Verifiable Delay Functions and More from Isogenies and Pairings

Luca De Feo

based on joint work with J. Burdges, S. Masson, C. Petit, A. Sanso
IBM Research Zürich

December 4, 2019, ECC, Bochum

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

Distributed lottery

Participants **A, B, . . . , Z** want to agree on a random winning ticket.

Flawed protocol

- Each participant x broadcasts a random string s_x;
- Winning ticket is $H(s_A, \dots, s_Z)$.

Distributed lottery

Participants **A, B, . . . , Z** want to agree on a random winning ticket.

Flawed protocol

- Each participant x broadcasts a random string s_x;
- Winning ticket is $H(s_A, \ldots, s_Z)$.

Cheating participant **Z** waits to see all other strings, then brute-forces s_Z to win lottery.

Distributed lottery

Participants **A, B, . . . , Z** want to agree on a random winning ticket.

Flawed protocol

- Each participant x broadcasts a random string s_x;
- Winning ticket is $H(s_A, \ldots, s_Z)$.

Cheating participant **Z** waits to see all other strings, then brute-forces s_Z to win lottery.

Fixes

- Make the hash function **slooooooooooooooooooooooooooooow**;
  - e.g., participants have 10 minutes to submit s_x,
  - outcome will be known after 20 minutes.

Distributed lottery

Participants **A, B, . . . , Z** want to agree on a random winning ticket.

Flawed protocol

- Each participant x broadcasts a random string s_x;
- Winning ticket is $H(s_A, \dots, s_Z)$.

Cheating participant **Z** waits to see all other strings, then brute-forces s_Z to win lottery.

Fixes

- Make the hash function **slooooooooooooooooooooooooooooow**;
  - e.g., participants have 10 minutes to submit s_x,
  - outcome will be known after 20 minutes.
- Make it possible to verify $w = H(s_A, \dots, s_Z)$ **fast**.

Verifiable Delay Functions (Boneh, Bonneau, Bünz, Fisch 2018)

Wanted

Function (family) $f : X \to Y$ s.t.:

- Evaluating $f(x)$ takes long time:
  - uniformly long time,
  - on almost all random inputs x,
  - even after having seen many values of $f(x')$,
  - even given massive number of processors;
- Verifying $y = f(x)$ is efficient:
  - ideally, exponential separation between evaluation and verification.

Verifiable Delay Functions (Boneh, Bonneau, Bünz, Fisch 2018)

Wanted

Function (family) $f : X \to Y$ s.t.:

- Evaluating $f(x)$ takes long time:
  - uniformly long time,
  - on almost all random inputs x,
  - even after having seen many values of $f(x')$,
  - even given massive number of processors;
- Verifying $y = f(x)$ is efficient:
  - ideally, exponential separation between evaluation and verification.

Exercise

Verifiable Delay Functions (Boneh, Bonneau, Bünz, Fisch 2018)

Wanted

Function (family) $f : X \to Y$ s.t.:

- Evaluating $f(x)$ takes long time:
  - uniformly long time,
  - on almost all random inputs x,
  - even after having seen many values of $f(x')$,
  - even given massive number of processors;
- Verifying $y = f(x)$ is efficient:
  - ideally, exponential separation between evaluation and verification.

Exercise

Think of a function you like with these properties

Verifiable Delay Functions (Boneh, Bonneau, Bünz, Fisch 2018)

Wanted

Function (family) $f : X \to Y$ s.t.:

- Evaluating $f(x)$ takes long time:
  - uniformly long time,
  - on almost all random inputs x,
  - even after having seen many values of $f(x')$,
  - even given massive number of processors;
- Verifying $y = f(x)$ is efficient:
  - ideally, exponential separation between evaluation and verification.

Exercise

Think of a function you like with these properties

Got it?

Verifiable Delay Functions (Boneh, Bonneau, Bünz, Fisch 2018)

Wanted

Function (family) $f : X \to Y$ s.t.:

- Evaluating $f(x)$ takes long time:
  - uniformly long time,
  - on almost all random inputs x,
  - even after having seen many values of $f(x')$,
  - even given massive number of processors;
- Verifying $y = f(x)$ is efficient:
  - ideally, exponential separation between evaluation and verification.

Exercise

Think of a function you like with these properties

Got it?

You're probably wrong!

Sequentiality

Ideal functionality:

$$y = f(x) = \underbrace{H(H(\cdots(H(x))))}_{T \text{ times}}$$

- Sequential assuming hash output "unpredictability",
- but how do you verify? (you're not allowed to say "SNARKs")

VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.

• x

VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

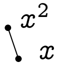
- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.



VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

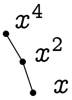
- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.



VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

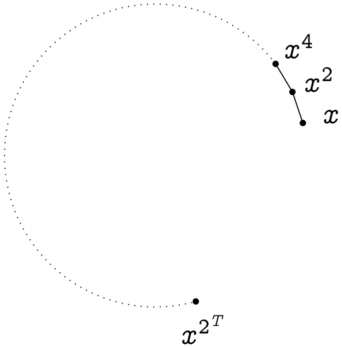
- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.



VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

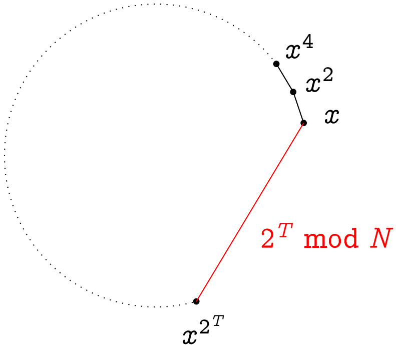
- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.



VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.

Verification

Interactive proofs that $y = f(x)$,
(non interactivity via Fiat-Shamir):

VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.

Verification

Interactive proofs that $y = f(x)$,

(non interactivity via Fiat-Shamir):

Pietrzak '19:

- Proof size $O(\log(T))$,
- Hard to find (non-trivial) $w \in G$ of known order

  $\Rightarrow$ Proof is sound.

VDFs from groups of unknown order (inspired by Rivest–Shamir–Wagner time-lock puzzle)

Setup

A group of unknown order, e.g.:

- $\mathbb{Z}/N\mathbb{Z}$ with $N = pq$ an RSA modulus, p, q unknown (e.g., generated by some trusted authority),
- Class group of imaginary quadratic order.

Evaluation

With delay parameter T:

$$\begin{aligned} f : G &\longrightarrow G \\ x &\longmapsto x^{2^T} \end{aligned}$$

Conjecturally, fastest algorithm is repeated squaring.

Verification

Interactive proofs that $y = f(x)$,
(non interactivity via Fiat-Shamir):

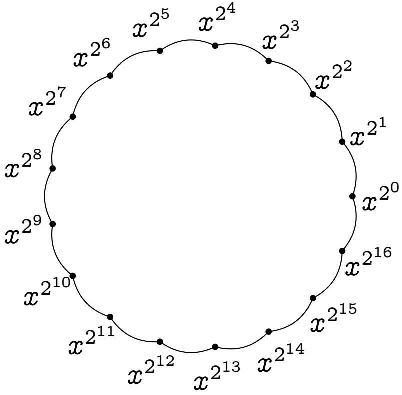
Pietrzak '19:

- Proof size $O(\log(T))$,
- Hard to find (non-trivial) $w \in G$ of known order $\Rightarrow$ Proof is sound.

Wesolowski '19:

- Proof size $O(1)$,
- More emphad hoc security assumption.

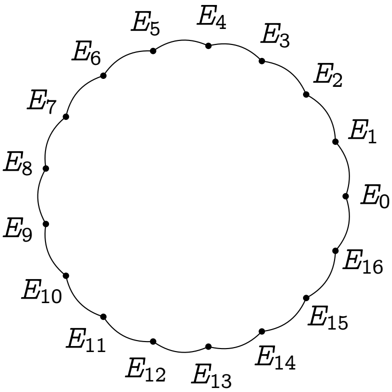
Where have I seen this before?



Where have I seen this before?

Isogeny cycles

- Vertices are elliptic curves:
  - Ordinary,
  - Supersingular $/\mathbb{F}_p$.
- Edges are horizontal isogenies.

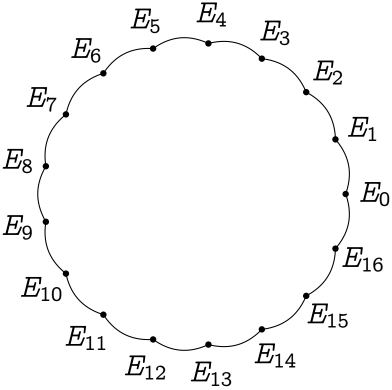


Where have I seen this before?

Isogeny cycles

- Vertices are elliptic curves:
  - Ordinary,
  - Supersingular $/\mathbb{F}_p$.
- Edges are horizontal isogenies.
- The class group of $\mathrm{End}(E)$ acts upon the cycle:

isogeny	$\leftrightarrow$	ideal
endomorphism	$\leftrightarrow$	principal ideal
degree	$\leftrightarrow$	norm
dual	$\leftrightarrow$	complex conjugate
cycle size	$\leftrightarrow$	order of the ideal

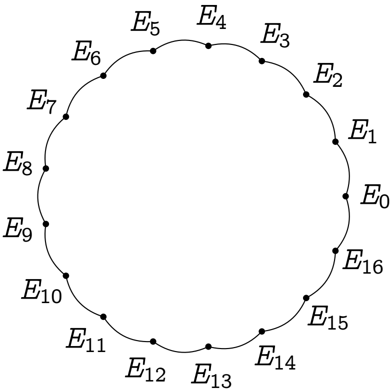


Where have I seen this before?

