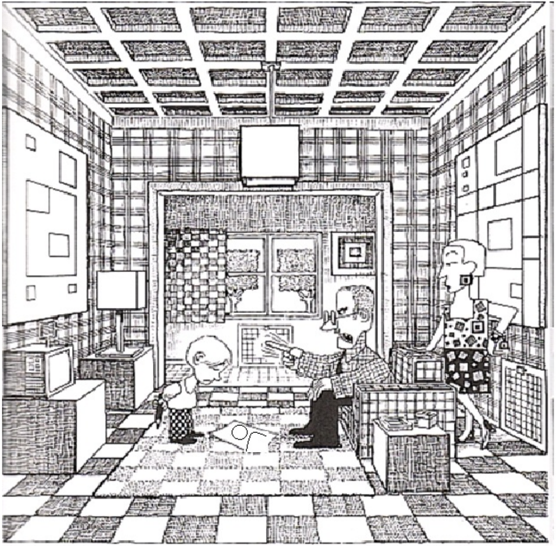
On new $\sqrt{\text{é}}$lu's formulae and their applications to CSIDH and B-SIDH constant-time implementations

Gora Adj, Jesús-Javier Chi-Domínguez and
Francisco Rodríguez-Henríquez

Universitat de Lleida
Tampere University

Crypto reading group at University of Waterloo, October.8.2020

Context



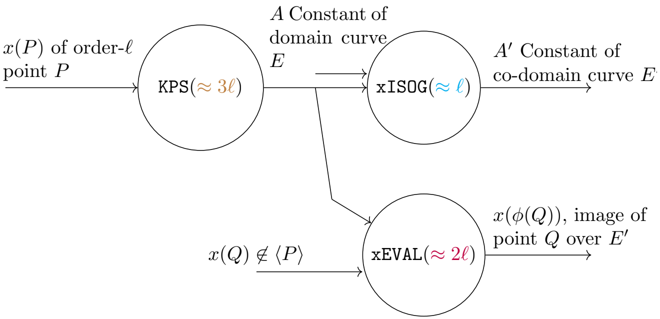
Computing degree-ℓ isogenies using $\sqrt{\text{é}}$lu's formulas

- Since July 1971, Vélu's formulae have been used to construct and evaluate degree-ℓ isogenies in the vast majority of isogeny-based cryptographic primitives.
- Traditional Vélu's formulae have a combined computational expense of about 6ℓ field operations to construct and evaluate degree-ℓ isogenies.
- Recently, Bernstein, de Feo, Leroux and Smith presented in ANTS'2020 [BFLS ANTS'20], a new approach for computing degree-ℓ isogenies at a reduced cost of just $\tilde{O}(\sqrt{\ell})$ field operations.

Computing degree-ℓ isogenies using $\sqrt{\text{é}}$lu's formulas

- Nevertheless, the authors of [BFLS ANTS'20] left open several intriguing research lines such as,
  - The concrete computational and memory cost of the novel $\sqrt{\text{é}}$lu formulae
  - The concrete impact of the $\sqrt{\text{é}}$lu formulae on constant-time implementations of relevant isogeny-based protocols

Computing degree-ℓ isogenies using Vélu's formulas



- For decades now, Vélu's formulae have been widely used to construct and evaluate degree-ℓ isogenies, using three main blocks,
  - KPS [Sort of a pre-computation building block. Cost: $\approx (3\ell)$**M**]
  - xISOG [Finds the image curve. Cost: $\approx (\ell)$**M**]
  - xEVAL [Evaluate a point. Cost: $\approx (2\ell)$**M**]

Vélu's formulae for computing KPS and xEVAL

- Montgomery curves can be used to efficiently compute isogenies using Vélu's formulas. Suppose we want the image of a point Q under an ℓ-isogeny ϕ, where $\ell = 2k + 1$.

Vélu's formulae for computing KPS and xEVAL

- Montgomery curves can be used to efficiently compute isogenies using Vélu's formulas. Suppose we want the image of a point Q under an ℓ-isogeny ϕ, where $\ell = 2k + 1$.
- KPS: For each $1 \leq i \leq k$ we let $(X_i : Z_i) = x([i]P)$, where $\langle P \rangle = \ker(\phi)$. Cost: $\approx 3\ell$.

Vélu's formulae for computing KPS and xEVAL

- Montgomery curves can be used to efficiently compute isogenies using Vélu's formulas. Suppose we want the image of a point Q under an ℓ-isogeny ϕ, where $\ell = 2k + 1$.
- KPS: For each $1 \leq i \leq k$ we let $(X_i : Z_i) = x([i]P)$, where $\langle P \rangle = \ker(\phi)$. Cost: $\approx 3\ell$.
- xEVAL: One can compute $(X' : Z') = x(\phi(Q))$ from $(X_Q : Z_Q) = x(Q)$ as,

$$X' = X_P \Big(\prod_{i=1}^{k} \left[(X_Q - Z_Q)(X_i + Z_i) + (Z_Q + Z_Q)(X_i - Z_i) \right] \Big)^2$$

$$Z' = Z_P \Big(\prod_{i=1}^{k} \left[(X_Q - Z_Q)(X_i + Z_i) - (Z_Q + Z_Q)(X_i - Z_i) \right] \Big)^2$$

  Cost: $\approx 2\ell$.

