SeaSign: Compact isogeny signatures from class group actions

Luca De Feo[1], Steven D. Galbraith[2]

[1]Université Paris Saclay – UVSQ, France
[2]University of Auckland, New Zeland

May 23, 2019, Eurocrypt, Darmstadt

Slides online at `https://defeo.lu/docet`

Post-quantum isogeny primitives

SIDH (Jao, De Feo 2011)

- Pronounce S–I–D–H;
- Based on random isogeny walks in the full supersingular graph over $\mathbb{F}_{p^2}$;
- Basis for the NIST KEM candidate SIKE;
- Better asymptotic quantum security;
- Short keys, slow.

CSIDH (Couveignes 1996; Rostovtsev Stolbunov 2006; Castryck, Lange, Martindale, Panny, Renes 2018)

- Pronounce Sea–Side;
- Based on random isogeny walks in the $\mathbb{F}_p$-restricted supersingular isogeny graph;
- Straightforward generalization of Diffie–Hellman;
- More “natural” security assumption;
- Shorter keys, slower.

Post-quantum isogeny primitives

SIDH (Jao, De Feo 2011)

- Pronounce S–I–D–H;
- Based on random isogeny walks in the full supersingular graph over $\mathbb{F}_{p^2}$;
- Basis for the NIST KEM candidate SIKE;
- Better asymptotic quantum security;
- Short keys, slow.
- Crappy signatures (slow, large). Not this talk.

CSIDH (Couveignes 1996; Rostovtsev Stolbunov 2006; Castryck, Lange, Martindale, Panny, Renes 2018)

- Pronounce Sea–Side;
- Based on random isogeny walks in the $\mathbb{F}_p$-restricted supersingular isogeny graph;
- Straightforward generalization of Diffie–Hellman;
- More “natural” security assumption;
- Shorter keys, slower.

Post-quantum isogeny primitives

SIDH (Jao, De Feo 2011)

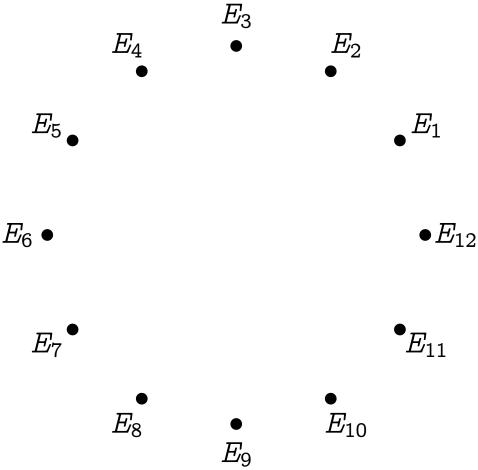
- Pronounce S–I–D–H;
- Based on random isogeny walks in the full supersingular graph over $\mathbb{F}_{p^2}$;
- Basis for the NIST KEM candidate SIKE;
- Better asymptotic quantum security;
- Short keys, slow.
- Crappy signatures (slow, large).
  Not this talk.

CSIDH (Couveignes 1996; Rostovtsev Stolbunov 2006; Castryck, Lange, Martindale, Panny, Renes 2018)

- Pronounce Sea–Side;
- Based on random isogeny walks in the $\mathbb{F}_p$-restricted supersingular isogeny graph;
- Straightforward generalization of Diffie–Hellman;
- More “natural” security assumption;
- Shorter keys, slower.
- Also crappy signatures, but different!
  This talk.

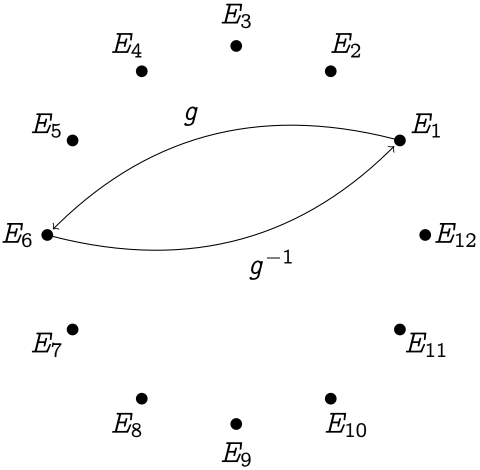
What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;



