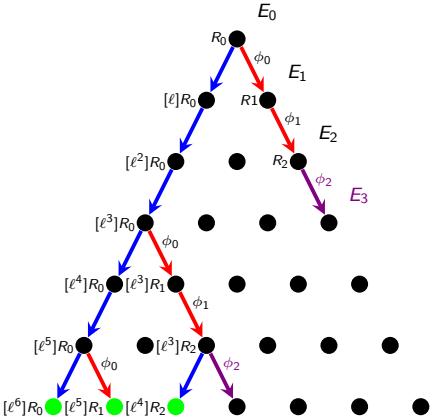


Large Degree Isogeny Computations

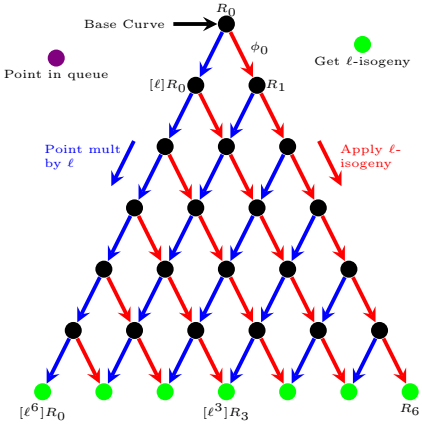


$e = 7$

$\phi_2 := E_2/\langle[\ell^4]R_2\rangle$
$E_3 = \phi_2(E_2)$

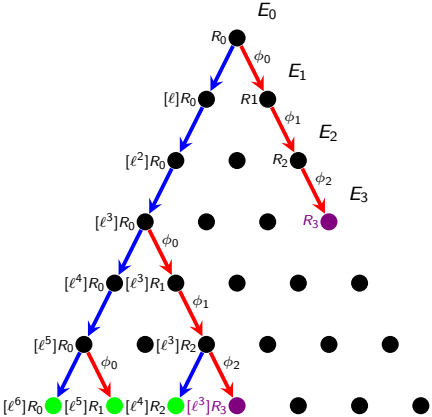


Large Degree Isogeny Computations

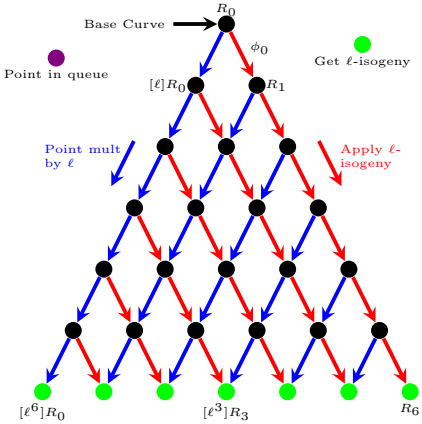


$e = 7$

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

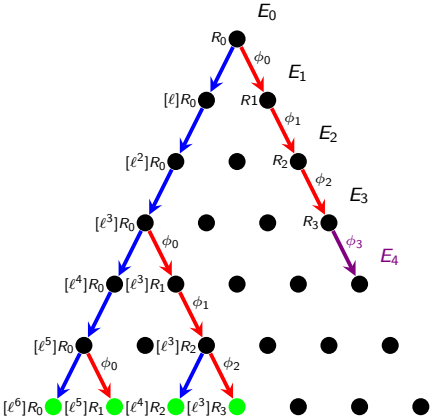


Large Degree Isogeny Computations

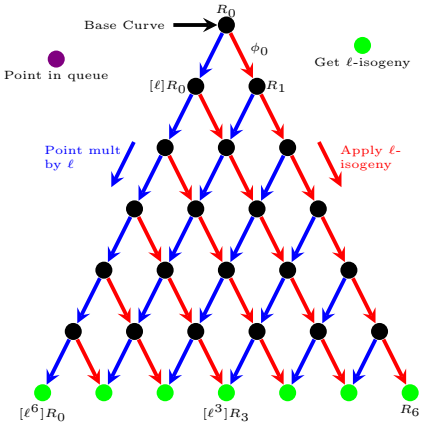


$e = 7$

$\phi_3 := E_3/\langle[\ell^3]R_3\rangle$
$E_4 = \phi_3(E_3)$

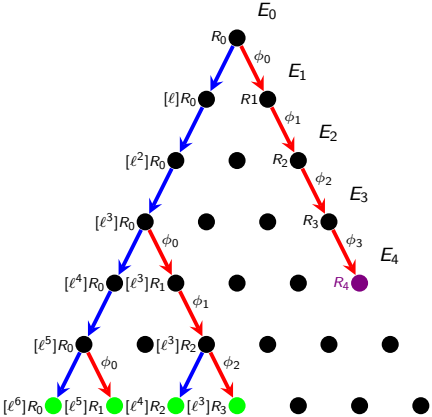


Large Degree Isogeny Computations

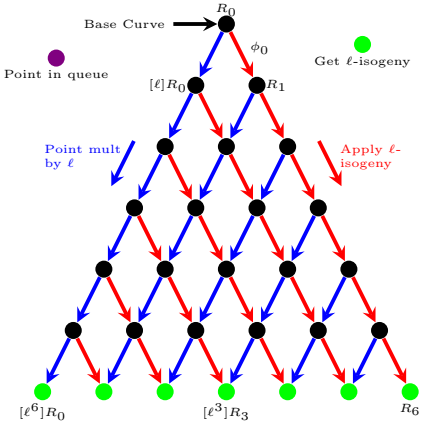


$e = 7$

$R_4 = \phi_3(R_3)$
Order of R_4 is ℓ^3

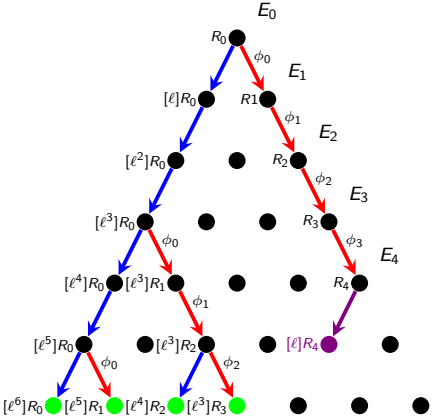


Large Degree Isogeny Computations

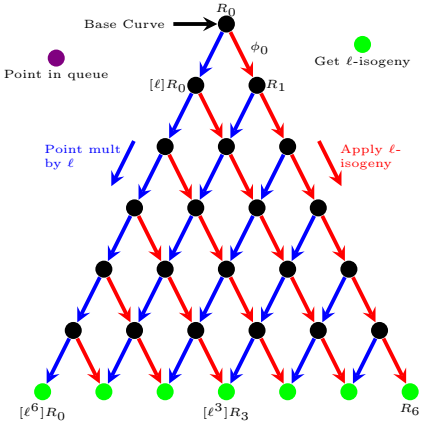


$e = 7$