Isogeny cycles

- Vertices are elliptic curves:
  - Ordinary, Couveignes–Rostovtsev–Stolbunov
  - Supersingular $/\mathbb{F}_p$. CSIDH
- Edges are horizontal isogenies.
- The class group of $\mathrm{End}(E)$ acts upon the cycle:

isogeny	$\leftrightarrow$	ideal
endomorphism	$\leftrightarrow$	principal ideal
degree	$\leftrightarrow$	norm
dual	$\leftrightarrow$	complex conjugate
cycle size	$\leftrightarrow$	order of the ideal

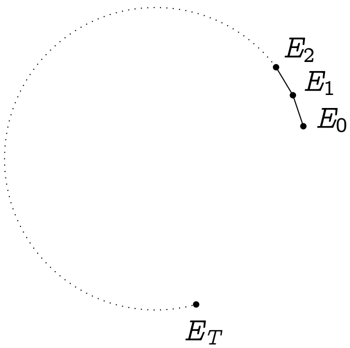


Sloooooooooooooooooooooooooooooooooooooow isogenies

Setup

With delay parameter T:

- A laaaaaaaaaaaaaaaaaaaaaaaarge isogeny cycle,

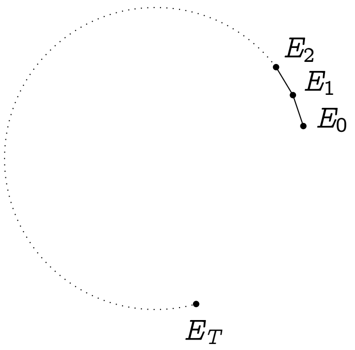


Sloooooooooooooooooooooooooooooooooooooow isogenies

Setup

With delay parameter T:

- A laaaaaaaaaaaaaaaaaaaaaaaarge isogeny cycle,
- A starting curve E_0,
- An isogeny $\phi : E_0 \to E_T$ of degree 2^T.



Slooooooooooooooooooooooooooooooooooooooow isogenies

Setup

With delay parameter T:

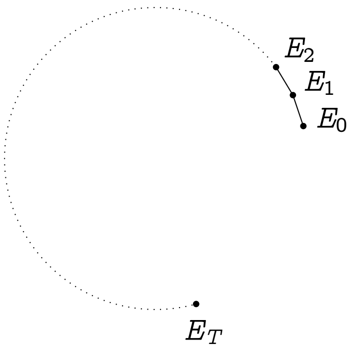
- A laaaaaaaaaaaaaaaaaaaaaaarge isogeny cycle,
- A starting curve E_0,
- An isogeny $\phi : E_0 \to E_T$ of degree 2^T.

Evaluation

ϕ **is** the VDF:

$$\begin{aligned} \phi : E_0(\mathbb{F}_p) &\longrightarrow E_T(\mathbb{F}_p) \\ P &\longmapsto \phi(P) \end{aligned}$$

Conjecturally, no faster way than composing degree 2 isogenies.



Sloooooooooooooooooooooooooooooooooooooow isogenies

Setup

With delay parameter T:

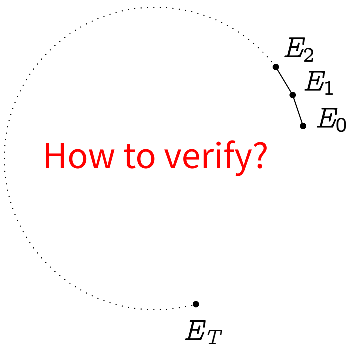
- A laaaaaaaaaaaaaaaaaaaaaaaarge isogeny cycle,
- A starting curve E_0,
- An isogeny $\phi : E_0 \to E_T$ of degree 2^T.

Evaluation

ϕ **is** the VDF:

$$\begin{aligned} \phi : E_0(\mathbb{F}_p) &\longrightarrow E_T(\mathbb{F}_p) \\ P &\longmapsto \phi(P) \end{aligned}$$

Conjecturally, no faster way than composing degree 2 isogenies.



Isogeny <3 Pairing

Theorem

Let $\phi : E \to E'$ be an isogeny and $\hat{\phi} : E' \to E$ its dual. Let e_N be the Weil pairing of E and e'_N that of E'. Then

$$e_N(P, \hat{\phi}(Q)) = e'_N(\phi(P), Q),$$

for any $P \in E[N]$ and $Q \in E'[N]$.

Corollary

$$e'_N(\phi(P), \phi(Q)) = e_N(P, Q)^{\deg \phi}.$$

Refresher: Boneh–Lynn–Shacham (BLS) signatures

Setup:
- Elliptic curve $E/\mathbb{F}_p$, s.t $N | \#E(\mathbb{F}_p)$ for a large prime N,
- (Weil) pairing $e_N : E[N] \times E[N] \to \mathbb{F}_{p^k}$ for some small embedding degree k,
- A decomposition $E[N] = X_1 \times X_2$, with $X_1 = \langle P \rangle$.
- A hash function $H : \{0,1\}^* \to X_2$.

Private key: $s \in \mathbb{Z}/N\mathbb{Z}$.

Public key: sP.

Sign: $m \mapsto sH(m)$.