Vélu's formulae for computing KPS and xEVAL

- Montgomery curves can be used to efficiently compute isogenies using Vélu's formulas. Suppose we want the image of a point Q under an ℓ-isogeny ϕ, where $\ell = 2k + 1$.
- KPS: For each $1 \leq i \leq k$ we let $(X_i : Z_i) = x([i]P)$, where $\langle P \rangle = \ker(\phi)$. Cost: $\approx 3\ell$.
- xEVAL: One can compute $(X' : Z') = x(\phi(Q))$ from $(X_Q : Z_Q) = x(Q)$ as,

$$X' = X_P\left(\prod_{i=1}^{k}\left[(X_Q - Z_Q)(X_i + Z_i) + (Z_Q + Z_Q)(X_i - Z_i)\right]\right)^2$$

$$Z' = Z_P\left(\prod_{i=1}^{k}\left[(X_Q - Z_Q)(X_i + Z_i) - (Z_Q + Z_Q)(X_i - Z_i)\right]\right)^2$$

  Cost: $\approx 2\ell$.
- xISOG: Omitted.

The new $\sqrt{\text{élu}}$

Computing degree-ℓ isogenies using $\sqrt{\text{é}}$lu's formulas

- Recently, Bernstein, de Feo, Leroux and Smith presented in ANTS'2020 a new approach for computing degree-ℓ isogenies at a reduced cost of just $\tilde{O}(\sqrt{\ell})$ field operations.
- This improvement was obtained by observing that the polynomial product embedded in the isogeny computations could be speedup via a baby-step giant-step method

Computing elliptic resultants [Outline]

- Given $E_A/\mathbb{F}_q$ an order-ℓ point $P \in E_A(\mathbb{F}_q)$, and some value $\alpha \in \mathbb{F}_q$ we want to efficiently evaluate the polynomial,

$$h_S(\alpha) = \prod_{i}^{\ell-1}(\alpha - x([i]P)).$$

Computing elliptic resultants [Outline]

- Given $E_A/\mathbb{F}_q$ an order-ℓ point $P \in E_A(\mathbb{F}_q)$, and some value $\alpha \in \mathbb{F}_q$ we want to efficiently evaluate the polynomial,

$$h_S(\alpha) = \prod_{i}^{\ell-1} (\alpha - x([i]P)).$$

- From Lemma 4.3 of [BFLS ANTS'20],

$$(X - x(P+Q))(X - x(P-Q)) = X^2 + \frac{F_1(x(P), x(Q))}{F_0(x(P), x(Q))} X + \frac{F_2(x(P), x(Q))}{F_0(x(P), x(Q))}$$

where,

$$F_0(Z, X) = Z^2 - 2XZ + X^2;$$
$$F_1(Z, X) = -2(XZ^2 + (X^2 + 2AX + 1)Z + X);$$
$$F_0(Z, X) = X^2 Z^2 - 2XZ + 1.$$

Computing elliptic resultants [Outline]

- This suggests a rearrangement à la Baby-step Giant-step as,

$$h(\alpha) = \prod_{i \in \mathcal{I}} \prod_{j \in \mathcal{J}} (\alpha - x([i + s \cdot j]P))(\alpha - x([i - s \cdot j]P))$$

- Now $h(\alpha)$ can be efficiently computed by calculating the resultants of polynomials of the form,

$$h_I \leftarrow \prod_{x_i \in \mathcal{I}} (Z - x_i)) \in \mathbb{F}_q[Z]$$

$$E_J(\alpha) \leftarrow \prod_{x_j \in \mathcal{J}} \left(F_0(Z, x_j)\alpha^2 + F_1(Z, x_j)\alpha + F_2(Z, x_j)\right) \in \mathbb{F}_q[Z]$$

Our implementation of $\sqrt{\text{élu}}$

- The most demanding operations of $\sqrt{\text{élu}}$ require computing four different resultants of the form $\texttt{Res}_Z(f(Z), g(Z))$ of two polynomials $f, g \in \mathbb{F}_q[Z]$.

Our implementation of $\sqrt{\text{élu}}$

- The most demanding operations of $\sqrt{\text{élu}}$ require computing four different resultants of the form $\texttt{Res}_Z(f(Z), g(Z))$ of two polynomials $f, g \in \mathbb{F}_q[Z]$.
- Those four resultants are computed using a remainder tree approach supported by carefully tailored Karatsuba polynomial multiplications

Our implementation of $\sqrt{\text{élu}}$

- The most demanding operations of $\sqrt{\text{élu}}$ require computing four different resultants of the form $\mathtt{Res}_Z(f(Z), g(Z))$ of two polynomials $f, g \in \mathbb{F}_q[Z]$.
- Those four resultants are computed using a remainder tree approach supported by carefully tailored Karatsuba polynomial multiplications
- In practice, the computational cost of computing degree-ℓ isogenies using $\sqrt{\text{élu}}$, is close to $K(\sqrt{\ell})^{\log_2 3}$ field operations, with K a constant.