Order of $[\ell]R_4$ is ℓ^2

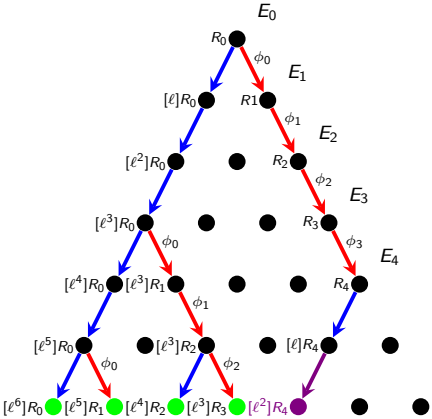


Large Degree Isogeny Computations

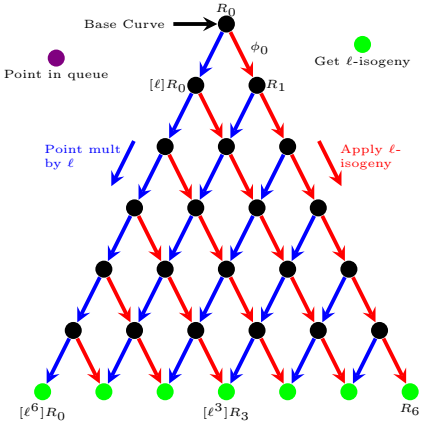


$e = 7$

Order of $[\ell^2]R_4$ is ℓ

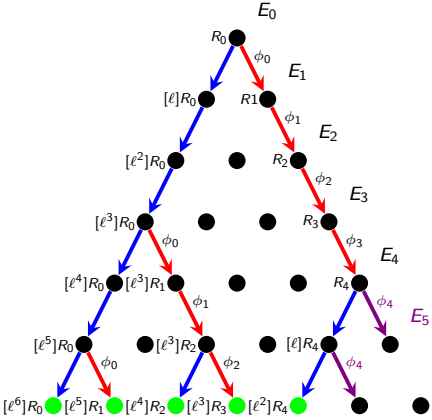


Large Degree Isogeny Computations

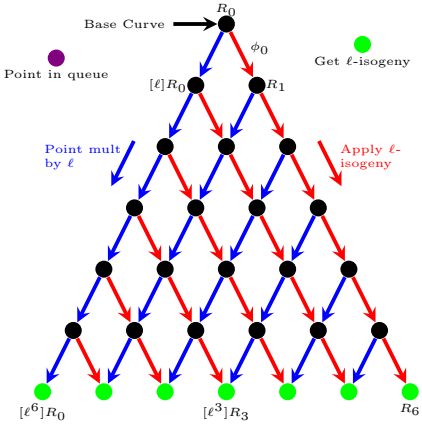


$e = 7$

$\phi_4 := E_4/\langle[\ell^2]R_2\rangle$
$E_5 = \phi_4(E_4)$

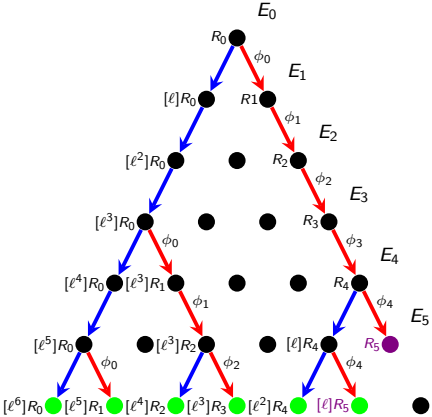


Large Degree Isogeny Computations

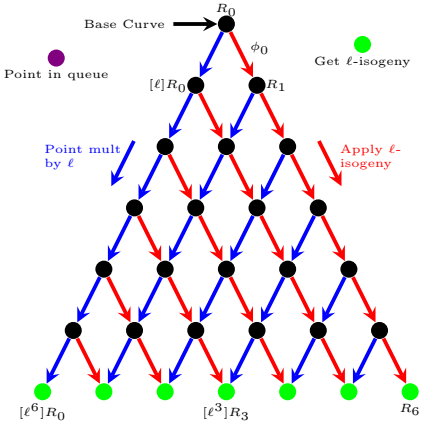


$e = 7$

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

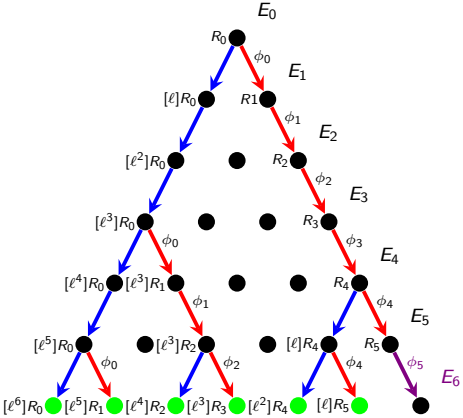


Large Degree Isogeny Computations

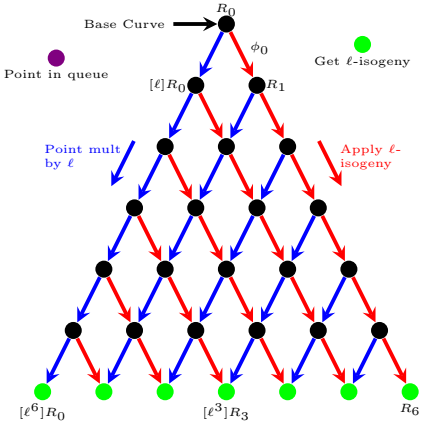


$e = 7$

$\phi_5 := E_5/\langle[\ell]R_5\rangle$
$E_6 = \phi_5(E_5)$

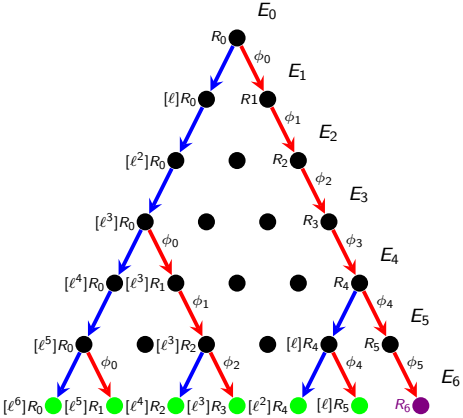


Large Degree Isogeny Computations

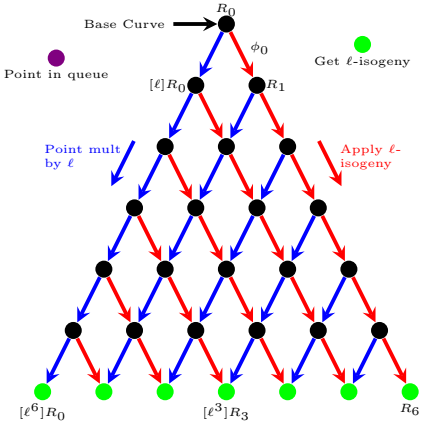


