Implementation of isogeny-based cryptography

2/23/2021

NIST PQC Seminar Series

David Jao, University of Waterloo
and evolutionQ, Inc.

Isogenies

An isogeny is a non-constant morphism between elliptic curves which preserves base points.

For computational purposes, the emphasis is on separable isogenies over finite fields.

Let $\phi : E_1 \to E_2$ be an isogeny.

Define $\ker \phi = \{ P \in E_1 \mid \phi(P) = O_E \}$.

Obviously, ϕ determines $\ker \phi$.

Theorem: If ϕ is separable, then $\ker \phi$ determines ϕ up to isomorphism, and

$$\deg \phi = |\ker \phi|$$

Vélu's formulas (1971):

Let $S \subseteq E$ be a finite subgroup. Then

$$\phi_x(P) \stackrel{\text{def}}{=} x(P) + \sum_{Q \in S \setminus \{0_E\}} \left[x(P+Q) - x(Q) \right]$$

$$\phi_y(P) \stackrel{\text{def}}{=} y(P) + \sum_{Q \in S \setminus \{0_E\}} \left[y(P+Q) - y(Q) \right]$$

defines a separable isogeny $\phi(P) = (\phi_x(P), \phi_y(P))$ with $\ker \phi = S$.

Vélu's formulas require $O(d)$ operations ($d = \deg \phi$).

However, these operations take place in a field of definition for the points in $\ker \phi$.

- It is possible for an isogeny ϕ to be defined over F (equivalently, for $\ker \phi$ to be defined over F) even when not all points in $\ker \phi$ are defined over F.

- If extensions of a finite field are used, then Vélu's formulas cost $\approx O(d^4)$ basic operations
- $\sqrt{\text{élu}}$ (Bernstein et al. ia.cr/2020/341) is faster for large d

Couveignes Rostovtsev Stolbunov scheme (CRS)

(Devised in 1996, published 2006)

$$\begin{tikzcd} E \arrow[r, "\phi_{\mathfrak{a}}"] \arrow[d, "\phi_{\mathfrak{b}}"'] & \mathfrak{a} * E \arrow[d] \\ \mathfrak{b} * E \arrow[r] & \mathfrak{a} * \mathfrak{b} * E = \\ & \mathfrak{b} * \mathfrak{a} * E \end{tikzcd}$$

$E/\mathbb{F}_p$ ordinary

$\mathfrak{a}, \mathfrak{b} \in Cl(End(E))$

$*$ is the complex multiplication operator, defined by $\ker \phi_{\mathfrak{a}} = \bigcap_{\psi \in \mathfrak{a}} \ker \psi$.

- For this scheme to be feasible, $\mathfrak{a}$ and $\mathfrak{b}$ must be chosen to be smooth ideals.
- The scheme is still very slow because the kernel points are defined over a field extension.

Supersingular Isogeny Diffie-Hellman (SIDH)

(Jao, De Feo, and Plût, 2011)

Uses an assumption formulated by Charles, Goren, and Lauter (2006)

$$\begin{array}{ccc} E & \xrightarrow{\phi_A} & E_A \\ \phi_B \downarrow & & \downarrow \\ E_B & \longrightarrow & E_{AB} \end{array}$$

$E/\mathbb{F}_p$ supersingular, $\boxed{p = 2^e 3^f - 1}$, $tr(E) = 0$.

$E[2^e] \cong (\mathbb{Z}/2^e)^2$
$E[3^f] \cong (\mathbb{Z}/3^f)^2$ both defined over $\mathbb{F}_{p^2}$

Alice sends $(E_A, \phi_A|_{E[3^f]})$ $\ker \phi_A \subset E[2^e]$, $\deg \phi_A = 2^e$

Bob sends $(E_B, \phi_B|_{E[2^e]})$ $\ker \phi_B \subset E[3^f]$, $\deg \phi_B = 3^f$

- Since all points are defined over $\mathbb{F}_{p^2}$, "fast" Vélu can be used. Furthermore, only isogenies of degree 2 or 3 are needed.