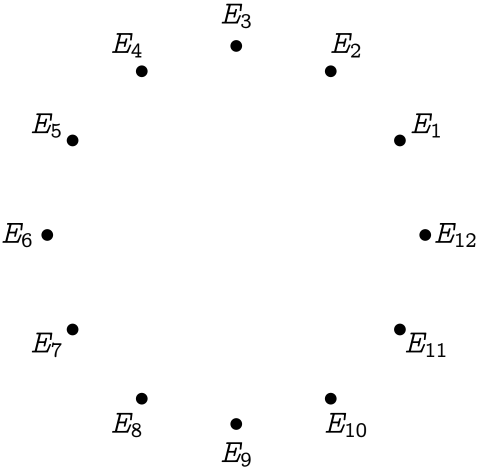
What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;



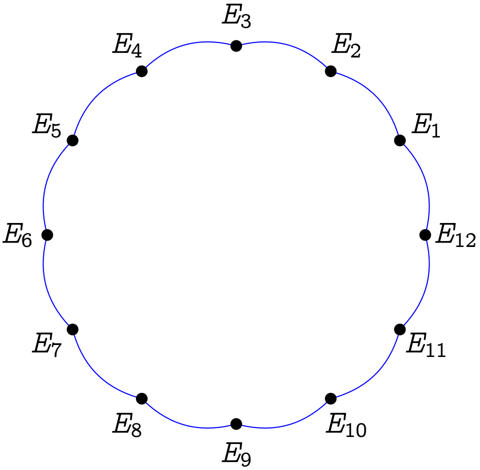
What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:



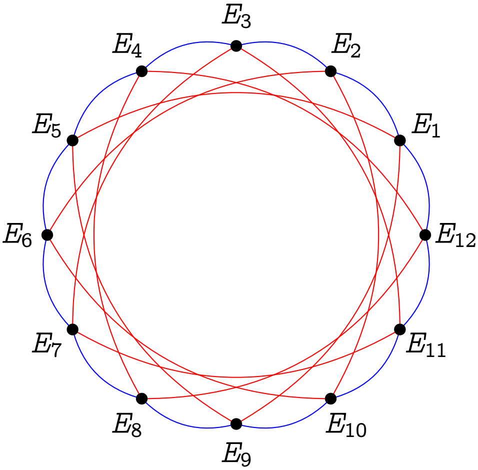
What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2,



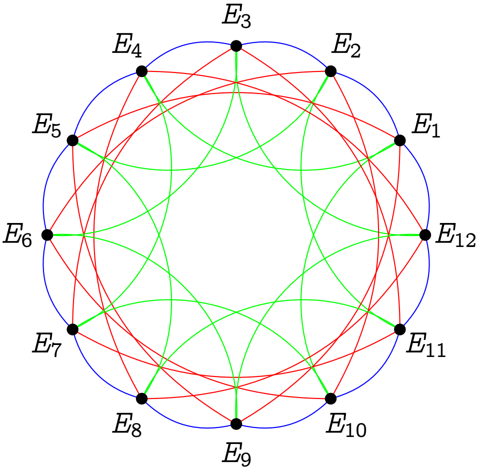
What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3,



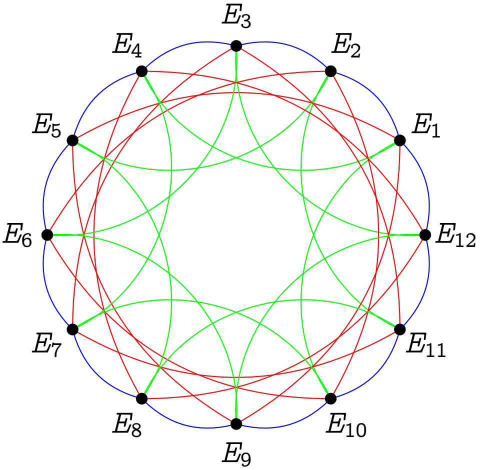
What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, …



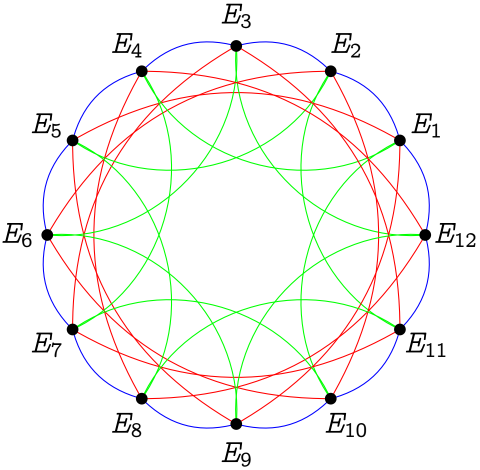
What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, …
- Graph structure isomorphic to a Cayley graph;



What is CSIDH?