$e = 7$

$R_6 = \phi_5(R_5)$
Order of R_6 is ℓ

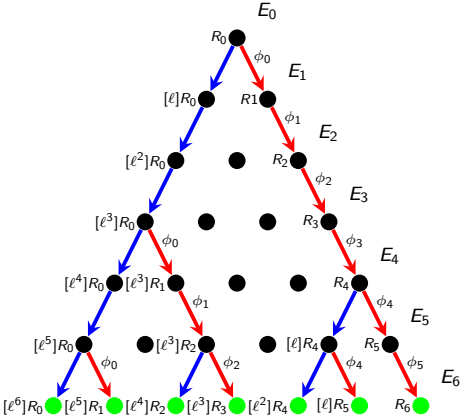


Large Degree Isogeny Computations

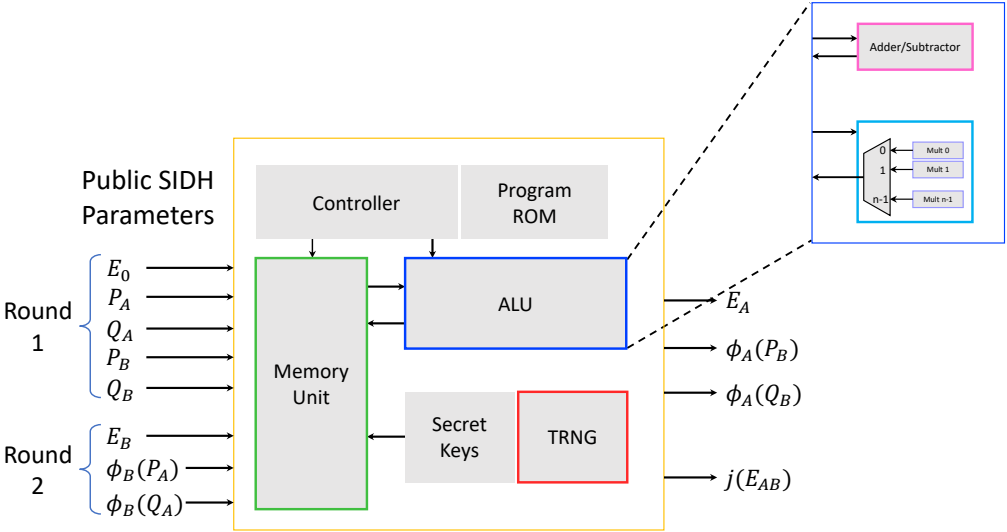


$e = 7$

$\phi_6 := E_6/\langle[\ell]R_6\rangle$
$E_7 = \phi_6(E_6)$

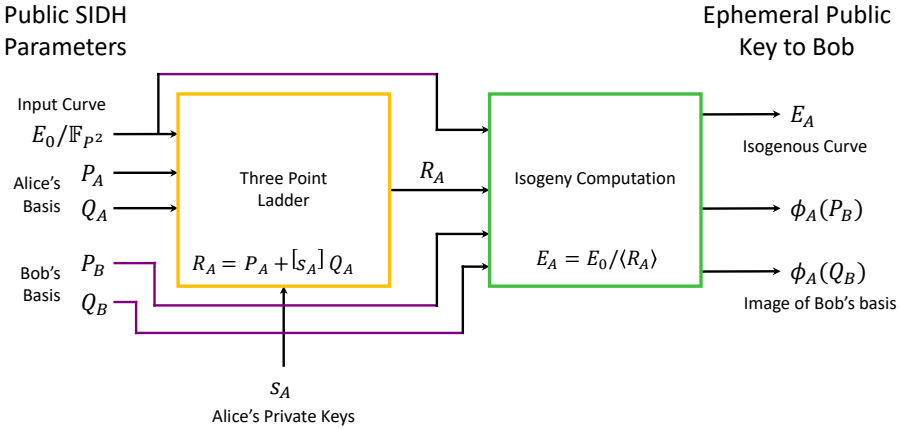


High-level Hardware Architecture for SIDH



Fast Kernel Computations

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



Field Multiplication

- Field multiplication performs $C = A \times B \mod p$
- Choice of multar multiplier is crucial: Montgomery multiplication
- Systloic 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 $w \leq e_A$

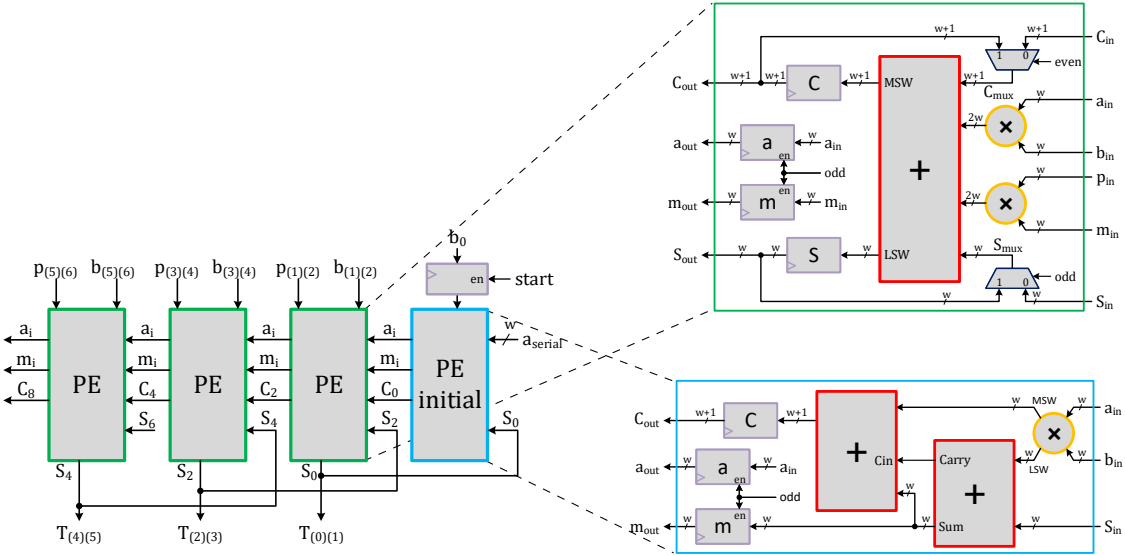
Coarsely Integrated Operand Scanning (CIOS):

- Alternate between multiplication and reduction
- Longer Critical Path: 1 Mult + 1 Additions
- 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)