Our implementation of $\sqrt{\text{élu}}$

- The most demanding operations of $\sqrt{\text{élu}}$ require computing four different resultants of the form $\texttt{Res}_Z(f(Z), g(Z))$ of two polynomials $f, g \in \mathbb{F}_q[Z]$.
- Those four resultants are computed using a remainder tree approach supported by carefully tailored Karatsuba polynomial multiplications
- In practice, the computational cost of computing degree-ℓ isogenies using $\sqrt{\text{élu}}$, is close to $K(\sqrt{\ell})^{\log_2 3}$ field operations, with K a constant.
- $\sqrt{\text{élu}}$ is easily parallelizable. A two-core implementation can compute the four resultants in parallel, yielding an expected extra saving of around 35%.

Our implementation of $\sqrt{\text{élu}}$

- The most demanding operations of $\sqrt{\text{élu}}$ require computing four different resultants of the form $\texttt{Res}_Z(f(Z), g(Z))$ of two polynomials $f, g \in \mathbb{F}_q[Z]$.
- Those four resultants are computed using a remainder tree approach supported by carefully tailored Karatsuba polynomial multiplications
- In practice, the computational cost of computing degree-ℓ isogenies using $\sqrt{\text{élu}}$, is close to $K(\sqrt{\ell})^{\log_2 3}$ field operations, with K a constant.
- $\sqrt{\text{élu}}$ is easily parallelizable. A two-core implementation can compute the four resultants in parallel, yielding an expected extra saving of around 35%.
- Full details are available at: https://eprint.iacr.org/2020/1109.

Computing degree-ℓ isogenies using $\sqrt{\text{é}}$lu's formulas: KPS

Algorithm 1 KPS: Baby-step Giant-step method

Input: An elliptic curve $E_A/\mathbb{F}_q : y^2 = x^3 + Ax^2 + x$; $P \in E_A(\mathbb{F}_q)$ of odd prime order ℓ.

Output: $\mathcal{I} = \{x([i]P) \mid i \in I\}$, $\mathcal{J} = \{x([j]P) \mid j \in J\}$, and $\mathcal{K} = \{x([k]P) \mid k \in K\}$ such that (I, J) is an index system for S, and $K = S \backslash (I \pm J)$

1. $b \leftarrow \lfloor \sqrt{\ell - 1}/2 \rfloor$; $b' \leftarrow \lfloor (\ell - 1)/4b \rfloor$
2. $I \leftarrow \{2b(2i + 1) \mid 0 \leq i < b'\}$
3. $J \leftarrow \{2j + 1 \mid 0 \leq j < b\}$
4. $K \leftarrow S \backslash (I \pm J)$
5. $\mathcal{I} \leftarrow \{x([i]P) \mid i \in I\}$
6. $\mathcal{J} \leftarrow \{x([j]P) \mid j \in J\}$
7. $\mathcal{K} \leftarrow \{x([k]P) \mid k \in K\}$
8. **return** $\mathcal{I}, \mathcal{J}, \mathcal{K}$

- **Time cost**: $\approx 3b$ Point Additions $= 18b\mathbf{M}$
- **Memory cost**: $\leq 8b$ Field elements.

Computing degree-ℓ isogenies using $\sqrt{\text{é}}$lu: xISOG

Algorithm 2 Computing xISOG

Input: An elliptic curve $E_A/\mathbb{F}_q : y^2 = x^3 + Ax^2 + x$; $P \in E_A(\mathbb{F}_q)$ of odd prime order ℓ; $\mathcal{I}, \mathcal{J}, \mathcal{K}$ from KPS.

Output: $A' \in \mathbb{F}_q$ such that $E_{A'}/\mathbb{F}_q : y^2 = x^3 + A'x^2 + x$ is the image curve of a separable isogeny with kernel $\langle P \rangle$.

1. $h_I \leftarrow \prod_{x_i \in \mathcal{I}}(Z - x_i)) \in \mathbb{F}_q[Z]$
2. $E_{0,J} \leftarrow \prod_{x_j \in \mathcal{J}} \left(F_0(Z, x_j) + F_1(Z, x_j) + F_2(Z, x_j)\right) \in \mathbb{F}_q[Z]$
3. $E_{1,J} \leftarrow \prod_{x_j \in \mathcal{J}} \left(F_0(Z, x_j) - F_1(Z, x_j) + F_2(Z, x_j)\right) \in \mathbb{F}_q[Z]$
4. $R_0 \leftarrow \texttt{Res}_Z(h_I, E_{0,J}) \in \mathbb{F}_q$
5. $R_1 \leftarrow \texttt{Res}_Z(h_I, E_{1,J}) \in \mathbb{F}_q$
6. $M_0 \leftarrow \prod_{x_k \in \mathcal{K}}(1 - x_k) \in \mathbb{F}_q$
7. $M_1 \leftarrow \prod_{x_k \in \mathcal{K}}(-1 - x_k) \in \mathbb{F}_q$
8. $d \leftarrow \left(\frac{A-2}{A+2}\right)^{\ell} \left(\frac{M_0 R_0}{M_1 R_1}\right)^8$
9. **return** $2\left(\frac{1+d}{1-d}\right)$

- **Time cost**: $\approx \left(36b^{\log_2 3} + 4b\log_2 b + 19b + 3\log_2 b + 16\right)/2\mathbf{M}$
- **Memory cost**: $\leq 3b\log_2 b$ field elements.[shared with xEVAL]

Computing degree-ℓ isogenies using $\sqrt{\text{é}}$lu: xEVAL

Algorithm 3 Computing xEVAL

Input: An elliptic curve $E_A/\mathbb{F}_q : y^2 = x^3 + Ax^2 + x$; $P \in E_A(\mathbb{F}_q)$ of odd prime order ℓ; the x-coordinate $\alpha \neq 0$ of a point $Q \in E_A(\mathbb{F}_q)\backslash\langle P\rangle$; $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$ from KPS.

