Verifiable Delay Functions from Supersingular Isogenies and Pairings

Luca De Feo[1] *Simon Masson*[2] Christophe Petit[3] Antonio Sanso[4]

[1]IBM Research Zürich, CH.

[2]Thales and Université de Lorraine, Nancy, FR.

[3]University of Birmingham, UK.

[4]Adobe Inc. and Ruhr Universität Bochum, DE.

December 9th, 2019

Definition and examples

Definition and examples

VDF based on isogenies and pairings

Implementation and comparison

Definition

A *verifiable delay function* (VDF) is a function $f : X \longrightarrow Y$ such that

1. it takes T steps to evaluate, even with massive amounts of parallelism
2. the output can be verified efficiently.

Definition

A *verifiable delay function* (VDF) is a function $f : X \longrightarrow Y$ such that

1. it takes T steps to evaluate, even with massive amounts of parallelism
2. the output can be verified efficiently.

- Setup$(\lambda, T) \longrightarrow$ public parameters pp
- Eval$(pp, x) \longrightarrow$ output y such that $y = f(x)$, and a proof π (requires T steps)
- Verify$(pp, x, y, \pi) \longrightarrow$ yes or no.

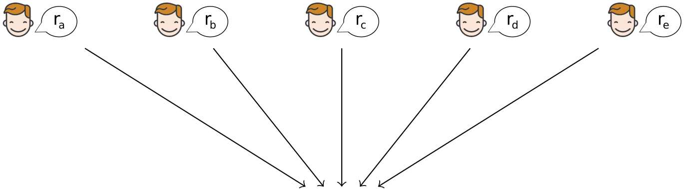
Application. Distributed randomness

Application. Distributed randomness

Application. Distributed randomness

$$R = r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$$

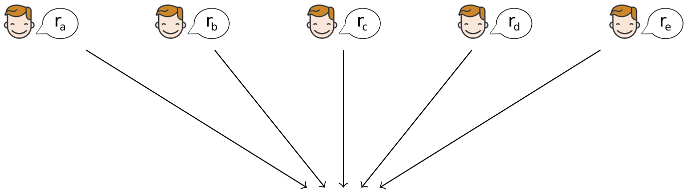
Application. Distributed randomness



$$R = r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$$

seems to be random...

Application. Distributed randomness

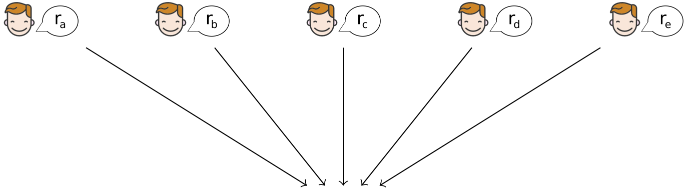


$$R = r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$$

seems to be random...

but someone can control the randomness !

Application. Distributed randomness



$$R = r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$$

seems to be random...

but someone can control the randomness !

Idea: slow things down by *adding delay*.

Application. Distributed randomness

$$R = r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$$

$$\downarrow$$

$$h(R) \in X$$

$$\Big\downarrow \text{VDF}$$

$$f(h(R))$$

seems to be random...

but someone can control the randomness !

Idea: slow things down by *adding delay*.

VDF based on RSA.

Setup. $\mathbb{Z}/N\mathbb{Z}$ where N is a RSA modulus

VDF based on RSA.

Setup. $\mathbb{Z}/N\mathbb{Z}$ where N is a RSA modulus

Evaluation. $y = x^{2^T} \mod N$.

VDF based on RSA.

Setup. $\mathbb{Z}/N\mathbb{Z}$ where N is a RSA modulus

Evaluation. $y = x^{2^T} \mod N$.

Verification. The evaluator also sends a proof π to convince the verifier.

VDF based on RSA.

Setup. $\mathbb{Z}/N\mathbb{Z}$ where N is a RSA modulus

Evaluation. $y = x^{2^T} \mod N$.

Verification. The evaluator also sends a proof π to convince the verifier.

- **Wesolowski verification.** [Eurocrypt '19]
  π is short
  Verification is fast.
- **Pietrzak verification.** [ITCS '19]
  π computation is more efficent
  Verification is slower.

Different security assumptions.

VDF based on RSA.

If one knows the factorization of N, the evaluation can be computed using

$$h(x)^{2^T} \equiv h(x)^{2^T \bmod \varphi(N)} \mod N$$

Need a *trusted setup* to choose N.

VDF based on RSA.