Verifiy: $e_N(P, sH(m)) = e_N(sP, H(m))$.

$$\begin{tikzcd} X_1 \times X_2 \arrow[r, "{[s] \times 1}"] \arrow[d, "{1 \times [s]}"'] & X_1 \times X_2 \arrow[d, "e_N"] \\ X_1 \times X_2 \arrow[r, "e_N"'] & \mathbb{F}_{p^k} \end{tikzcd}$$

US patent 8,250,367 (Broker, Charles and Lauter 2012)

Signatures from isogenies + pairings

- Replace the secret $[s] : E \to E$ with an isogeny $\phi : E \to E'$;
- Define decompositions

$$E[N] = X_1 \times X_2, \qquad E'[N] = Y_1 \times Y_2,$$

  s.t. $\phi(X_1) = Y_1$ and $\phi(X_2) = Y_2$;
- Define a hash function $H : \{0,1\}^* \to Y_2$.

$$\begin{tikzcd}
X_1 \times Y_2 \arrow[r, "\phi \times 1"] \arrow[d, "1 \times \hat\phi"'] & Y_1 \times Y_2 \arrow[d, "e'_N"] \\
X_1 \times X_2 \arrow[r, "e_N"'] & \mathbb{F}_{p^k}
\end{tikzcd}$$

Isogeny VDF (principle)

Setup

- Pairing friendly curve E,
- Isogeny $\phi : E \to E'$ of degree ℓ^T,
- Point $P \in X_1$, image $\phi(P) \in Y_1$.

Evaluation

Input: random $Q \in Y_2$,

Output: $\hat{\phi}(Q) \in X_2$.

Verification

$$e_N(P, \hat{\phi}(Q)) \stackrel{?}{=} e'_N(\phi(P), Q).$$

Instantiation over $\mathbb{F}_p$

The curves

$$\left.\begin{array}{l}\bullet\ \text{Need a \textit{large enough} isogeny class;}\\ \bullet\ \text{Need pairing friendliness;}\end{array}\right\} \Rightarrow \text{supersingular curves.}$$

Technicalities

Instantiation over $\mathbb{F}_p$

The curves

- Need a *large enough* isogeny class;
- Need pairing friendliness;

$\Big\} \Rightarrow$ supersingular curves.

Technicalities

- Choose $p + 1 = N \cdot f$,
  - for degree $\ell = 2$ also need $8|f$;

Instantiation over $\mathbb{F}_p$

The curves

$$\left.\begin{array}{l}\bullet\ \text{Need a \textit{large enough} isogeny class;}\\ \bullet\ \text{Need pairing friendliness;}\end{array}\right\} \Rightarrow \text{supersingular curves.}$$

Technicalities

- Choose $p+1 = N \cdot f$,
  - for degree $\ell = 2$ also need $8|f$;
- Choose $E/\mathbb{F}_p$ on an ℓ-isogeny cycle
  - If $\ell = 2 \Rightarrow$ choose E with maximal endomorphism ring;
  - Otherwise $\left(\frac{-p}{\ell}\right) = 1$.

Instantiation over $\mathbb{F}_p$

The curves

- Need a *large enough* isogeny class;
- Need pairing friendliness;

$\Big\}\Rightarrow$ supersingular curves.

Technicalities

- Choose $p+1 = N\cdot f$,
  - for degree $\ell = 2$ also need $8|f$;
- Choose $E/\mathbb{F}_p$ on an ℓ-isogeny cycle
  - If $\ell = 2 \Rightarrow$ choose E with maximal endomorphism ring;
  - Otherwise $\left(\frac{-p}{\ell}\right) = 1$.
- There are only two ℓ^T-isogenies from E, choose any.

Instantiation over $\mathbb{F}_p$

The curves

$$\left.\begin{array}{l}\bullet\ \text{Need a \textit{large enough} isogeny class;}\\ \bullet\ \text{Need pairing friendliness;}\end{array}\right\} \Rightarrow \text{supersingular curves.}$$

Technicalities

- Choose $p + 1 = N \cdot f$,
  - for degree $\ell = 2$ also need $8|f$;
- Choose $E/\mathbb{F}_p$ on an ℓ-isogeny cycle
  - If $\ell = 2 \Rightarrow$ choose E with maximal endomorphism ring;
  - Otherwise $\left(\frac{-p}{\ell}\right) = 1$.
- There are only two ℓ^T-isogenies from E, choose any.
- Set $X_2 = E[N] \cap E(\mathbb{F}_p)$ and X_1 as the other eigenspace of Frobenius:
  - Short notation: $X_1 = E[(N, \pi + 1)]$, $X_2 = E[(N, \pi - 1)]$.
  - Similarly: $Y_1 = E'[(N, \pi + 1)]$, $Y_2 = E'[(N, \pi - 1)]$.

Instantiation over $\mathbb{F}_{p^2}$

There's nothing special with isogeny cycles

- May as well use isogeny walks in the full supersingular graph
  (like Charles–Goren–Lauter, SIDH, ...)
- But we still need a canonical decomposition $E[N] = X_1 \times X_2$
  $\Rightarrow$ start from $E/\mathbb{F}_p$.