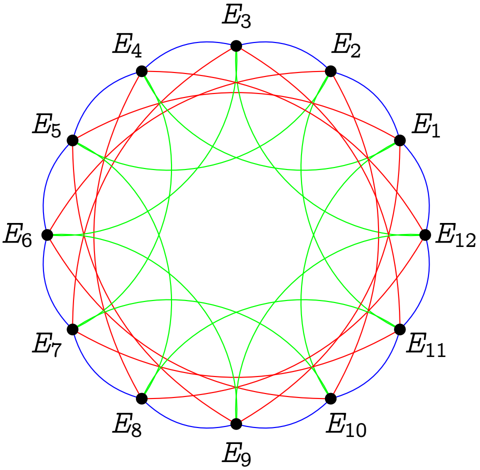
- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, …
- Graph structure isomorphic to a Cayley graph;
- Good algorithm to do random walks in the graph.



What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, …
- Graph structure isomorphic to a Cayley graph;
- Good algorithm to do random walks in the graph.

Key exchange:

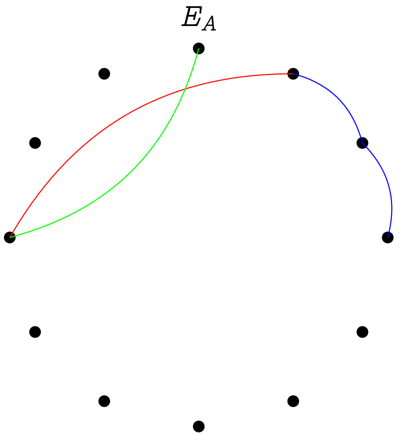


What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, …
- Graph structure isomorphic to a Cayley graph;
- Good algorithm to do random walks in the graph.

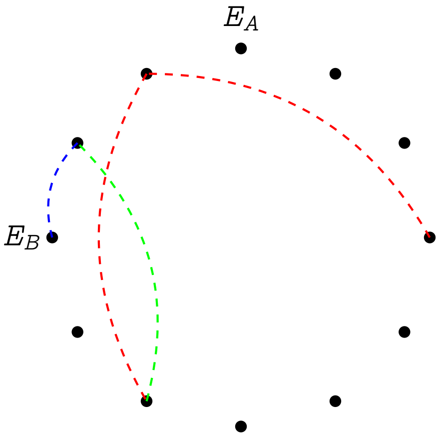
Key exchange:

- Alice picks secret $a = g_2^{a_2} g_3^{a_3} g_5^{a_5} \cdots$,



What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, …
- Graph structure isomorphic to a Cayley graph;
- Good algorithm to do random walks in the graph.

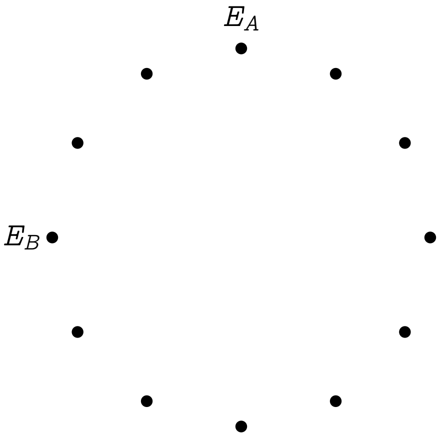


Key exchange:

- Alice picks secret $a = g_2^{a_2} g_3^{a_3} g_5^{a_5} \cdots$,
- Bob picks secret $b = g_2^{b_2} g_3^{b_3} g_5^{b_5} \cdots$,

What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, ...
- Graph structure isomorphic to a Cayley graph;
- Good algorithm to do random walks in the graph.