If one knows the factorization of N, the evaluation can be computed using

$$h(x)^{2^T} \equiv h(x)^{2^T \bmod \varphi(N)} \mod N$$

Need a *trusted setup* to choose N.

This VDF also works in another *group of unknown order*.

VDF based on RSA.
If one knows the factorization of N, the evaluation can be computed using

$$h(x)^{2^T} \equiv h(x)^{2^T \bmod \varphi(N)} \mod N$$

Need a *trusted setup* to choose N.
This VDF also works in another *group of unknown order*.
VDF based on class group. Let $K = \mathbb{Q}(\sqrt{-D})$ and O_K its ring of integers.

$$\text{ClassGroup}(D) = \text{Ideals}(O_K)/\text{PrincipalIdeals}(O_K)$$

This group is finite and it is hard to compute $\#\text{ClassGroup}(D)$.

VDF based on RSA.
If one knows the factorization of N, the evaluation can be computed using

$$h(x)^{2^T} \equiv h(x)^{2^T \bmod \varphi(N)} \mod N$$

Need a *trusted setup* to choose N.
This VDF also works in another *group of unknown order*.
VDF based on class group. Let $K = \mathbb{Q}(\sqrt{-D})$ and O_K its ring of integers.

$$\mathrm{ClassGroup}(D) = \mathrm{Ideals}(O_K)/\mathrm{PrincipalIdeals}(O_K)$$

This group is finite and it is hard to compute $\#\mathrm{ClassGroup}(D)$.

VDF	**pro**	**con**
RSA	fast verification	trusted setup not post-quantum
Class group	small parameters	slow verification not post-quantum

VDF based on isogenies and pairings

Our new verifiable delay functions.

1. Use isogenies to compute the evaluation step.
2. Use a pairing equation to verify the evaluation.

Our new verifiable delay functions.

1. Use isogenies to compute the evaluation step.
2. Use a pairing equation to verify the evaluation.

- What is an isogeny ?
- What is a pairing ?

Pairing-friendly elliptic curves.

Definition

A pairing is a bilinear non-degenerate application

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$$

where $\mathbb{G}_i$ are groups of prime order r.

Pairing-friendly elliptic curves.

Definition

A pairing is a bilinear non-degenerate application

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$$

where $\mathbb{G}_i$ are groups of prime order r.

For an elliptic curve, we can choose $\mathbb{G}_1 = \langle P \rangle$ and $\mathbb{G}_2 = \langle Q \rangle$ with P, Q points of the curve of order r.
A curve is *pairing-friendly* if P and Q are efficiently computable.

Pairing-friendly elliptic curves.

Definition

A pairing is a bilinear non-degenerate application

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$$

where $\mathbb{G}_i$ are groups of prime order r.

For an elliptic curve, we can choose $\mathbb{G}_1 = \langle P \rangle$ and $\mathbb{G}_2 = \langle Q \rangle$ with P, Q points of the curve of order r.
A curve is *pairing-friendly* if P and Q are efficiently computable.
Applications. BLS signature, identity-based encryption, etc.

Isogenies of elliptic curves.

Definition (Isogeny)

An isogeny between two elliptic curves E and E' is an algebraic map φ such that $\varphi(0_E) = 0_{E'}$.

Isogenies of elliptic curves.

Definition (Isogeny)

An isogeny between two elliptic curves E and E' is an algebraic map φ such that $\varphi(0_E) = 0_{E'}$.

- Isogenies of elliptic curves are also group morphisms.
- In our case, the degree of an isogeny is the size of its kernel.
- Isogenies of small degree are efficiently computable.

Isogenies of elliptic curves.

Definition (Isogeny)

An isogeny between two elliptic curves E and E' is an algebraic map φ such that $\varphi(0_E) = 0_{E'}$.

- Isogenies of elliptic curves are also group morphisms.
- In our case, the degree of an isogeny is the size of its kernel.
- Isogenies of small degree are efficiently computable.

Example. $E : y^2 = x^3 - x$ and $K = (1, 0)$ of order 2.

$$\begin{array}{rrcl} \varphi : & E & \longrightarrow & E/\langle K \rangle \\ & (x, y) & \longmapsto & \left(\frac{x^2 - x + 2}{x - 1}, y \frac{x^2 - 2x - 1}{x^2 - 2x + 1} \right) \end{array}$$

Isogenies of elliptic curves.

Definition (Isogeny)

An isogeny between two elliptic curves E and E' is an algebraic map φ such that $\varphi(0_E) = 0_{E'}$.

- Isogenies of elliptic curves are also group morphisms.
- In our case, the degree of an isogeny is the size of its kernel.
- Isogenies of small degree are efficiently computable.