Output: The x-coordinate of $\phi(Q)$, where ϕ is a separable isogeny with kernel $\langle P\rangle$.

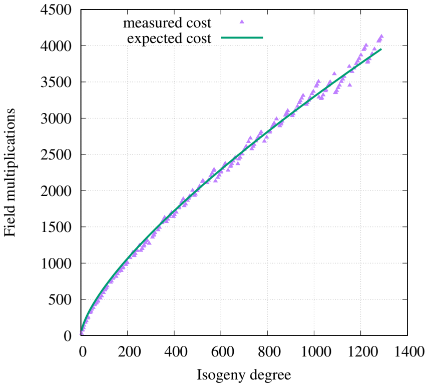
1. $h_I \leftarrow \prod_{x_i \in \mathcal{I}}(Z - x_i)) \in \mathbb{F}_q[Z]$
2. $E_{0,J} \leftarrow \prod_{x_j \in \mathcal{J}} \left(F_0(Z, x_j)/\alpha^2 + F_1(Z, x_j)/\alpha + F_2(Z, x_j)\right) \in \mathbb{F}_q[Z]$
3. $E_{1,J} \leftarrow \prod_{x_j \in \mathcal{J}} \left(F_0(Z, x_j)\alpha^2 + F_1(Z, x_j)\alpha + F_2(Z, x_j)\right) \in \mathbb{F}_q[Z]$
4. $R_0 \leftarrow \mathtt{Res}_Z(h_I, E_{0,J}) \in \mathbb{F}_q$
5. $R_1 \leftarrow \mathtt{Res}_Z(h_I, E_{1,J}) \in \mathbb{F}_q$
6. $M_0 \leftarrow \prod_{x_k \in \mathcal{K}}(1/\alpha - x_k) \in \mathbb{F}_q$
7. $M_1 \leftarrow \prod_{x_k \in \mathcal{K}}(\alpha - x_k) \in \mathbb{F}_q$
8. **return** $(M_0 R_0)^2/(M_1 R_1)^2$

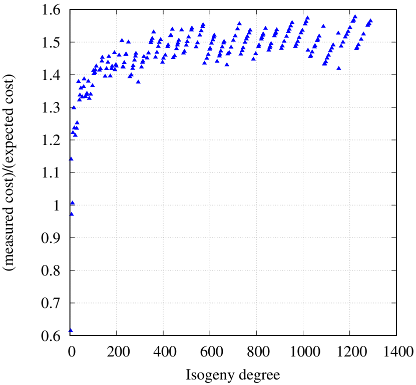
- **Time cost**: $\approx \left(36b^{\log_2 3} + 4b\log_2 b + 19b + 3\log_2 b + 16\right)/2\mathbf{M}$
- **Memory cost**: $\leq 3b\log_2 b$ field elements.[shared with xISOG]

Improving the computation of √élu

0. For Steps 2-3 of `xEVAL`: Compute $E_{0,J}$ and from it, obtain directly $E_{1,J}$
1. For `xEVAL`: Compute $E_{0,J}$ by expressing $\left(F_0(Z, x_j)z^2 + F_1(Z, x_j)xz + F_2(Z, x_j)x^2\right)$ as a quadratic polynomial $aZ^2 + bZ + c$. This formulation saves 3 field multiplications per polynomial $E_{0,j}$, $0 \leq j < \#\mathcal{J}$ as compared with a more direct approach.
2. For multi-core environments: compute the two resultants of `xEVAL` and the two resultants of `xISOG` in parallel

Cost model for computing degree-ℓ isogenies using $\sqrt{\text{é}}$lu





Computing a degree-ℓ isogeny. Let $b = \frac{\sqrt{\ell - 1}}{2}$.

$$\text{Expected Cost}(b) = 4\left(9b^{\log_2(3)}\left(1 - 2\left(\frac{2}{3}\right)^{\log_2(b)+1}\right) + 2b\log_2(b)\right)$$
$$+ 3\left(\left(1 - \frac{1}{3^{\log_2(b)+1}}\right)b^{\log_2(3)}\right) + 37b + 3\log_2(b) + 16$$
$$\approx 39 \cdot b^{\log_2(3)}$$

This approximation is a lower bound

Karatsuba Vs. Schönage-FFT polynomial multiplication

- Is it accurate that the asymptotic cost of $\sqrt{}$élu is $\tilde{O}(\sqrt{\ell})$ as claimed by [BFLS ANTS'20]?

Karatsuba Vs. Schönage-FFT polynomial multiplication

- Is it accurate that the asymptotic cost of $\sqrt{\text{é}}\text{lu}$ is $\tilde{O}(\sqrt{\ell})$ as claimed by [BFLS ANTS'20]?
  Yes. Using variants of the FFT multiplication along with Schönage's method for polynomial multiplication, the asymptotic cost of $\sqrt{\text{é}}\text{lu}$ is indeed $\tilde{O}(\sqrt{\ell})$.