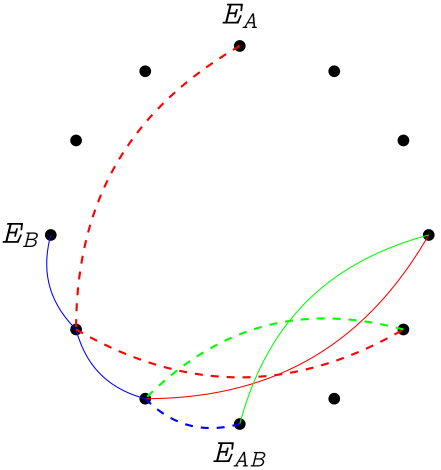


Key exchange:

- Alice picks secret $a = g_2^{a_2} g_3^{a_3} g_5^{a_5} \cdots$,
- Bob picks secret $b = g_2^{b_2} g_3^{b_3} g_5^{b_5} \cdots$,
- They exchange $E_A = a * E_1$ and $E_B = b * E_1$,

What is CSIDH?

- A set of supersingular elliptic curves over $\mathbb{F}_p$;
- A group action by an abelian group G;
- Only efficient to evaluate the action of some small degree generators $g \in G$, e.g.:
  - degree 2, degree 3, degree 5, …
- Graph structure isomorphic to a Cayley graph;
- Good algorithm to do random walks in the graph.



Key exchange:

- Alice picks secret $a = g_2^{a_2} g_3^{a_3} g_5^{a_5} \cdots$,
- Bob picks secret $b = g_2^{b_2} g_3^{b_3} g_5^{b_5} \cdots$,
- They exchange $E_A = a * E_1$ and $E_B = b * E_1$,
- Shared secret is
  $E_{AB} = (ab) * E_1 = a * E_B = b * E_A$.

A Σ-protocol from Diffie–Hellman[1]

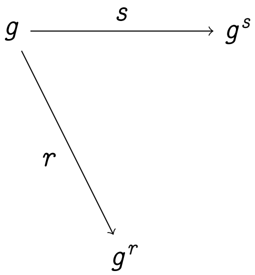
- A key pair (s, g^s);

$$g \xrightarrow{\quad s \quad} g^s$$

[1] Kids, do not try this at home! Use Schnorr!

A Σ-protocol from Diffie–Hellman[1]

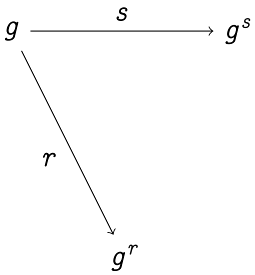
- A key pair (s, g^s);
- Commit to a random element g^r;



[1] Kids, do not try this at home! Use Schnorr!

A Σ-protocol from Diffie–Hellman[1]

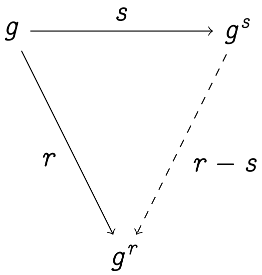
- A key pair (s, g^s);
- Commit to a random element g^r;
- Challenge with bit $b \in \{0, 1\}$;



[1] Kids, do not try this at home! Use Schnorr!

A Σ-protocol from Diffie–Hellman[1]

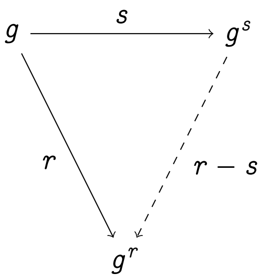
- A key pair (s, g^s);
- Commit to a random element g^r;
- Challenge with bit $b \in \{0, 1\}$;
- Respond with $c = r - b \cdot s \mod \#G$;



[1] Kids, do not try this at home! Use Schnorr!

A Σ-protocol from Diffie–Hellman[1]

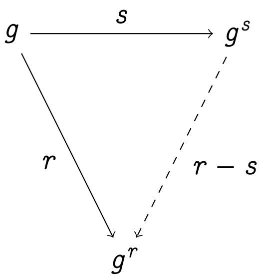
- A key pair (s, g^s);
- Commit to a random element g^r;
- Challenge with bit $b \in \{0, 1\}$;
- Respond with $c = r - b \cdot s \mod \#G$;
- Verify that $g^c(g^s)^b = g^r$.