Technicalities

Instantiation over $\mathbb{F}_{p^2}$

There's nothing special with isogeny cycles

- May as well use isogeny walks in the full supersingular graph (like Charles–Goren–Lauter, SIDH, ...)
- But we still need a canonical decomposition $E[N] = X_1 \times X_2$ $\Rightarrow$ start from $E/\mathbb{F}_p$.

Technicalities

- $p + 1 = N \cdot f$, no conditions on (p, ℓ);

Instantiation over $\mathbb{F}_{p^2}$

There's nothing special with isogeny cycles

- May as well use isogeny walks in the full supersingular graph
  (like Charles–Goren–Lauter, SIDH, ...)
- But we still need a canonical decomposition $E[N] = X_1 \times X_2$
  $\Rightarrow$ start from $E/\mathbb{F}_p$.

Technicalities

- $p + 1 = N \cdot f$, no conditions on (p, ℓ);
- There are exponentially many ℓ^T-isogenies, choose any (pseudorandomly);

Instantiation over $\mathbb{F}_{p^2}$

There's nothing special with isogeny cycles

- May as well use isogeny walks in the full supersingular graph (like Charles–Goren–Lauter, SIDH, ...)
- But we still need a canonical decomposition $E[N] = X_1 \times X_2$
$\Rightarrow$ start from $E/\mathbb{F}_p$.

Technicalities

- $p + 1 = N \cdot f$, no conditions on (p, ℓ);
- There are exponentially many ℓ^T-isogenies, choose any (pseudorandomly);
- Impossible to hash into $Y_2 = \phi(X_2)$:
  - Domain of VDF is all of $E'[N]$;
  - To make the protocol sound we compose $\hat{\phi}$ with the trace of $E/\mathbb{F}_{p^2}$.

Comparison

	Wesolowski		Pietrzak		Ours	
	RSA	class group	RSA	class group	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$
proof size	$O(1)$	$O(\log(T))$	$O(1)$	$O(\log(T))$	—	—
aggregatable	yes	yes	yes	yes	—	—
watermarkable	yes	yes	yes	yes	(yes)	(yes)
perfect soundness	no	no	no	no	yes	yes
long setup	no	no	no	no	yes	yes
trusted setup	yes	no	yes	no	yes	yes
best attack	$L_N(1/3)$	$L_N(1/2)$	$L_N(1/3)$	$L_N(1/2)$	$L_p(1/3)$	$L_p(1/3)$
quantum annoying	no	no	no	no	no	yes

Implementation

- PoC implementation in SageMath (re-implemented Montgomery isogenies);
- $p + 1 = N \cdot 2^{1244} \cdot 63$, enables time/memory compromise in evaluation.

Protocol	**Step**	**Parameters size** ($T \approx 2^{16}$)	**Time**	**Throughput**
$\mathbb{F}_p$ graph	Setup	238 kb	—	0.75 isog/ms
	Evaluation	—	—	0.75 isog/ms
	Verification	—	0.3 s	—
$\mathbb{F}_{p^2}$ graph	Setup	491 kb	—	0.35 isog/ms
	Evaluation	—	—	0.23 isog/ms
	Verification	—	4 s	—

Table: Benchmarks (Intel Core i7-8700 @3.20GHz) at 128 bits of security (aggressively optimizing for size).

Security

Attacks

Security goal

Given the isogeny $\phi : E \to E$, the adversary is allowed $\mathtt{poly}(T)$ precomputation.

Later, it is given a random $Q \in Y_2$:
its probability of computing $\hat{\phi}(Q)$ in less than "T steps" must be negligible.

Attack avenues:

1. Speed-up/parallelize isogeny computation;
2. Solve the pairing equation;
3. Find isogeny *shortcuts*.

Attacking the computation?

RSA: $$x \longmapsto x^2 \mod N$$

Isogenies: $$x \longmapsto x\frac{x\alpha_i - 1}{x - \alpha_i} \mod p$$

($\alpha_1, \dots, \alpha_T$ depend on the chosen isogeny)

e.g., $\log_2 N \approx 2048$, $\log_2 p \approx 1500$.

No speedup? Even with unlimited parallelism? Really?

See Bernstein, Sorenson. Modular exponentiation via the explicit Chinese remainder theorem.

Attacking the pairing

A pairing inversion problem:

$$e(P, ???) = e(\phi(P), Q)$$

Quantum: Broken by Shor's algorithm;

Classical: Subexponential $L_p(1/3)$ attack.