Commutative SIDH (CSIDH)

(Castryck, Lange, Martindale, Panny, Renes, 2018)

$$\begin{array}{ccc} E & \longrightarrow & \mathfrak{a} * E \\ \downarrow & & \downarrow \\ \mathfrak{b} * E & \longrightarrow & \mathfrak{a} * \mathfrak{b} * E \end{array}$$

$E/\mathbb{F}_p$ supersingular $\quad \mathfrak{a}, \mathfrak{b} \in Cl(\mathrm{End}_{\mathbb{F}_p}(E))$

$$p = 4 \prod_{i=1}^{n} \ell_i - 1 \quad , \ tr(E) = 0$$

- Construction of E and p allows the use of isogenies of degree ℓ_i with all kernel points defined over $\mathbb{F}_{p^2}$.
- Hence "fast" Vélu can be used, so CSIDH is faster than CRS.
- CSIDH is slower than SIDH (because CSIDH uses degrees $\ell_i \geq 3$ and SIDH uses degree $= 2$ or 3)

Security of SIDH / SIKE

SIDH can be broken by an invalid key attack

- Galbraith, Petit, Shani, Ti   ia.cr/2016/859

SIKE = SIDH + HHK / FO

Proposed parameter sizes:

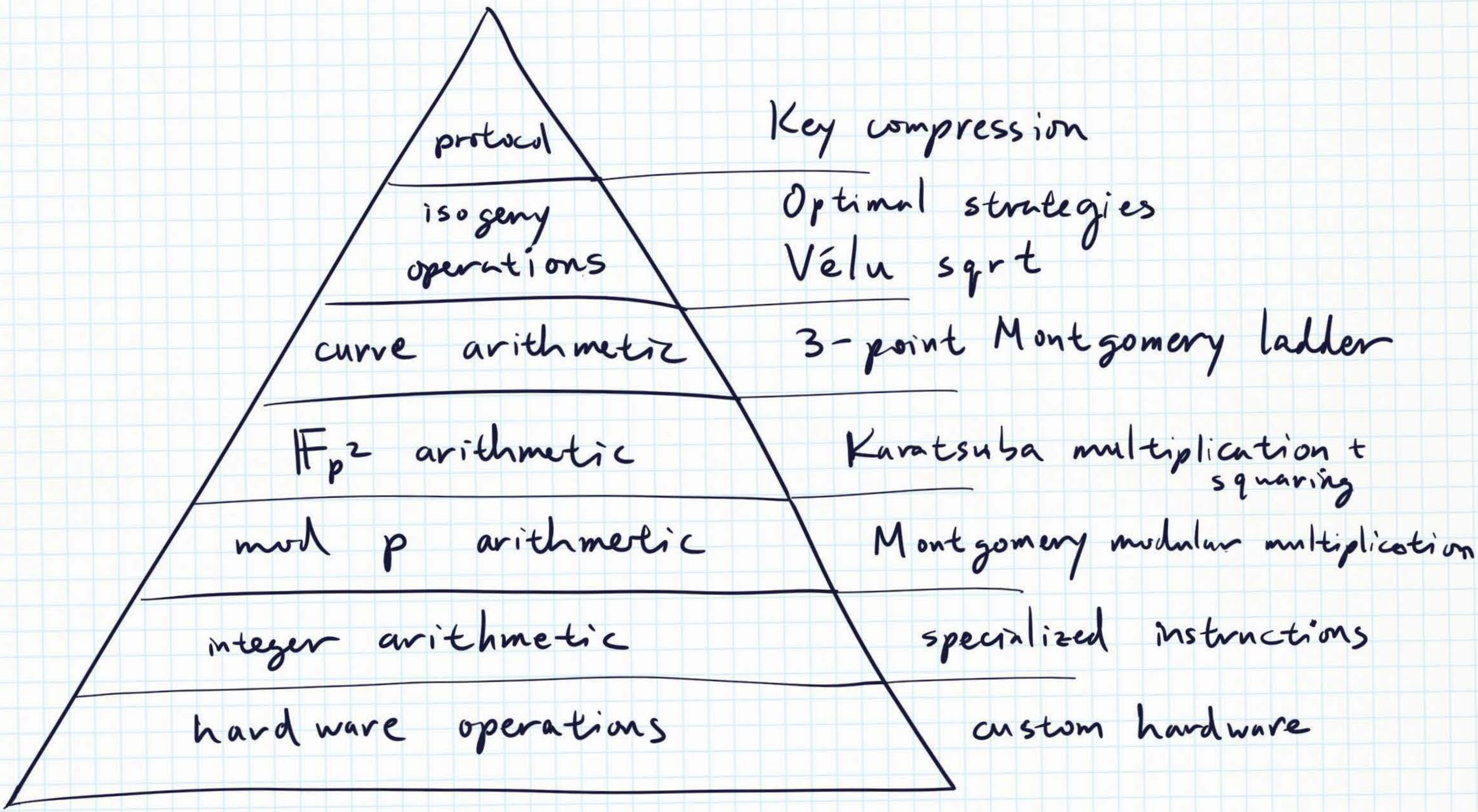
$\log_2 p$	|PK| (bytes)	NIST level
434	330	1
503	378	2
610	462	3
751	564	5