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	$((a_0+a_1)\cdot(a_0-a_1), 2a_0 \cdot a_1)$	$2M+3A$
a^{-1}	$(a_0 \cdot (a_0^2+a_1^2)^{-1}, -a_1 \cdot (a_0^2+a_1^2)^{-1}$	$4M+2A+1I$

SIKE Control Flow

KEY GENERATION (BOB)

Bob's secret key s_B

Public Parameters
Alice's values
Bob's values

SIKE Control Flow

KEY GENERATION (BOB)

Isogeny

Bob's secret key s_B → $E_B = E_0/\langle P_B + [s_B]Q_B \rangle$

Public Parameters
Alice's values
Bob's values

SIKE Control Flow

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 $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

Public Parameters
Alice's values
Bob's values

SIKE Control Flow

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 $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

Public Parameters
Alice's values
Bob's values

KEY ENCAPSULATION (Alice)

Alice's secret message m

Bob's public key pk_B

SIKE Control Flow

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 $pk_B = \{E_B, \phi_B(P_A), \phi_B(Q_A)\}$

Public Parameters
Alice's values
Bob's values

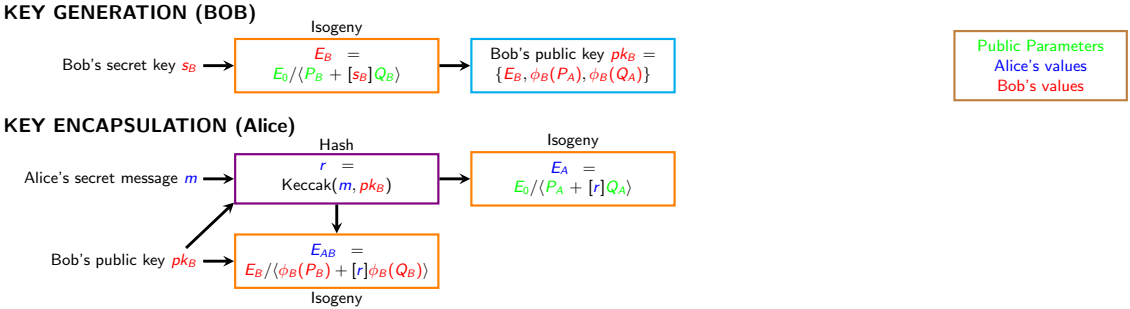
KEY ENCAPSULATION (Alice)

Hash

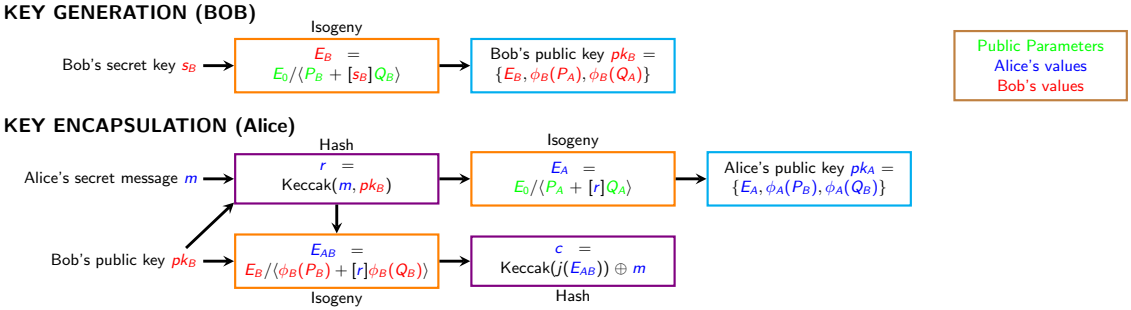
Alice's secret message m → $r = \text{Keccak}(m, pk_B)$