Karatsuba Vs. Schönage-FFT polynomial multiplication

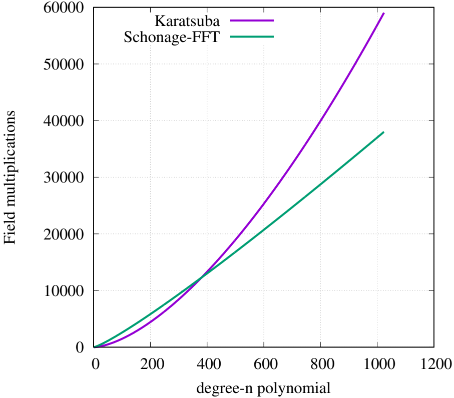
- But then why is better in practice to use Karatsuba polynomial multplication for computing $\sqrt{\text{élu}}$ is $\tilde{O}(\sqrt{\ell})$?

Karatsuba Vs. Schönage-FFT polynomial multiplication

- But then why is better in practice to use Karatsuba polynomial multplication for computing $\sqrt{}$élu is $\tilde{O}(\sqrt{\ell})$?
  Due to the hidden constants in the Schönage-FFT polynomial multiplication, Karatsuba is a more economical approach for polynomials of degree less than ≈ 300. Notice that it is always possible to combine these two approaches.

Karatsuba Vs. Schönage-FFT polynomial multiplication

- But then why is better in practice to use Karatsuba polynomial multplication for computing $\sqrt{\text{élu}}$ is $\tilde{O}(\sqrt{\ell})$?



Constant-time $\sqrt{\text{élu}}$

- No branches with secret conditions.
- $\sqrt{\text{élu}}$ is a multiplicative-inverse-free procedure
- The three procedures KPS, xISOG and xEVAL are completely deterministic.
- The size of the sets $\mathcal{I}, \mathcal{J}$ and $\mathcal{K}$ as defined in KPS, are a function of the [public] parameter ℓ.
- All the polynomial coefficients involved in the $\sqrt{\text{élu}}$ computation are different than zero. Hence, independently of the order-ℓ point P, the cost of the primitives KPS, xISOG and xEVAL is always the same.
- The remainder tree, which is at the heart of the two resultant computations, takes the same cost for either xISOG or xEVAL.
- We have ran a few number of examples changing the point P, and the computational costs of KPS, xISOG and xEVAL, remain the same.

$\sqrt{\text{é}}$lu memory cost

- Memory analysis for Vélu [direct approach at a computational cost of $\approx 6\ell$]:
  - KPS requires to compute and store $\ell - 1$ field elements.
- Memory analysis for $\sqrt{\text{é}}$lu:
  - Less than $4b$ points, equivalent to $8b$ field elements, are computed and stored in KPS.
  - The computation of the trees determined by the polynomial h_I in Step 1 of xISOG and xEVAL, requires the storage of no more than $3b \log_2 b$ field elements.
  - Hence, $\sqrt{\text{é}}$lu's memory cost is of about $8b + 3b \log_2 b$ field elements.

Conclusion: For any odd degree-ℓ, $\sqrt{\text{é}}$lu always requires less memory storage than traditional Vélu's formulae.

Two isogeny-based protocols

$\sqrt{\text{élu}}$ impact on isogeny-based protocols

- What is the expected impact of $\sqrt{\text{élu}}$ for SIDH or SIKE?

$\sqrt{\text{é}}$lu impact on isogeny-based protocols

- What is the expected impact of $\sqrt{\text{é}}$lu for SIDH or SIKE?
  None

$\sqrt{\text{é}}$lu impact on isogeny-based protocols

- What is the expected impact of $\sqrt{\text{é}}$lu for CSIDH?

$\sqrt{\text{é}}$lu impact on isogeny-based protocols

- What is the expected impact of $\sqrt{\text{é}}$lu for CSIDH?
  Promising. As of yet, we have only seen a moderate acceleration for the CSIDH-512 and CSIDH-1024 instantiations, but for CSIDH-1792 the savings are impressive.

$\sqrt{\text{é}}$lu impact on isogeny-based protocols

- What is the expected impact of $\sqrt{\text{é}}$lu for B-SIDH?

$\sqrt{\text{é}}$lu impact on isogeny-based protocols

- What is the expected impact of $\sqrt{\text{é}}$lu for B-SIDH?
  Huge. B-SIDH is the big winner among the isogeny-based protocols

Overviewing the CSIDH [Castryck-Lange-Martindale-Panny-Renes Asiacrypt'18]