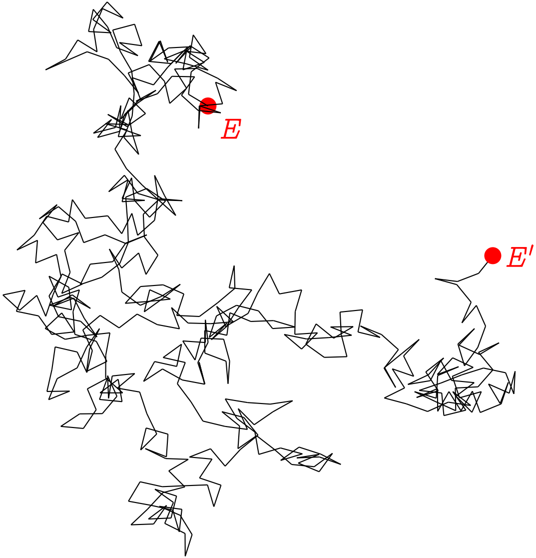
Note: Solving the equation gives the true value of $\hat{\phi}(Q)$ (perfect soundness)

Computing *shortcuts*

- Isogeny degree $= \ell^T \leftrightarrow$ walk length $= T$;
  - e.g., for delay $\approx$ 1 hour, $T \approx 2^{20}$;

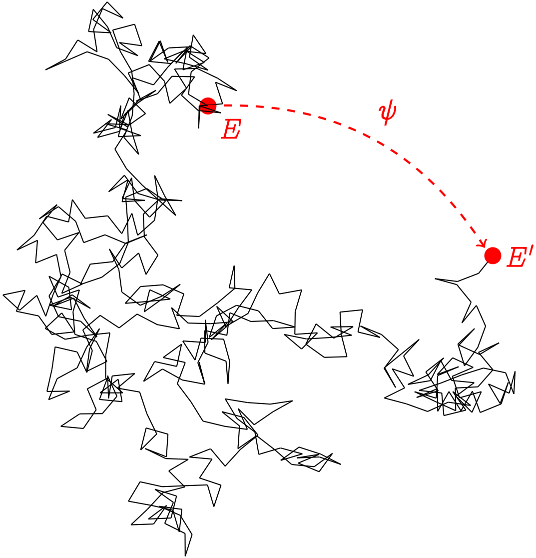
Computing *shortcuts*

- Isogeny degree $= \ell^T$ $\leftrightarrow$ walk length $= T$;
  - e.g., for delay ≈ 1 hour, $T \approx 2^{20}$;
  - Typically much larger than graph diameter ($= O(\log p) \approx 2^{10}$).
  - (which isogeny graph is meant depends on the variant)



Computing *shortcuts*

- Isogeny degree $= \ell^T \leftrightarrow$ walk length $= T$;
  - e.g., for delay $\approx$ 1 hour, $T \approx 2^{20}$;
  - Typically much larger than graph diameter ($= O(\log p) \approx 2^{10}$).
  - (which isogeny graph is meant depends on the variant)
- Goal: find a *shortcut*, i.e., a shorter walk.



$\mathrm{End}(E)$ gives shortcuts

$\mathbb{F}_p$ case

- $\mathrm{End}_{\mathbb{F}_p}(E) \subset \mathbb{Q}(\sqrt{-p})$:
  the class group $\mathrm{Cl}(-4p)$ acts on the set of supersingular curves $/\mathbb{F}_p$;
- Structure of $\mathrm{Cl}(-4p)$

  $\Updownarrow$

  relations between ideal classes

  $\Updownarrow$

  shortcuts in the graph.
  - see CSI-FiSh signatures (Beullens–Kleinjung–Vercauteren);
  - akin to attack on class group VDF.
- Some additional work to find endomorphism ω such that $\omega \circ \hat{\psi}(Q) = \hat{\phi}(Q)$.

$\mathrm{End}(E)$ gives shortcuts

$\mathbb{F}_p$ case

- $\mathrm{End}_{\mathbb{F}_p}(E) \subset \mathbb{Q}(\sqrt{-p})$:
  the class group $\mathrm{Cl}(-4p)$ acts on the set of supersingular curves $/\mathbb{F}_p$;
- Structure of $\mathrm{Cl}(-4p)$

  $\Updownarrow$

  relations between ideal classes

  $\Updownarrow$

  shortcuts in the graph.
  - see CSI-FiSh signatures (Beullens–Kleinjung–Vercauteren);
  - akin to attack on class group VDF.
- Some additional work to find endomorphism ω such that $\omega \circ \hat{\psi}(Q) = \hat{\phi}(Q)$.

General case (both $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$)

- $\mathrm{End}(E)$ isomorphic to an order in a quaternion algebra;
- Structure of $\mathrm{End}(E)$ (or $\mathrm{End}(E')$)

  $\Updownarrow$

  shortcuts (through $\mathbb{F}_{p^2}$).
  - Related to attacks on the Charles–Goren–Lauter hash function.
- Additional work to find $\omega \in \mathrm{End}(E)$.