Bob's public key pk_B

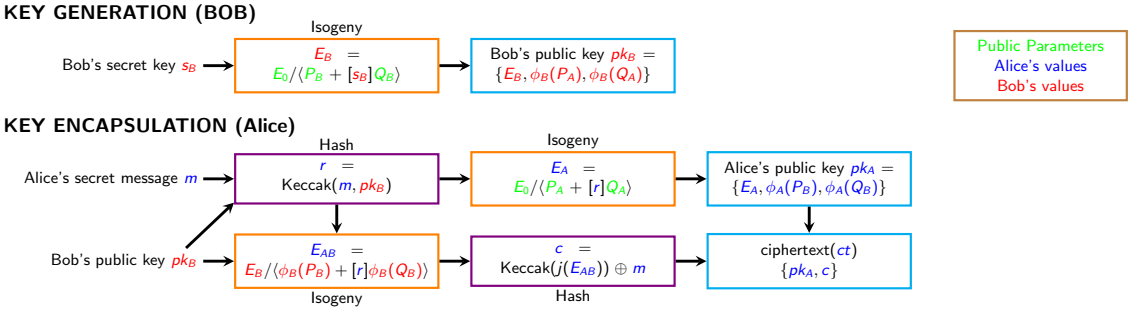
SIKE Control Flow



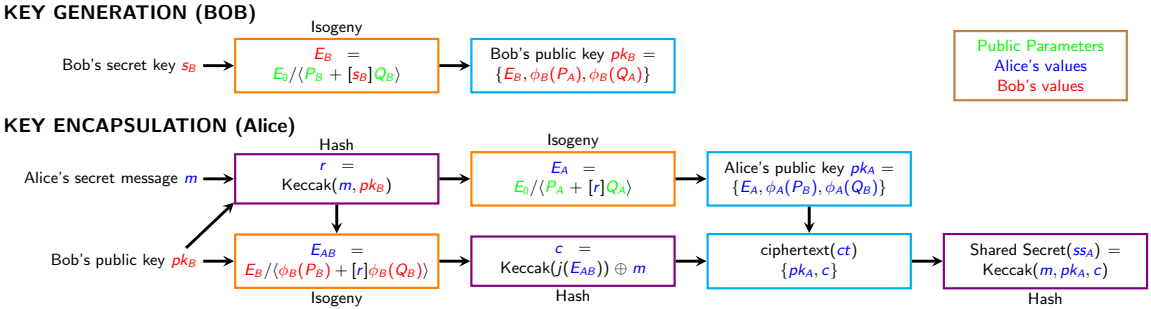
SIKE Control Flow



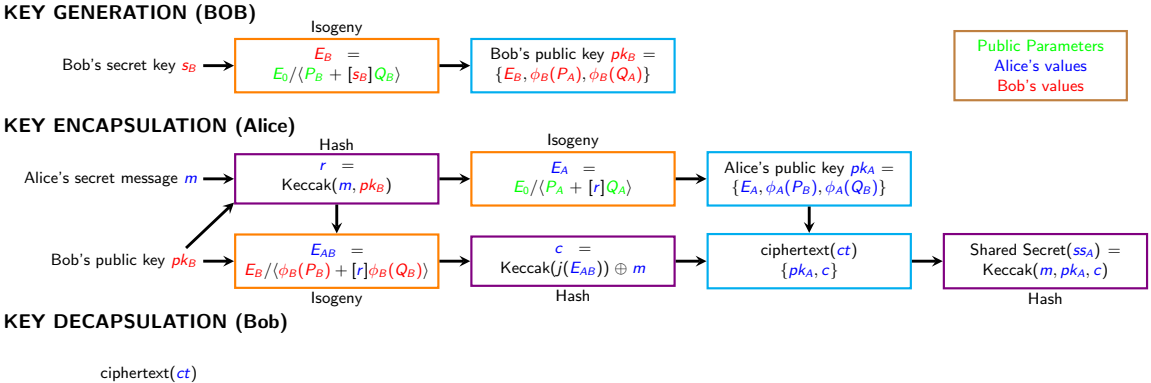
SIKE Control Flow



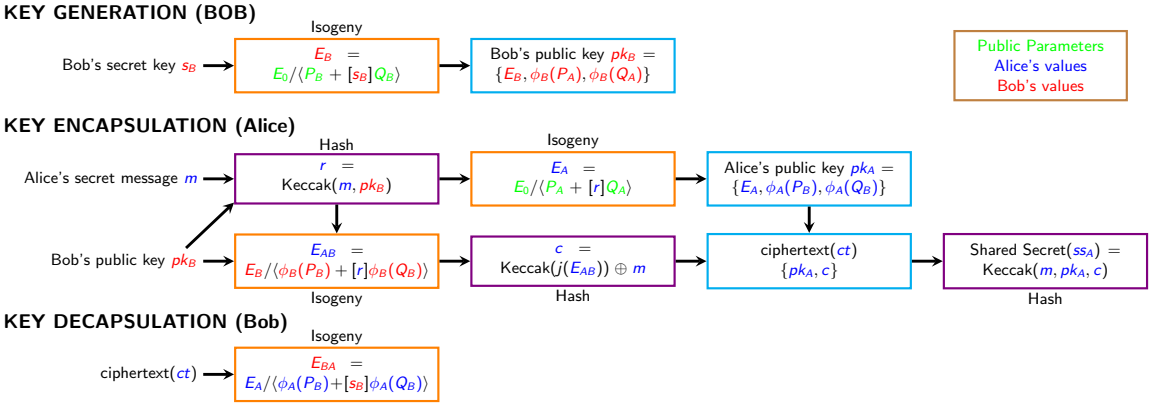
SIKE Control Flow



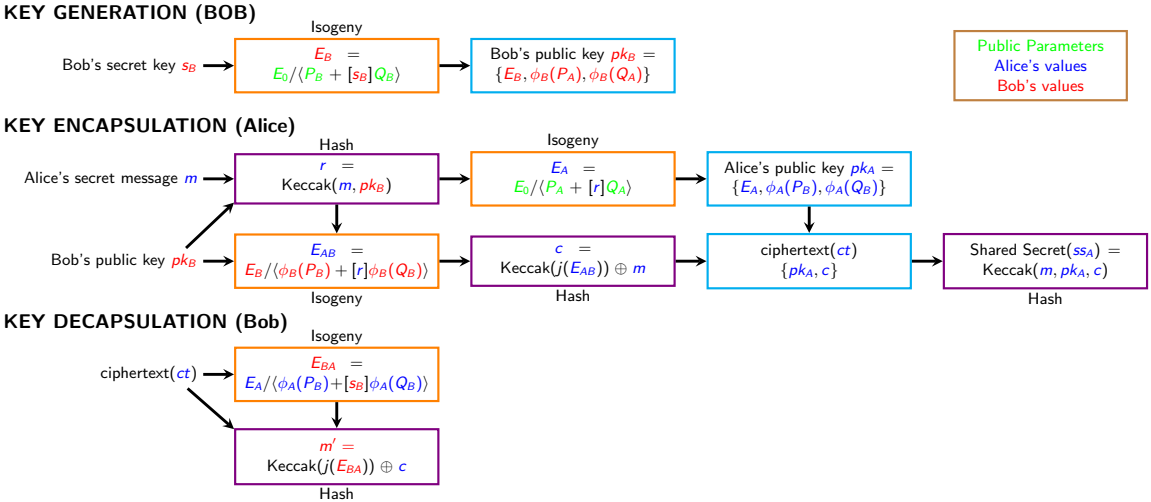
SIKE Control Flow



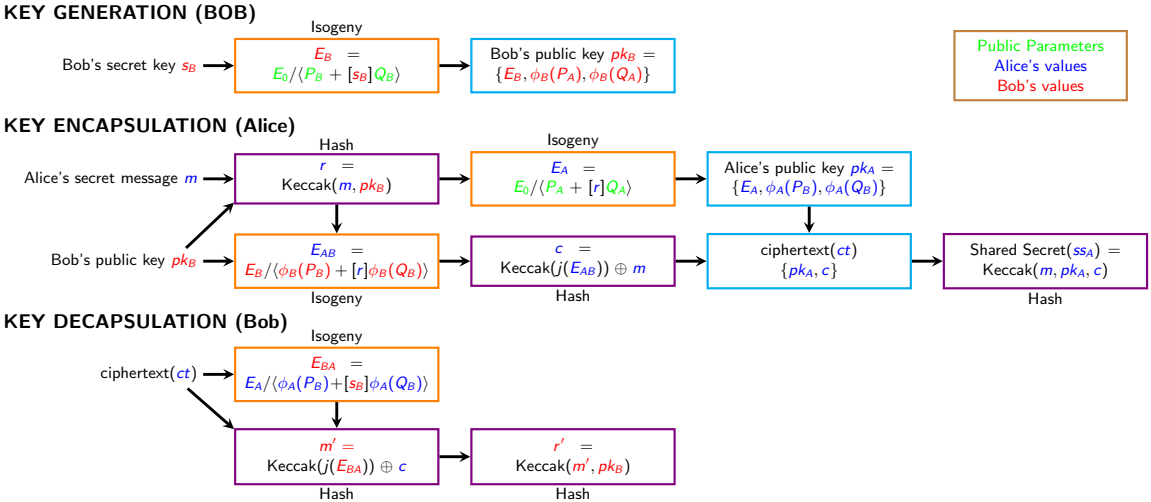
SIKE Control Flow



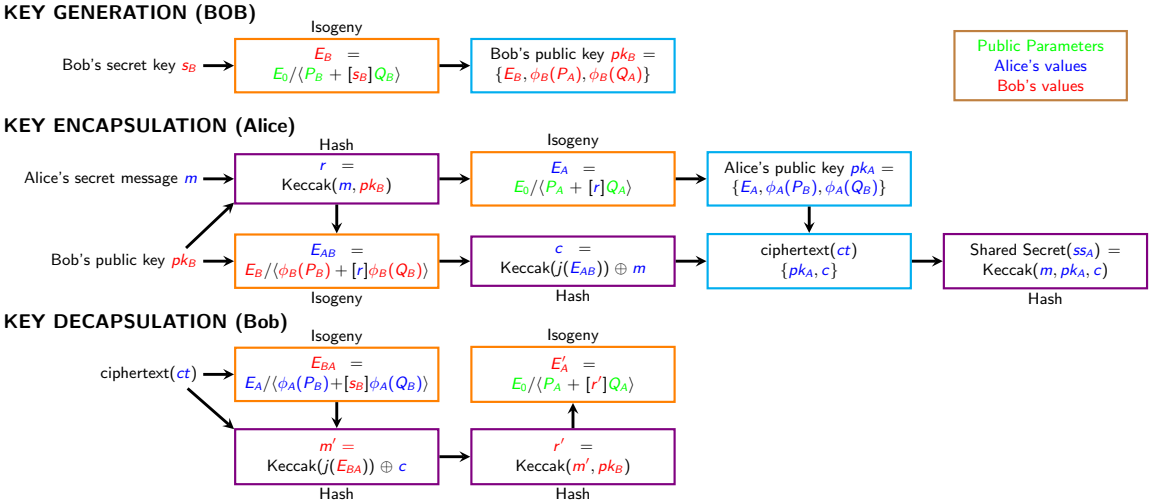
SIKE Control Flow



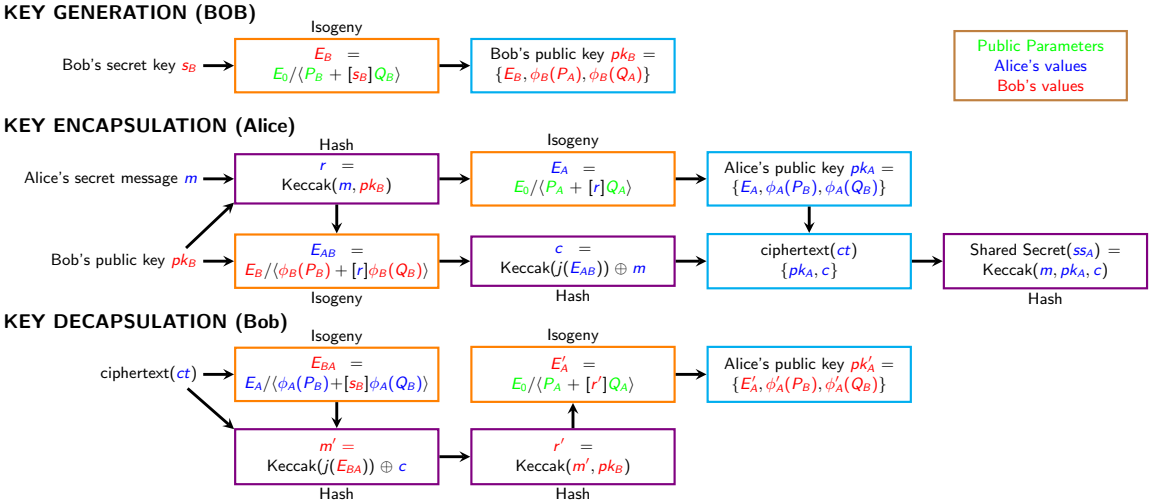
SIKE Control Flow



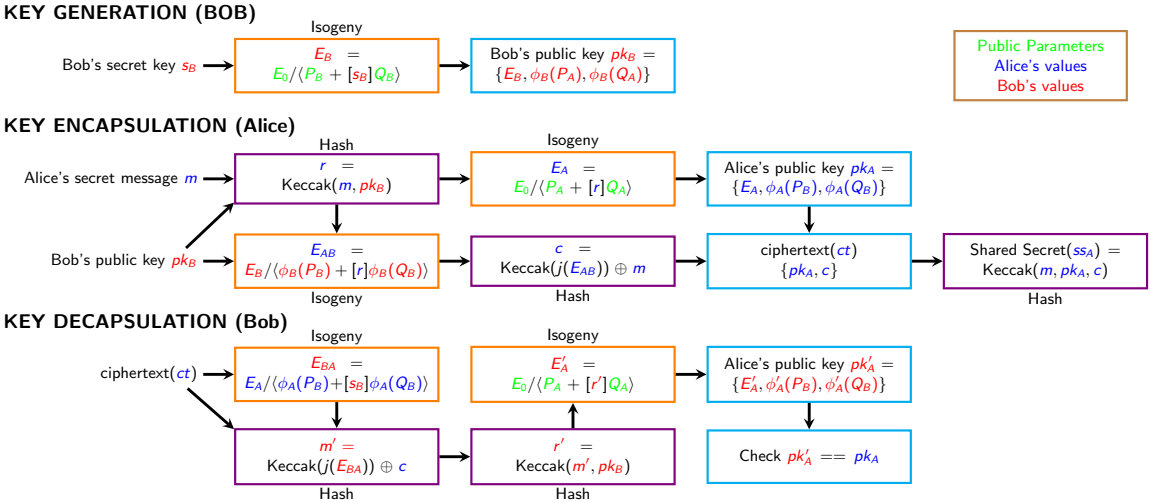
SIKE Control Flow



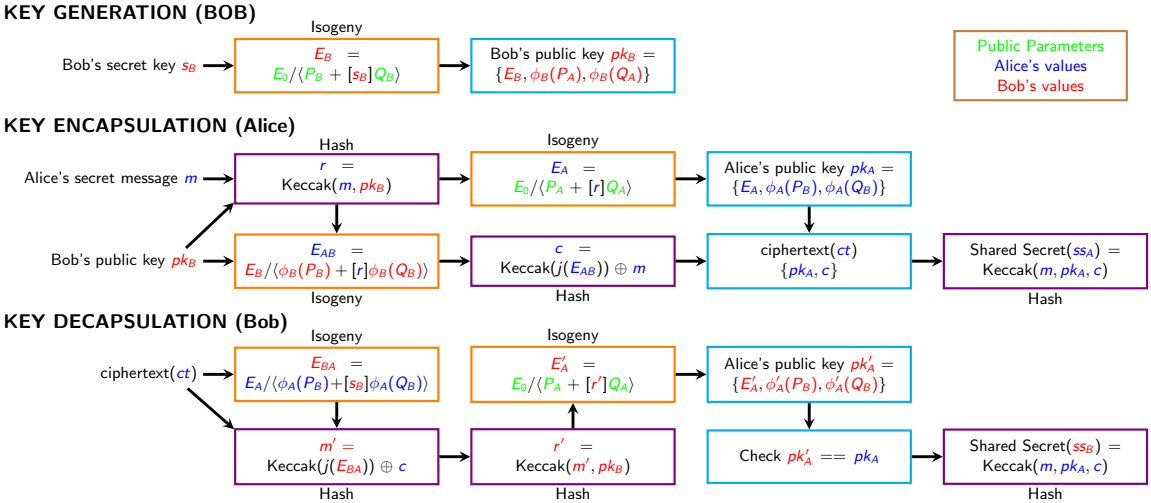
SIKE Control Flow



SIKE Control Flow

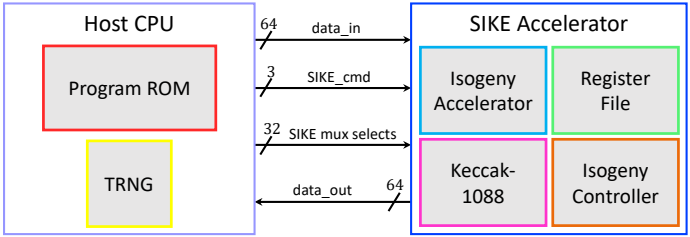


SIKE Control Flow



SIKE in FPGA

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