Public parameter:
$E/\mathbb{F}_p\colon By^2 = x^3 + Ax^2 + x,$

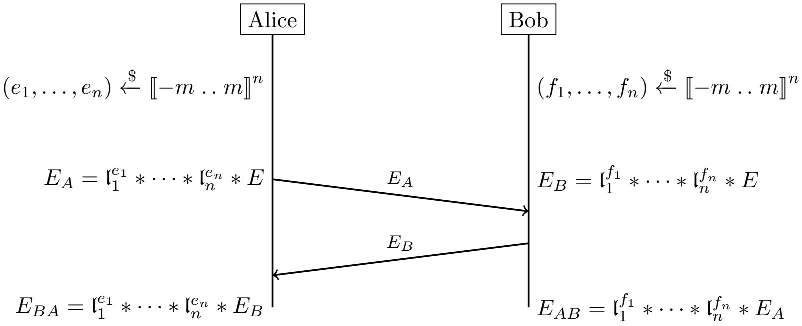


Figure: CSIDH key-exchange protocol

CSIDH works over a finite field $\mathbb{F}_p$, where p is a prime of the form

$$p := 4\prod_{i=1}^{n} \ell_i - 1$$

Overviewing the CSIDH [Castryck-Lange-Martindale-Panny-Renes Asiacrypt'18]

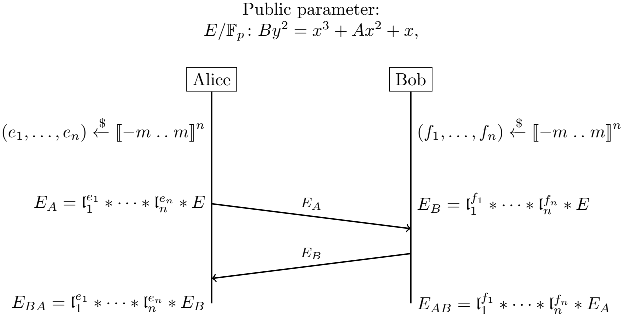


Figure: CSIDH key-exchange protocol

$(p+1)/4 = 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 47 \cdot 53 \cdot 59 \cdot 61 \cdot 67 \cdot 71 \cdot 73 \cdot 79 \cdot 83 \cdot \mathbf{89} \cdot 97 \cdot 101 \cdot 103 \cdot$
$107 \cdot 109 \cdot 113 \cdot 127 \cdot 131 \cdot 137 \cdot 139 \cdot 149 \cdot 151 \cdot 157 \cdot 163 \cdot 167 \cdot 173 \cdot 179 \cdot 181 \cdot 191 \cdot 193 \cdot 197 \cdot 199 \cdot 211 \cdot 223 \cdot$
$227 \cdot 229 \cdot 233 \cdot 239 \cdot 241 \cdot 251 \cdot 257 \cdot 263 \cdot 269 \cdot 271 \cdot 277 \cdot 281 \cdot 283 \cdot 293 \cdot 307 \cdot 311 \cdot 313 \cdot 317 \cdot 331 \cdot 337 \cdot 347 \cdot$
$349 \cdot 353 \cdot 359 \cdot 367 \cdot 373 \cdot \mathbf{587}$

Playing the B-SIDH [Costello Asiacrypt'20]

Public parameter:

$E/\mathbb{F}_{p^2}\colon By^2 = x^3 + Ax^2 + x,$

$P_a, Q_a \in E[p+1]$ of order M, and $P_b, Q_b \in E[p-1]$ of order N

Alice		Bob
$sk_a \xleftarrow{\$} [\![0 \mathinner{..} M-1]\!]$		$sk_b \xleftarrow{\$} [\![0 \mathinner{..} N-1]\!]$
$R_a = P_a + [sk_a]Q_a$		$R_b = P_b + [sk_b]Q_b$
$\phi_a\colon E \to E/\langle R_a \rangle$	$E_a, \phi_a(P_b), \phi_a(Q_b)$ →	$\phi_b\colon E \to E/\langle R_b \rangle$
$E_a = E/\langle R_a \rangle$		$E_b = E/\langle R_b \rangle$
	← $E_b, \phi_b(P_a), \phi_b(Q_a)$	
$E_{ab} = E_b/\langle \phi_b(R_a) \rangle$		$E_{ab} = E_a/\langle \phi_a(R_b) \rangle$

Figure: B-SIDH protocol for a prime p such that $M|(p+1)$ and $N|(p-1)$.

Alice and Bob work in the $(p+1)$- and $(p-1)$-torsion of a set of supersingular curves defined over $\mathbb{F}_{p^2}$ and the set of their quadratic twist, respectively.

Playing the B-SIDH [Costello Asiacrypt'20]

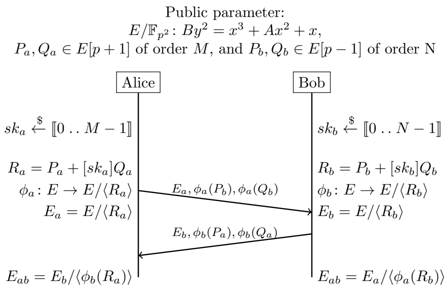


Figure: B-SIDH protocol for a prime p such that $M|(p+1)$ and $N|(p-1)$.