Security analysis:

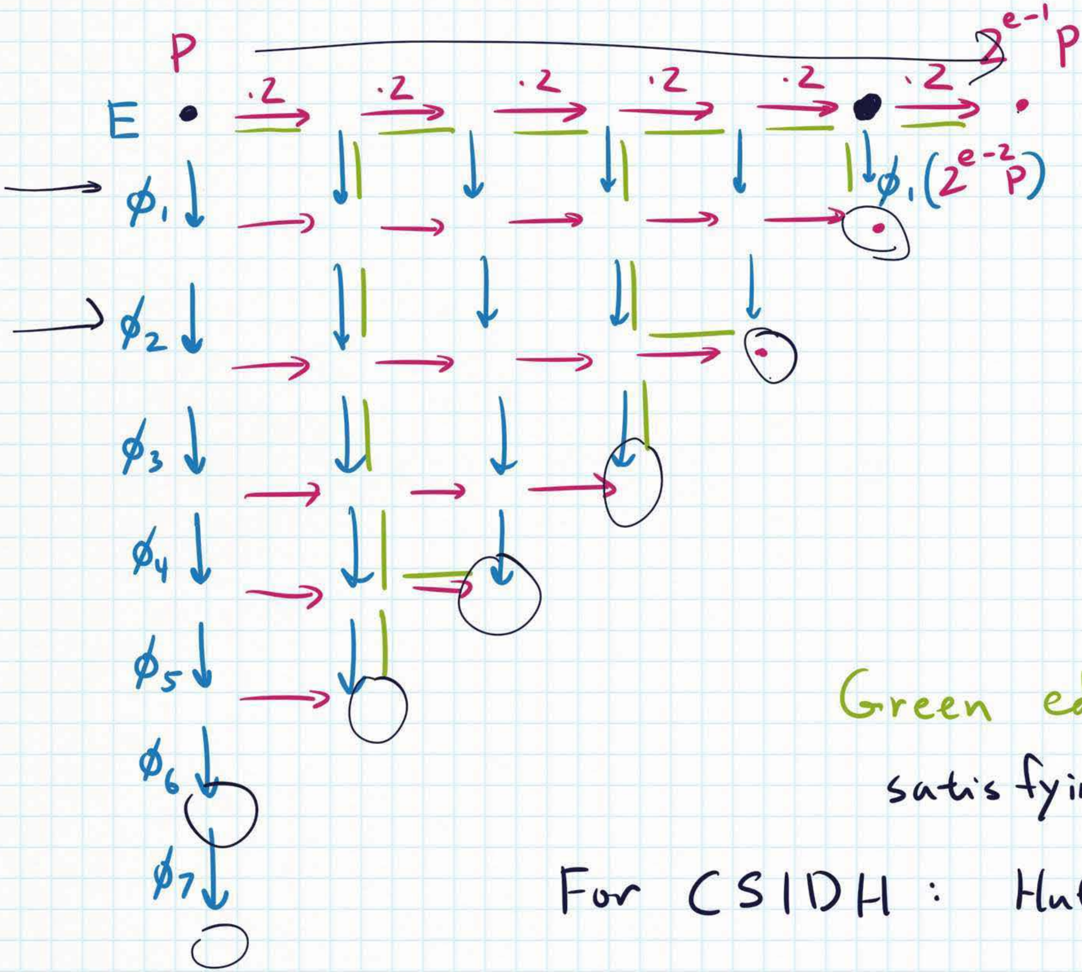
Longa, Wang, Szefer   ia.cr/2020/1457

Jaques & Schrottenloher   ia.cr/2020/424

Jaques & Schanck,   ia.cr/2019/103


protocol
isogeny operations
curve arithmetic
$\mathbb{F}_{p^2}$ arithmetic
mod p arithmetic
integer arithmetic
hardware operations
Key compression
Optimal strategies
Vélu sqrt
3-point Montgomery ladder
Karatsuba multiplication + squaring
Montgomery modular multiplication
specialized instructions
custom hardware

Optimal strategies



Let $\ker \phi = \langle P \rangle$, $\operatorname{ord}(P) = 2^e$
$= \deg \phi$.