Isogeny Operations

Total number of $\mathbb{F}_{p^2}$ arithmetic operations in fastest known isogeny formulas (from SIKE submission)

Isogeny Operation	$\mathbb{F}_{p^2}$ Mult.	$\mathbb{F}_{p^2}$ Squaring	$\mathbb{F}_{p^2}$ Addition
xDBL	4	2	4
get_2_isog	0	2	1
eval_2_isog	4	0	6
xTPL	7	5	10
get_3_isog	2	3	12
eval_3_isog	4	2	4

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

SIDH and SIKE in PC

	SIDH	SIKE	CSIDH
Key size (quantum 128-bit)	330 bytes	330 bytes	64 bytes
Running time (x86-64)	2.5 ms	4.5 ms	50 ms
Compressed SIDH/SIKE	196 bytes 6.5 ms	195 bytes 8.6 ms	
Exponential quantum security	✓	✓	✗
Active attack security	✗	✓	✓
Direct key validation	✗	✗	✓
Digital signatures	✓	✗	✓
NIST candidate	✗	✓	✗

SIKE Performance in FPGA

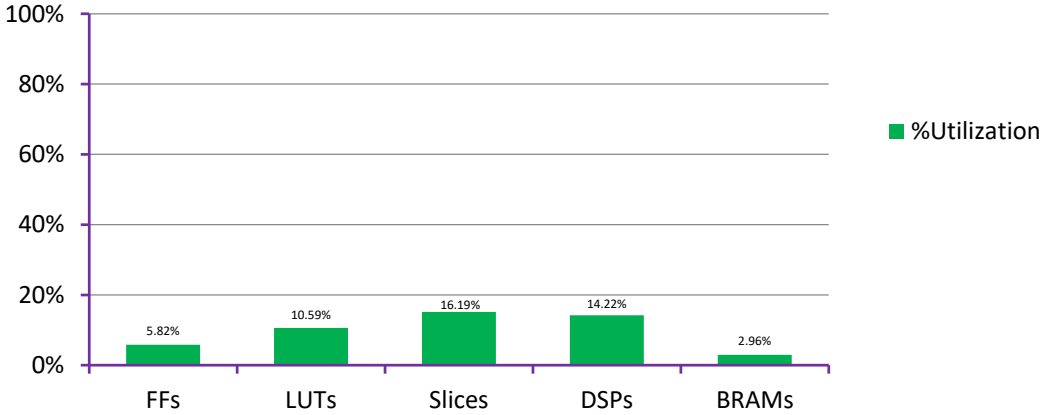