Example. $E : y^2 = x^3 - x$ and $K = (1, 0)$ of order 2.

$$\begin{array}{rccc} \varphi : & E & \longrightarrow & E/\langle K \rangle \\ & (x, y) & \longmapsto & \left(\frac{x^2 - x + 2}{x - 1}, y \frac{x^2 - 2x - 1}{x^2 - 2x + 1} \right) \end{array}$$

From $\varphi : E \to E'$, there always exists a dual isogeny $\hat{\varphi} : E' \to E$ such that $\varphi \circ \hat{\varphi} = [\deg \varphi]$.

Isogenies of elliptic curves.

Definition (Isogeny)

An isogeny between two elliptic curves E and E' is an algebraic map φ such that $\varphi(0_E) = 0_{E'}$.

- Isogenies of elliptic curves are also group morphisms.
- In our case, the degree of an isogeny is the size of its kernel.
- Isogenies of small degree are efficiently computable.

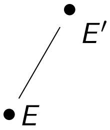
Example. $E : y^2 = x^3 - x$ and $K = (1, 0)$ of order 2.

$$\begin{array}{rccc} \varphi : & E & \longrightarrow & E/\langle K \rangle \\ & (x, y) & \longmapsto & \left(\frac{x^2-x+2}{x-1}, y\frac{x^2-2x-1}{x^2-2x+1} \right) \end{array}$$

From $\varphi : E \to E'$, there always exists a dual isogeny $\hat{\varphi} : E' \to E$ such that $\varphi \circ \hat{\varphi} = [\deg \varphi]$.

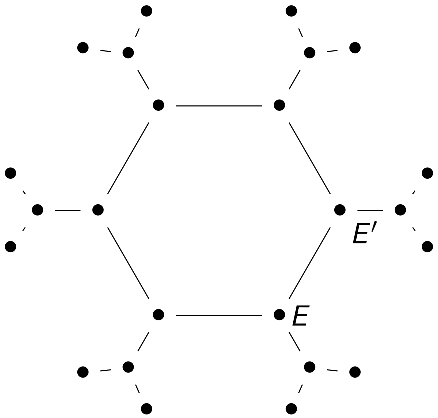
$$e(\varphi(P), Q) = e(P, \hat{\varphi}(Q))$$

Two types of elliptic curves:



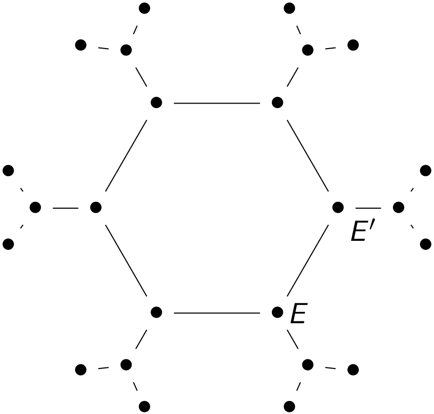
Two types of elliptic curves:

Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.



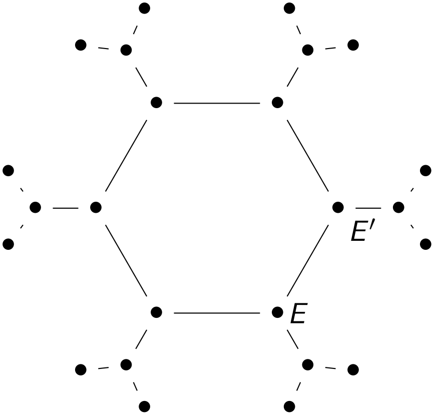
Two types of elliptic curves:

Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.



Two types of elliptic curves:

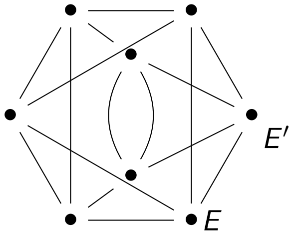
Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.



Two types of elliptic curves:

Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.

Supersingular curves $End(E)$ is a maximal order in the quaternion algebra $\mathbb{Q}_{p,\infty}$.

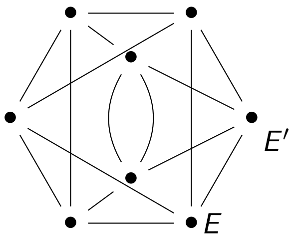


Two types of elliptic curves:

Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.

Supersingular curves $End(E)$ is a maximal order in the quaternion algebra $\mathbb{Q}_{p,\infty}$.