Write $\phi = \phi_1 \circ \phi_2 \circ \cdots \circ \phi_e$
where $\deg \phi_i = 2$, $i = 1$ to e.

Then:

$$\ker \phi_1 = \langle 2^{e-1} P \rangle$$

$$\ker \phi_2 = \langle \phi_1(2^{e-2} P) \rangle = \langle 2^{e-2} \phi_1(P) \rangle$$

etc.

Green edges denote optimal subgraph satisfying all required dependencies

For CSIDH: Hutchinson et al., ia.cr/2019/1121

Protocol optimizations – Point compression for SIDH

In SIDH, Alice sends $(E_A, \phi_A|_{E[3^f]})$ to Bob.

- Naive method: Choose basis $\{P, Q\}$ for $E[3^f]$, send coefficients of E_A and $\phi_A(P)$, $\phi_A(Q)$ to Bob.
- Compressed representation (saves 50% key size):
  - i) Fix a basis $\{P_A, Q_A\}$ for $E_A[3^f]$.
  - ii) Send the coefficient matrix of $\{\phi_A(P), \phi_A(Q)\}$ with respect to $\{P_A, Q_A\}$.
  - iii) Send an isomorphism invariant of E_A instead of E_A.

Latest results: Pereira et al. ia.cr/2020/431

- Code available at https://github.com/microsoft/PQCrypto-SIDH
- 21% faster encryption, 76% faster decryption compared to previous compressed SIDH.

Curve arithmetic

Montgomery curves ($By^2 = x^3 + Ax^2 + x$) are the current efficiency leaders, but other models have been studied:

- Huff (ia.cr/2020/526)
- Hessian (ia.cr/2019/1480)
- Edwards (ia.cr/2019/843, ia.cr/2019/110)
- Edwards + Huff (ia.cr/2011/430)

$\mathbb{F}_{p^2}$ arithmetic

$$\mathbb{F}_{p^2} = \mathbb{F}_p[i] \quad , \quad i = \sqrt{-1}$$

$$(a+bi)(c+di) = (ac - bd) + (ad + bc)\,i$$
$$= (ac - bd) + [(a+b)(c+d) - ac - bd]\,i$$

Karatsuba

$$(a+bi)^2 = (a^2 - b^2) + 2ab\,i = (a+b)(a-b) + 2ab\,i$$

Mod p arithmetic

Montgomery representation

- Karatsuba
- Squaring

Polynomial representation (Bouvier & Imbert ia.cr/2020/1385)

Specialized instructions

Example: UMAAL on ARM

$$a, b, c, d \mapsto (c, d \leftarrow a \cdot b + c + d)$$

a	b	c	d	$a \cdot b + c + d$
9	6	0	0	54
8	6	0	0	48
9	5	5	8	58
8	5	4	5	49
9	4	0	0	36
7	6	6	9	57
8	4	3	4	39
7	5	9	5	49
7	4	3	4	35

```
      456
  ×   789
  -------
       54
      48
      45
      40
      36
     42
     32
    35
 + 28
  -------
   359784
```

ARM Cortex M4 implementation

Anastasova, Azarderakhsh, Mozaffari-Kermani (ia.cr/2021/115)

p (bits)	434	503	610	751
time (ms)	850	1200	2430	3690
% improvement over SIKE rd.3	22.5 %	21.6	17.5	19.5

Key ideas :

- interleave additions / subtractions
- reduce mod p+1 instead of mod p
- use floating point registers as faster memory

Dedicated hardware and HW/SW codesign

p	434		503		610		751	
	time (ms)	AT (1000's)	time (ms)	AT (1000's)	time (ms)	AT (1000's)	time (ms)	AT (1000's)
[1]	11.3	393	14.1	545	21.6	900	27.8	1984
[2]	8.8	248	11.8	378	19.1	732	25.5	1542
[3] small	50.4	565	59.5	667	107.2	1202	179.6	2014
[3] fast	24.3	666	28.7	787	51.8	1420	60.8	1666
[4]	19.2	305	25.1	398	38.7	614	55.0	872

[1] Koziel et al. ia.cr/2019/711

[2] Elkhatibet al. ARITH 2020

[3] Massolino et al. ia.cr/2020/040

[4] Elkhatib et al., under review

[1,2] - pure hardware

[3,4] - HW/SW codesign