[1] Kids, do not try this at home! Use Schnorr!

A Σ-protocol from Diffie–Hellman[1]

- A key pair (s, g^s);
- Commit to a random element g^r;
- Challenge with bit $b \in \{0, 1\}$;
- Respond with $c = r - b \cdot s \mod \#G$;
- Verify that $g^c(g^s)^b = g^r$.



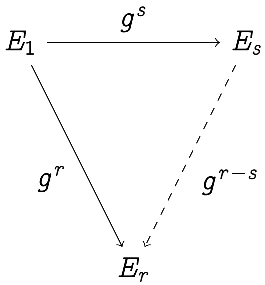
Zero-knowledge

Does not leak because:
c is uniformly distributed and independent from s.

[1] Kids, do not try this at home! Use Schnorr!

A Σ-protocol from Diffie–Hellman[1]

- A key pair (s, g^s);
- Commit to a random element g^r;
- Challenge with bit $b \in \{0, 1\}$;
- Respond with $c = r - b \cdot s \mod \#G$;
- Verify that $g^c (g^s)^b = g^r$.



Zero-knowledge

Does not leak because:
c is uniformly distributed and independent from s.

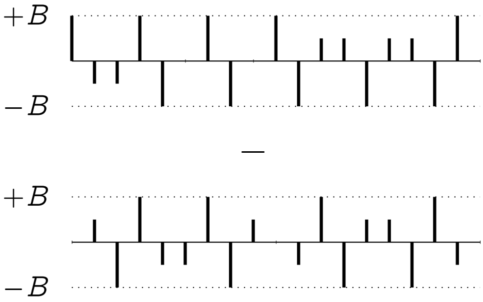
Unlike Schnorr, compatible with group action Diffie–Hellman.

[1] Kids, do not try this at home! Use Schnorr!

The trouble with groups of unknown structure

In CSIDH secrets look like: $g^{\vec{s}} = g_2^{s_2} g_3^{s_3} g_5^{s_5} \cdots$

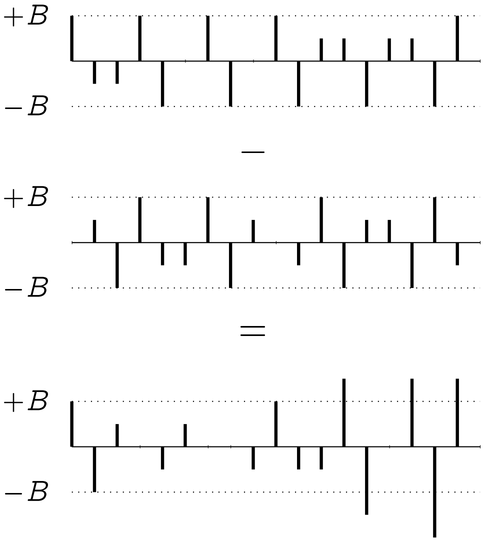
- the elements g_i are fixed,
- the secret is the exponent vector $\vec{s} = (s_2, s_3, \ldots) \in [-B, B]^n$,
- secrets must be sampled in a box $[-B, B]^n$ "large enough"…



The trouble with groups of unknown structure

In CSIDH secrets look like: $g^{\vec{s}} = g_2^{s_2} g_3^{s_3} g_5^{s_5} \cdots$

- the elements g_i are fixed,
- the secret is the exponent vector $\vec{s} = (s_2, s_3, \dots) \in [-B, B]^n$,
- secrets must be sampled in a box $[-B, B]^n$ "large enough"…



The leakage

With $\vec{s}, \vec{r} \xleftarrow{\$} [-B, B]^n$, the distribution of $\vec{r} - \vec{s}$ depends on the long term secret $\vec{s}$!

The two fixes

Compute the group structure and stop whining

- Already suggested by Couveignes (1996) and Rostovtsev–Stolbunov (2006).
- Computationally intensive (subexponential parameter generation).
- Technically not post-quantum (rather, post-post-quantum).
- Done last week by Beullens, Kleinjung and Vercauteren: CSI-FiSh (eprint:2019/498).
- Decent parameters, e.g.: 263 bytes, 390 ms, @NIST-1.
- Not this work.