NIST-Round 1 Submission and updated in Round 2.

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

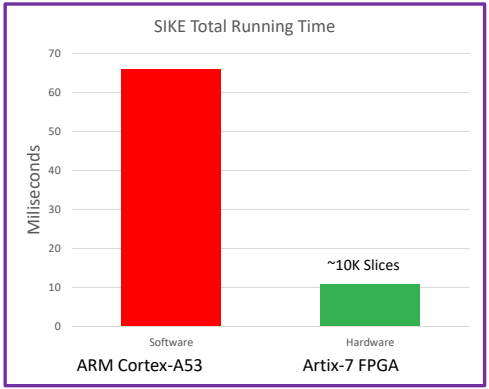
SIKE in FPGA Area Results

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

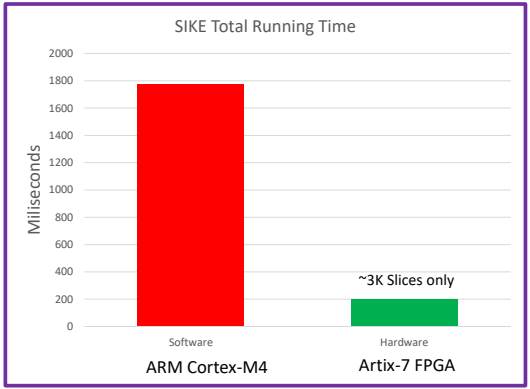


SIKE: Results for NIST level 1

Target: High Performance Edge



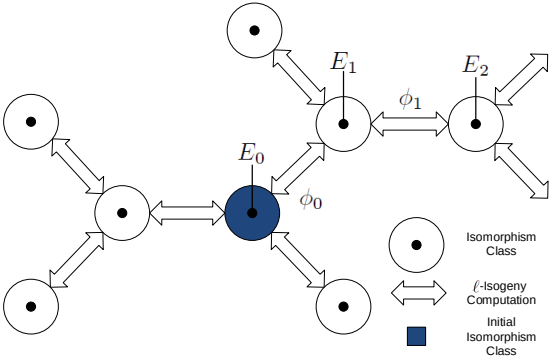
Target: Resource-constrained IoT



Side Channel Attacks on SIDH/SIKE

- Attacking isogeny computations: recover steps ϕ_i in secret walk ϕ
- Loop abort attack $\rightarrow$ Fault an implementation to stop isogeny operation early and give E_i
- Refined Power Analysis $\rightarrow$ Force zero-values to divulge ϕ_i

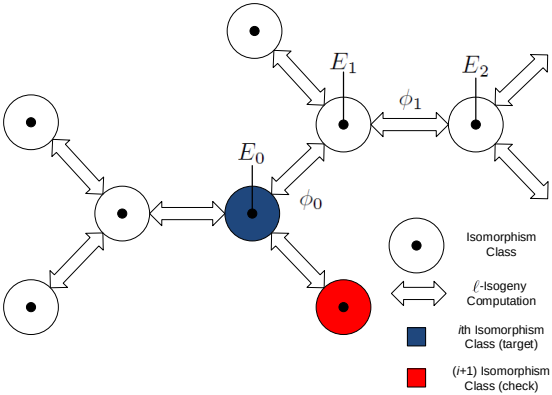
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$