Prime example: **B-SIDHp237**:

$p =$ 0x1B40F93CE52A207249237A4FF37425A798E914A74949FA343E8EA487FFFF

$$M = 4^3 \cdot \left(4 \cdot 3^4 \cdot 17 \cdot 19 \cdot 31 \cdot 37 \cdot 53^2\right)^6,$$

$$N = 7 \cdot 13 \cdot 43 \cdot 73 \cdot 103 \cdot 269 \cdot 439 \cdot 881 \cdot 883 \cdot 1321 \cdot 5479 \cdot 9181 \cdot 12541 \cdot 15803 \cdot 20161 \cdot 24043 \cdot 34843 \cdot 48437 \cdot 62753 \cdot 72577.$$

Playing the B-SIDH [Costello Asiacrypt'20]

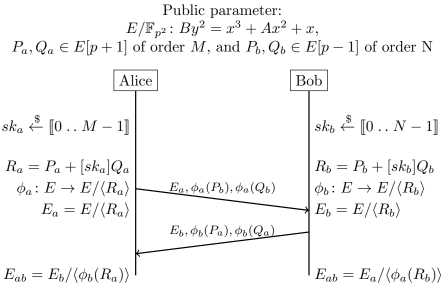


Figure: B-SIDH protocol for a prime p such that $M|(p+1)$ and $N|(p-1)$.

Prime example: **B-SIDHp253**:

$$p = \texttt{0x1935BECE108DC6C0AAD0712181BB1A414E6A8AAA6B510FC29826190FE7EDA80F},$$

$$M = 4^2 \cdot 3 \cdot 7^{16} \cdot 17^9 \cdot 31^8 \cdot 311 \cdot 571 \cdot 1321 \cdot 5119 \cdot 6011 \cdot 14207 \cdot 28477 \cdot 76667,$$

$$N = 11^{18} \cdot 19 \cdot 23^{13} \cdot 47 \cdot 79 \cdot 83 \cdot 89 \cdot 151 \cdot 3347 \cdot 17449 \cdot 33461 \cdot 51193.$$

Experiments and efficiency

Number of base field operations for constant-time CSIDH-512

Configuration	Group action evaluation	M	S	a	Cost	Saving (%)
tvelu	OAYT-style	0.641	0.172	0.610	0.813	—
	MCR-style	0.835	0.231	0.785	1.066	
	dummy-free	1.246	0.323	1.161	1.569	
svelu	OAYT-style	0.656	0.178	0.988	0.834	−2.583
	MCR-style	0.852	0.219	1.295	1.071	−0.469
	dummy-free	1.257	0.324	1.888	1.581	−0.765
hvelu	OAYT-style	0.624	0.165	0.893	0.789	2.952
	MCR-style	0.805	0.204	1.164	1.009	5.347
	dummy-free	1.198	0.301	1.696	1.499	4.461

Table: Number of field operation for the constant-time CSIDH-512 group action evaluation. Counts are given in millions of operations, averaged over 1024 random experiments. For computing the Cost column, it is assumed that $\mathbf{M} = \mathbf{S}$ and all addition counts are ignored. Last column labeled **Saving** corresponds to $\left(1 - \frac{\textbf{Cost}}{\text{baseline}}\right) \times 100$ and baseline equals to *tvelu* configuration.

Number of base field operations for constant-time CSIDH-1024

Configuration	Group action evaluation	M	S	a	Cost	Saving (%)
tvelu	OAYT-style	0.630	0.152	0.576	0.782	—
	MCR-style	0.775	0.190	0.695	0.965	
	dummy-free	1.152	0.259	1.012	1.411	
svelu	OAYT-style	0.566	0.138	0.963	0.704	9.974
	MCR-style	0.702	0.152	1.191	0.854	11.503
	dummy-free	1.046	0.230	1.746	1.276	9.568
hvelu	OAYT-style	0.552	0.133	0.924	0.685	12.404
	MCR-style	0.687	0.146	1.148	0.833	13.679
	dummy-free	1.027	0.221	1.679	1.248	11.552

Table: Number of field operation for the constant-time CSIDH-1024 group action evaluation. Counts are given in millions of operations, averaged over 1024 random experiments. For computing the Cost column, it is assumed that $\mathbf{M} = \mathbf{S}$ and all addition counts are ignored. Last column labeled **Saving** corresponds to $\left(1 - \frac{\textbf{Cost}}{\text{baseline}}\right) \times 100$ and baseline equals to *tvelu* configuration.

Number of base field operations for constant-time CSIDH-1792

Configuration	Group action evaluation	M	S	a	Cost	Saving (%)
tvelu	OAYT-style	1.385	0.263	1.137	1.648	—
	MCR-style	1.041	0.239	0.911	1.280	
	dummy-free	1.557	0.327	1.336	1.884	
svelu	OAYT-style	1.063	0.187	2.073	1.250	24.150
	MCR-style	0.807	0.154	1.550	0.961	24.922
	dummy-free	1.233	0.247	2.314	1.480	21.444
hvelu	OAYT-style	1.060	0.185	2.061	1.245	24.454
	MCR-style	0.797	0.151	1.522	0.948	25.938
	dummy-free	1.220	0.241	2.272	1.461	22.452

Table: Number of field operation for the constant-time CSIDH-1792 group action evaluation. Counts are given in millions of operations, averaged over 1024 random experiments. For computing the Cost column, it is assumed that **M** = **S** and all addition counts are ignored. Last column labeled **Saving** corresponds to $\left(1 - \frac{\textbf{Cost}}{\text{baseline}}\right) \times 100$ and baseline equals to *tvelu* configuration.