The two fixes

Compute the group structure and stop whining

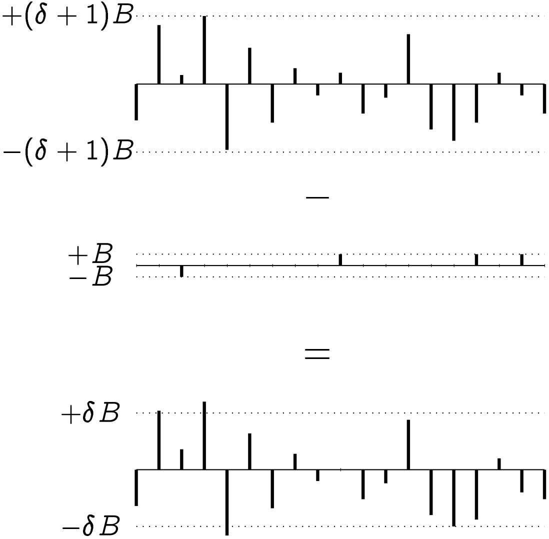
- Already suggested by Couveignes (1996) and Rostovtsev–Stolbunov (2006).
- Computationally intensive (subexponential parameter generation).
- Technically not post-quantum (rather, post-post-quantum).
- Done last week by Beullens, Kleinjung and Vercauteren: CSI-FiSh (eprint:2019/498).
- Decent parameters, e.g.: 263 bytes, 390 ms, @NIST-1.
- Not this work.

Do like the lattice people

- Use Fiat–Shamir with aborts (Lyubashevsky 2009).
- Huge increase in signature size and time.
- Compromise signature size/time with public key size.
- This work.

Rejection sampling

- Sample long term secret $\vec{s}$ in the usual box $[-B, B]^n$,
- Sample ephemeral $\vec{r}$ in a larger box $[-(\delta+1)B, (\delta+1)B]^n$,
- Throw away $\vec{r} - \vec{s}$ if it is out of the box $[-\delta B, \delta B]^n$.



Zero-knowledge

Theorem: $\vec{r} - \vec{s}$ is uniformly distributed in $[-\delta B, \delta B]^n$.

Problem: set δ so that rejection probability is low.

Performance

- For λ-bit security, protocol must be repeated λ times in parallel;
- $\delta = \lambda n$ for a rejection probability $\leq 1/3$;
- Signature size $\approx \lambda n$ coefficients $\in [-\delta B, \delta B]$;
- Sign/verify time linear in $\|\vec{r} - \vec{s}\|_\infty \approx \lambda^2 n^2 B$.

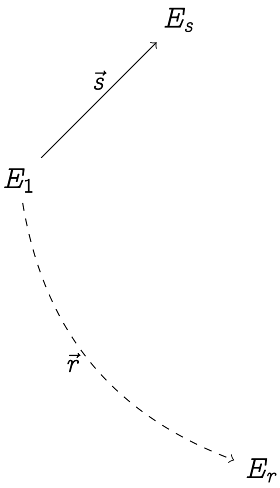
CSIDH instantiation (NIST-1)

Parameters:	$\lambda = 128$, $n = 74$, $B = 5$;
PK size:	64 B
SK size:	32 B
Signature:	20 KiB
Verify time:	10 hours
Sign time:	$3\times$ verify

Key/signature size compromise

- One key pair $(\vec{s}, E_s)$;
- Challenge $b \in \{0, 1\}$;
- Reveal $\vec{r} - b\vec{s}$;

$\rightarrow$ λ iterations;



Key/signature size compromise

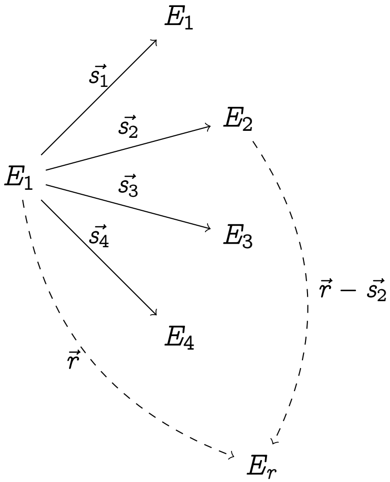
- One key pair $(\vec{s}, E_s)$;
- Challenge $b \in \{0, 1\}$;
- Reveal $\vec{r} - b\vec{s}$;