$\mathrm{End}(E)$ gives shortcuts

$\mathbb{F}_p$ case

- $\mathrm{End}_{\mathbb{F}_p}(E) \subset \mathbb{Q}(\sqrt{-p})$:
  the class group $\mathrm{Cl}(-4p)$ acts on the set of supersingular curves $/\mathbb{F}_p$;
- Structure of $\mathrm{Cl}(-4p)$

  $\Updownarrow$

  relations between ideal classes

  $\Updownarrow$

  shortcuts in the graph.
  - see CSI-FiSh signatures (Beullens–Kleinjung–Vercauteren);
  - akin to attack on class group VDF.
- Some additional work to find endomorphism ω such that $\omega \circ \hat{\psi}(Q) = \hat{\phi}(Q)$.

General case (both $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$)

- $\mathrm{End}(E)$ isomorphic to an order in a quaternion algebra;
- Structure of $\mathrm{End}(E)$ (or $\mathrm{End}(E')$)

  $\Updownarrow$

  shortcuts (through $\mathbb{F}_{p^2}$).
  - Related to attacks on the Charles–Goren–Lauter hash function.
- Additional work to find $\omega \in \mathrm{End}(E)$.

WE HAVE A PROBLEM!

No known way to construct supersingular curves without knowledge of $\mathrm{End}(E)$.

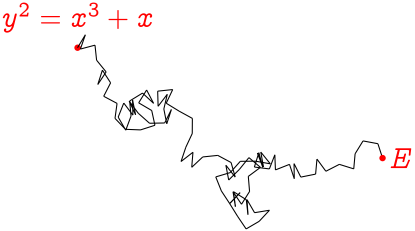
Only known fix: **Trusted setup**.

Trusted setup

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

- Start from a well known supersingular curve,

Trusted setup



- Start from a well known supersingular curve,
- Do a random walk,

Trusted setup

$y^2 = x^3 + x$

E

- Start from a well known supersingular curve,
- Do a random walk,
- Forget it.

Trusted setup

$y^2 = x^3 + x$

E

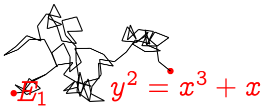
- Start from a well known supersingular curve,
- Do a random walk,
- Forget it.

	Classical		Quantum	
	$\mathbb{F}_p$ graph	$\mathbb{F}_{p^2}$ graph	$\mathbb{F}_p$ graph	$\mathbb{F}_{p^2}$ graph
Computing shortcuts	$L_p(1/2)$	$O(\sqrt{p})$	$\mathrm{polylog}(p)$	$O(\sqrt[4]{p})$
Pairing inversion	$L_p(1/3)$	$L_p(1/3)$	$\mathrm{polylog}(p)$	$\mathrm{polylog}(p)$

Quantum annoyance:

- Computing shortcuts in $\mathbb{F}_{p^2}$ is quantumly hard;
- Pairing inversion attacks must be run online, useless if Shor's algorithm takes much longer than target delay.

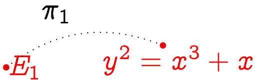
Distributed trusted setups



Mitigate trusted setup woes by distributing trust:

- Participant i performs a random walk (in $\mathbb{F}_p$),

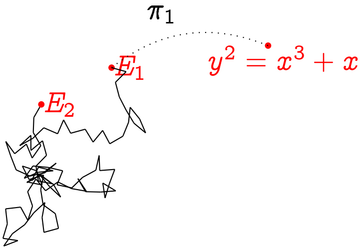
Distributed trusted setups



Mitigate trusted setup woes by distributing trust:

- Participant i performs a random walk (in $\mathbb{F}_p$),
- Publishes a proof of isogeny knowledge,

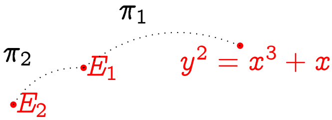
Distributed trusted setups



Mitigate trusted setup woes by distributing trust:

- Participant i performs a random walk (in $\mathbb{F}_p$),
- Publishes a proof of isogeny knowledge,
- Repeat.

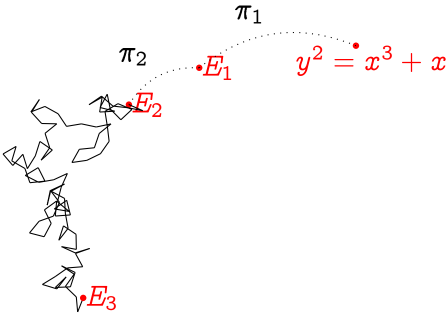
Distributed trusted setups



Mitigate trusted setup woes by distributing trust:

- Participant i performs a random walk (in $\mathbb{F}_p$),
- Publishes a proof of isogeny knowledge,
- Repeat.

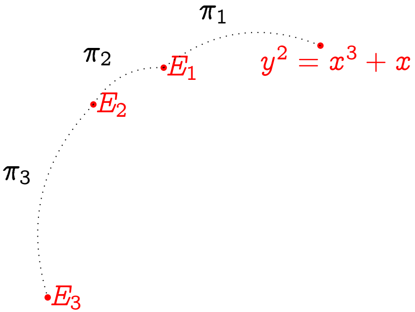
Distributed trusted setups



Mitigate trusted setup woes by distributing trust:

- Participant i performs a random walk (in $\mathbb{F}_p$),
- Publishes a proof of isogeny knowledge,
- Repeat.