Number of base field operations for the secret sharing phase of BSIDH

Configuration		Alice's side			Bob's side		
		M	a	Saving (%)	M	a	Saving (%)
tvelu	**B-SIDHp253**	1.831	3.936	—	1.529	3.277	—
	B-SIDHp255	1.931	4.127		1.305	2.795	
	B-SIDHp247	0.434	0.928		1.113	2.372	
	B-SIDHp237	0.053	0.115		4.872	10.377	
	B-SIDHp257	1.963	4.190		0.156	0.336	
$\sqrt{\text{é}}$lu	**B-SIDHp253**	0.462	1.741	74.768	0.390	1.517	74.493
	B-SIDHp255	0.505	1.943	73.847	0.362	1.338	72.261
	B-SIDHp247	0.203	0.653	53.226	0.449	1.541	59.659
	B-SIDHp237	0.053	0.115	00.000	1.183	4.585	75.718
	B-SIDHp257	0.555	2.077	71.727	0.108	0.306	30.769

Table: Number of base field operations for the secret sharing phase of BSIDH. Counts are given in millions of operations. Columns labeled **Saving** correspond to $\left(1 - \frac{\textbf{Cost}}{\text{baseline}}\right) \times 100$ and baseline equals to *tvelu* configuration.

Skylake Clock cycle timings for several key exchange isogeny-based protocols

Implementation	Protocol Instantiation	Mcycles
SIKE [NIST alternative candidate]	SIKEp434	22
Castryck *et al.* [Original CSIDH]	CSIDH-512 unprotected	4×155
Bernstein *et al.* [Original $\sqrt{\text{é}}$lu]	CSIDH-512 unprotected	4×153
	CSIDH-1024 unprotected	4×760
Cervantes-Vázquez *et al.* [LC'19 CSIDH imp]	CSIDH-512	4×238
Chi-Domínguez *et al.* [CSIDH with strategies]	CSIDH-512	4×230
Hutchinson *et al.* [CSIDH with strategies]	CSIDH-512	4×229
This work (**estimated**)	CSIDH-512	4×223
	B-SIDH-p253	119

Table: Skylake Clock cycle timings for a key exchange protocol for different instantiations of the SIDH, CSIDH, and B-SIDH protocols.

Open problems

- What would be the most attractive projects on $\sqrt{\text{é}}$lu related topics for an algorithmic oriented fellow?

Open problems

- What would be the most attractive projects on $\sqrt{\text{élu}}$ related topics for an algorithmic oriented fellow?
  - To tune-up the sets $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$ of KPS to hopefully obtained a reduced cost on $\sqrt{\text{élu}}$
  - To study more efficient ways to perform the [scaled] remainder tree associated to the computation of the resultants (See Bernstein's "Fast multiplications and its applications")
  - To study other polynomial multiplication methods (such as Toom-Cook)

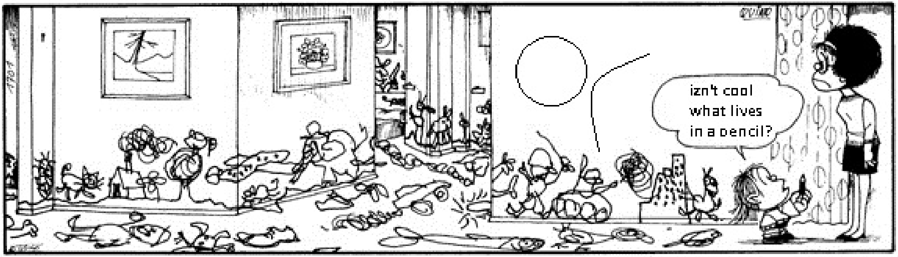
Open problems

- What would be the most attractive projects on $\sqrt{\text{élu}}$ related topics for a programming oriented fellow?

Open problems

- What would be the most attractive projects on $\sqrt{\text{é}}$lu related topics for a programming oriented fellow?
  - A C-code highly optimized implementation of B-SIDH and/or CSIDH using $\sqrt{\text{é}}$lu
  - A multi-core implementation of B-SIDH and/or CSIDH using $\sqrt{\text{é}}$lu
  - A hardware-software co-design implementation of B-SIDH and/or CSIDH using $\sqrt{\text{é}}$lu

Thanks



How to compute the resultants using the Schönage-FFT polynomial multiplication

- Let A be commutative ring where 2 invertible. For $n > 1$ a power of 2, c a square in A and $\zeta \in A$ a square of -1, let f, g be two polynomials in $A[x]/(x^n + c)$.
- To multiply f and g, one can split the problem into two smaller ones by reducing f, g to $f_-, g_- \in A[x]/(x^{n/2} - \zeta c^{1/2})$ and to $f_+, g_+ \in A[x]/(x^{n/2} + \zeta c^{1/2})$g.
- Then, the products $f_- g_-$, $f_+ g_+$ are computed, and subsequently embedded into $A[x]/(x^n + c)$ wherein $(f_- g_- + f_+ g_+)$ and $(f_- g_- - f_+ g_+)$ are calculated to finally recover $2fg$.
- Note that when c is an nth root in A, which in addition contains an nth root of -1, then the above procedure can be applied recursively to compute the product nfg at a cost of k multiplications in A and $\frac{3}{2} n \log_2(n)$ easy multiplications in A by constants.