$\rightarrow$ λ iterations;

Compromise: $\mathbf{t}$-bit challenges

- $\mathbf{2^t}$ key pairs $(\vec{s_i}, E_i)$;
- Challenge $b \in \{0, 2^t\}$;
- Reveal $\vec{r} - \vec{s_b}$;

$\rightarrow$ $\lambda/\mathbf{t}$ iterations;



Key/signature size compromise

- One key pair $(\vec{s}, E_s)$;
- Challenge $b \in \{0, 1\}$;
- Reveal $\vec{r} - b\vec{s}$;

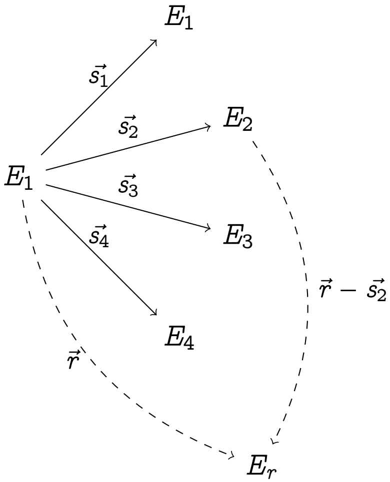
→ λ iterations;

→ Sample $r \xleftarrow{\$} [-\lambda nB, \lambda nB]$.

Compromise: t-bit challenges

- 2^{t} key pairs $(\vec{s_i}, E_i)$;
- Challenge $b \in \{0, 2^t\}$;
- Reveal $\vec{r} - \vec{s_b}$;

→ λ/t iterations;



Key/signature size compromise

- One key pair $(\vec{s}, E_s)$;
- Challenge $b \in \{0, 1\}$;
- Reveal $\vec{r} - b\vec{s}$;

$\rightarrow$ λ iterations;

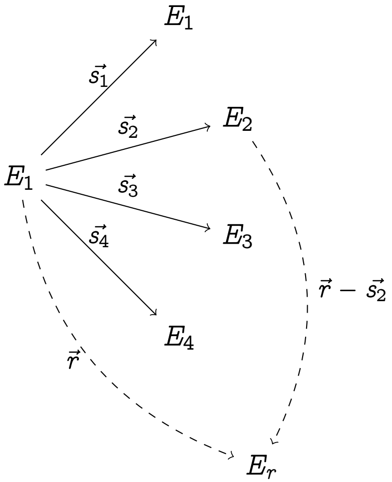
$\rightarrow$ Sample $r \xleftarrow{\$} [-\lambda nB, \lambda nB]$.

Compromise: t-bit challenges

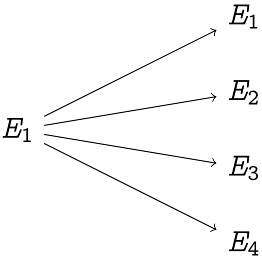
- $\mathbf{2^t}$ key pairs $(\vec{s_i}, E_i)$;
- Challenge $b \in \{0, 2^t\}$;
- Reveal $\vec{r} - \vec{s_b}$;

$\rightarrow$ $\lambda/\mathbf{t}$ iterations;

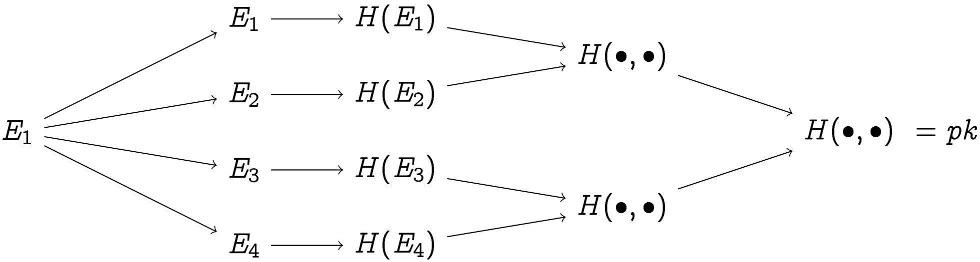
$\rightarrow$ Sample $r \xleftarrow{\$} [-\lambda nB/\mathbf{t}, \lambda nB/\mathbf{t}]$.