Side Channel Attacks on SIDH/SIKE

- Attacking isogeny computations: recover steps ϕ_i in secret walk ϕ
- Loop abort attack $\rightarrow$ Fault an implementation to stop isogeny operation early and give E_i
- Refined Power Analysis $\rightarrow$ Force zero-values to divulge ϕ_i

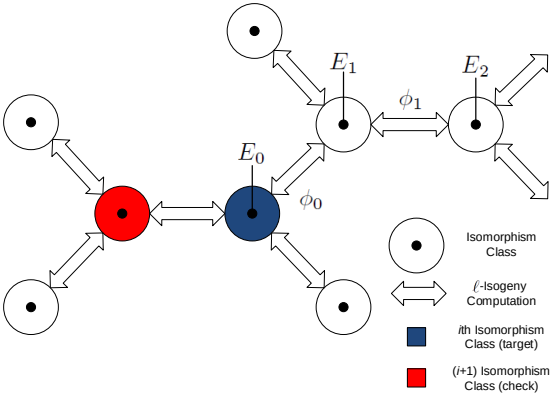
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$



Side Channel Attacks on SIDH/SIKE

- Attacking isogeny computations: recover steps ϕ_i in secret walk ϕ
- Loop abort attack $\rightarrow$ Fault an implementation to stop isogeny operation early and give E_i
- Refined Power Analysis $\rightarrow$ Force zero-values to divulge ϕ_i

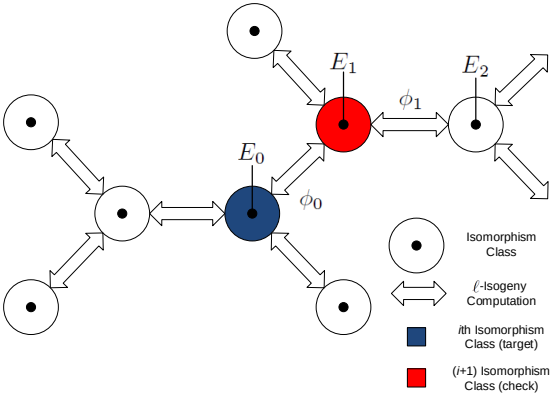
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$



Side Channel Attacks on SIDH/SIKE

- Attacking isogeny computations: recover steps ϕ_i in secret walk ϕ
- Loop abort attack $\rightarrow$ Fault an implementation to stop isogeny operation early and give E_i
- Refined Power Analysis $\rightarrow$ Force zero-values to divulge ϕ_i

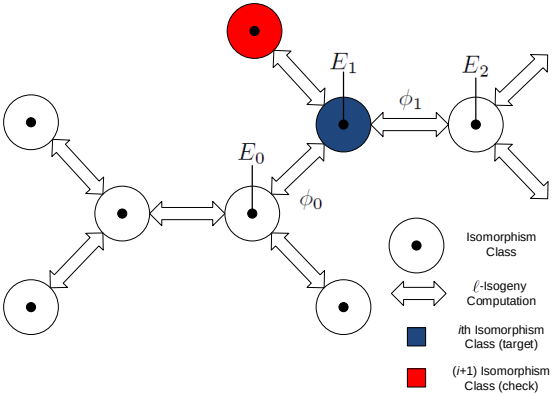
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$



Side Channel Attacks on SIDH/SIKE

- Attacking isogeny computations: recover steps ϕ_i in secret walk ϕ
- Loop abort attack $\rightarrow$ Fault an implementation to stop isogeny operation early and give E_i
- Refined Power Analysis $\rightarrow$ Force zero-values to divulge ϕ_i

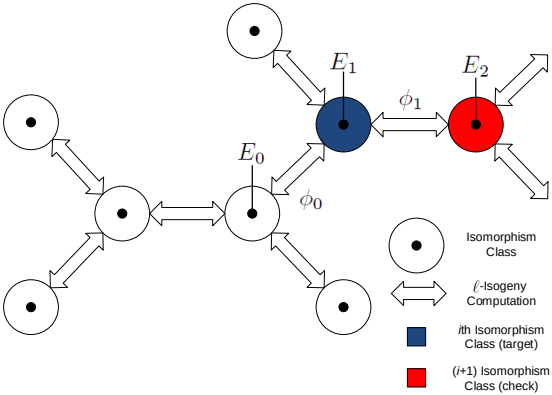
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$



Side Channel Attacks on SIDH/SIKE

- Attacking isogeny computations: recover steps ϕ_i in secret walk ϕ
- Loop abort attack $\rightarrow$ Fault an implementation to stop isogeny operation early and give E_i
- Refined Power Analysis $\rightarrow$ Force zero-values to divulge ϕ_i

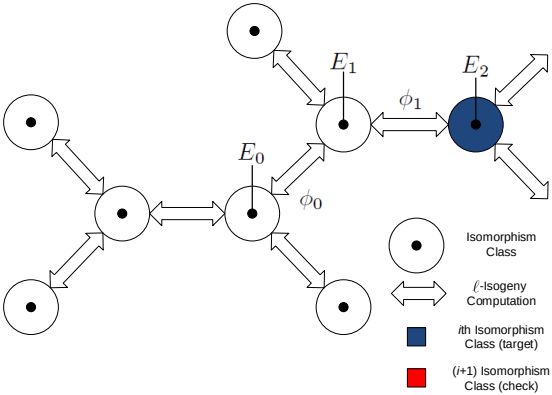
$$\phi = \phi_6 \cdot \phi_5 \cdot \phi_4 \cdot \phi_3 \cdot \phi_2 \cdot \phi_1 \cdot \phi_0$$



Side Channel Attacks on SIDH/SIKE

- Attacking isogeny computations: recover steps ϕ_i in secret walk ϕ
- Loop abort attack $\rightarrow$ Fault an implementation to stop isogeny operation early and give E_i
- Refined Power Analysis $\rightarrow$ Force zero-values to divulge ϕ_i

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



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.

Open Research Directions

- How to make isogenies FASTER?
  - Different curves, formulas, algorithms, isogeny base degrees, etc.
- How to design high-performance field arithmetic operators?
  - Addition, multiplication, inversion, etc.
- How can we attack and defend isogenies?
  - Side-channels, countermeasures, security analysis, etc.
- How to create efficient isogeny cryptosystems?
  - Signatures, hashing, PAKE, etc.
- How can we apply isogenies?
  - Blockchain, OT, broadcast encryption, etc.

Questions?

Thanks for your attention!