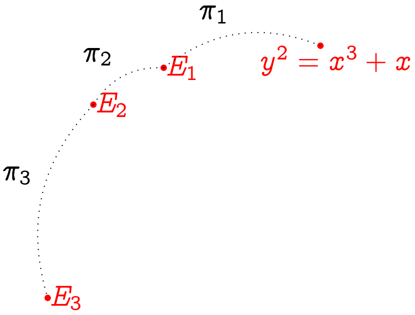
Distributed trusted setups



Mitigate trusted setup woes by distributing trust:

- Participant i performs a random walk (in $\mathbb{F}_p$),
- Publishes a proof of isogeny knowledge,
- Repeat.

Distributed trusted setups



Mitigate trusted setup woes by distributing trust:

- Participant i performs a random walk (in $\mathbb{F}_p$),
- Publishes a proof of isogeny knowledge,
- Repeat.

Proof options:

- Generic ZK proofs,
- Isogeny ZK proofs (SeaSign),
- Pairing proofs (not ZK!):

$$P, Q = \mathcal{H}(E_i, E_{i+1}),$$
$$e_i(P, \hat{\phi}_i(Q)) = e_{i+1}(\phi_i(P), Q).$$

Properties: asynchronous, robust against $n-1$ coalition, verification scales linearly, updatable, …

Beyond VDFs


DB
Ziel Destination
Gleis Platform/Voie
Mannheim-Friedrich 11
Gernsheim 17 Train is cancelled
Köln Hbf 7 Train is cancelled
Berlin Hbf 9 Train is cancelled
Passau Hbf 6 Train is cancelled
Siegen 16
Saarbrücken Hbf 20
Fulda 8 Train is cancelled
Bruxelles-Midi 19 Aujourd hui du qua
Hanau Hbf 5 ai 5 - Heute auf G
r DB-Zugverkehr beeinträchtigt. Bitte
nd informieren Sie sich auch im Internet

Watermarking

Goal: reward evaluator for its effort.

Watermarking: issue proof of evaluation tied to evaluator identity

$$\begin{tikzcd}
E & E_{\mathrm{mid}} \arrow[l, bend right, "\hat\phi_2"'] & E' \arrow[l, bend right, "\hat\phi_1"'] \arrow[ll, bend left, "\hat\phi = \hat\phi_2 \circ \hat\phi_1"]
\end{tikzcd}$$

Secret key: scalar $s \in \mathbb{Z}/N\mathbb{Z}$,

Public key: $s\phi(P) \in E'$ (+ proof of exponent knowledge),

Proof of work: $s\hat\phi_1(Q) \in E_{\mathrm{mid}}$,

Verification: $e_{\mathrm{mid}}(\phi_2(P), s\hat\phi_1(Q)) = e'(s\phi(P), Q)$.

Properties: blind (can be checked before the computation is complete).

Encryption to the future (time-locks)

Goal: encrypt now, decryption only possible after delay.

Applications: auctions, voting, …

Idea: start from Boneh–Franklin IBE, just add isogenies™.

Bidder		**Auctioneer**
		Publishes auction key $Q = \mathcal{H}(\mathrm{sid})$
		starts evaluating $\hat{\phi}(Q)$
samples random $s \in \mathbb{Z}/N\mathbb{Z}$		
computes $k = e(\phi(P), Q)^s$		
encrypts offer $o_k = \mathrm{Enc}_k(o)$		
sends (o_k, sP)	$\longrightarrow$	
		$\vdots$
		computes $k = e(sP, \hat{\phi}(Q))$
		decrypts o_k

Open questions

- Understand the impact of large memory requirements in evaluation; is a time/memory trade-off reasonable?

Open questions

- Understand the impact of large memory requirements in evaluation; is a time/memory trade-off reasonable?
- Remove trusted setup:
  - Hash into the supersingular set, or
  - Construct ordinary pairing friendly curves with large discriminant.

Open questions

- Understand the impact of large memory requirements in evaluation; is a time/memory trade-off reasonable?
- Remove trusted setup:
  - Hash into the supersingular set, or
  - Construct ordinary pairing friendly curves with large discriminant.
- Explore more advanced pairing+delay constructions.

Open questions

- Understand the impact of large memory requirements in evaluation; is a time/memory trade-off reasonable?
- Remove trusted setup:
  - Hash into the supersingular set, or
  - Construct ordinary pairing friendly curves with large discriminant.
- Explore more advanced pairing+delay constructions.
- Spend millions on dedicated hardware for 2-isogenies.

Open questions

- Understand the impact of large memory requirements in evaluation; is a time/memory trade-off reasonable?
- Remove trusted setup:
  - Hash into the supersingular set, or
  - Construct ordinary pairing friendly curves with large discriminant.
- Explore more advanced pairing+delay constructions.
- Spend millions on dedicated hardware for 2-isogenies.

Just Add Isogenies™!

Thank you

https://defeo.lu/

@luca_defeo