Public key compression

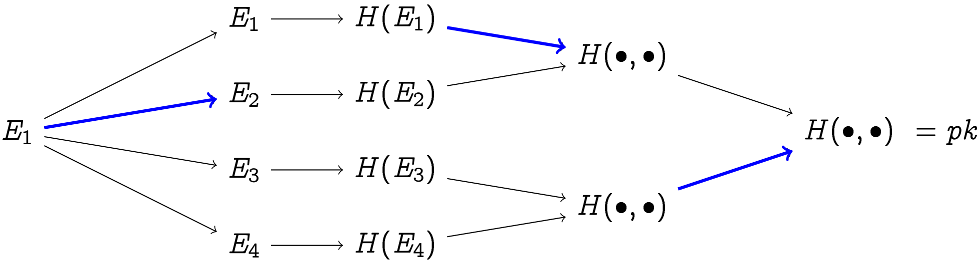


Public key compression



- Construct Merkle tree on top of public keys, root is the new public key;

Public key compression



- Construct Merkle tree on top of public keys, root is the new public key;
- Include Merkle proof in the signature.

Performance

	$t = 1$ **bit challenges**	$t = 16$ **bits challenges**	**PK compression**
Sig size	20 KiB	978 B	3136 B
PK size	64 B	4 MiB	32 B
SK size	32 B	16 B	1 MiB
Est. keygen time	30 ms	30 mins	30 mins
Est. sign time	30 hours	6 mins	6 mins
Est. verify time	10 hours	2 mins	2 mins
Asymptotic sig size	$O(\lambda^2 \log(\lambda))$	$O(\lambda t \log(\lambda))$	$O(\lambda^2 t)$

Performance

	$t = 1$ **bit challenges**	$t = 16$ **bits challenges**	**PK compression**
Sig size	20 KiB	978 B	3136 B
PK size	64 B	4 MiB	32 B
SK size	32 B	16 B	1 MiB
Est. keygen time	30 ms	30 mins	30 mins
Est. sign time	30 hours	6 mins	6 mins
Est. verify time	10 hours	2 mins	2 mins
Asymptotic sig size	$O(\lambda^2 \log(\lambda))$	$O(\lambda t \log(\lambda))$	$O(\lambda^2 t)$

Recent speed/size compromises by Decru, Panny and Vercauteren

Sig size	36 KiB	2 KiB	—
Est. sign time	30 mins	80 s	—
Est. verify time	20 mins	20 s	—

Security proofs

Standard proofs using forking lemma

- ROM only, non tight;
- Secret key space $\#[-B, B]^n \gg \sqrt{\#\mathbb{F}_p}$ to (heuristically) cover all the isogeny graph, but:
  - Public keys not uniformly sampled $\Rightarrow$ problematic random-self reduction;
  - Only managed to reduce to a one-out-of-2^{2t} isogeny walk problem.

Security proofs

Standard proofs using forking lemma

- ROM only, non tight;
- Secret key space $\#[-B,B]^n \gg \sqrt{\#\mathbb{F}_p}$ to (heuristically) cover all the isogeny graph, but:
  - Public keys not uniformly sampled $\Rightarrow$ problematic random-self reduction;
  - Only managed to reduce to a one-out-of-2^{2t} isogeny walk problem.

Alternative proofs based on *lossy keys* (Kiltz, Lyubashevsky and Schaffner 2018)

- ROM, QROM, tight!
- Requires $\#[-B,B]^n \ll \sqrt{\#\mathbb{F}_p}$:
  - Public keys cover a small fraction of the isogeny graph;
  - Asymptotically natural choice for quantum security;
- Additional assumption on indistinguishability of public keys.

Take home (msg, σ)

- By combining ideas from isogeny + lattice + hash based signatures, we give work to all cryptanalysts in this room.
- Post-quantum isogeny signatures are still far from practical.
- Post-post-quantum isogeny signatures look more realistic, you can start using them now if you are an isogeny hippie.
- Tons of open questions on classical and quantum security, and proofs.
- The isogenista dream: a one-pass post-quantum signature scheme based on walks in isogeny graphs.

Thank you

https://defeo.lu/

@luca_defeo