Supersingular curves can be defined over $\mathbb{F}_{p^2}$.



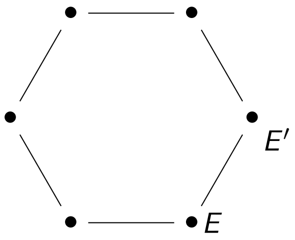
Two types of elliptic curves:

Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.

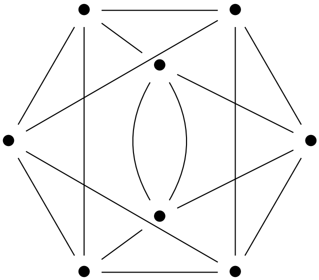
Supersingular curves $End(E)$ is a maximal order in the quaternion algebra $\mathbb{Q}_{p,\infty}$.

Supersingular curves can be defined over $\mathbb{F}_{p^2}$.

Looking only curves defined over $\mathbb{F}_p$, $\mathrm{End}_p(E)$ is an order in $\mathbb{Q}(\sqrt{-p})$.

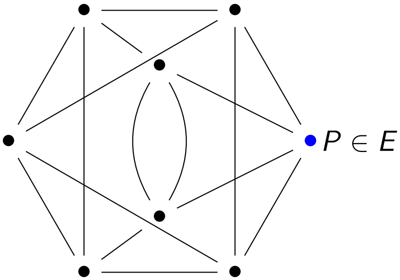


VDF over $\mathbb{F}_{p^2}$ supersingular curves.



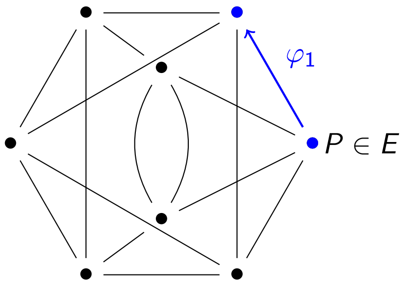
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.



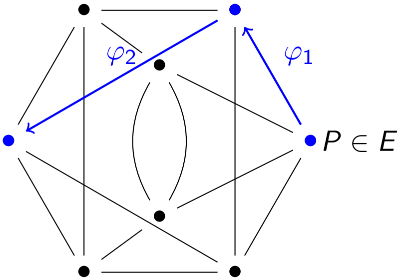
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.



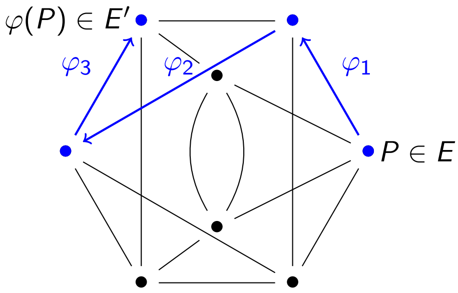
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.



VDF over $\mathbb{F}_{p^2}$ supersingular curves.

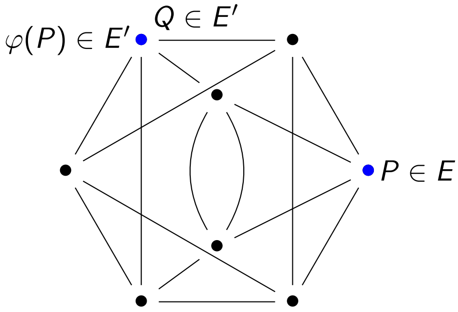
Setup A **public** walk in the isogeny graph.



VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.

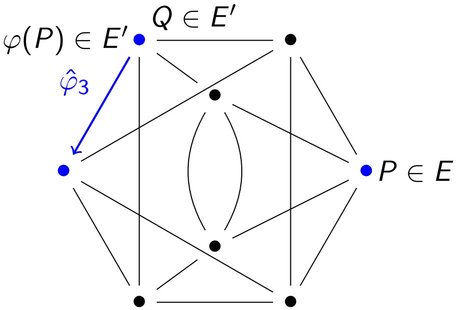
Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).



VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.

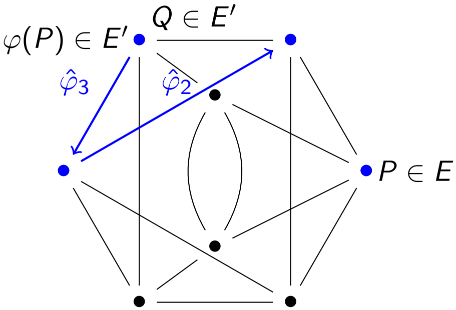
Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).



VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.

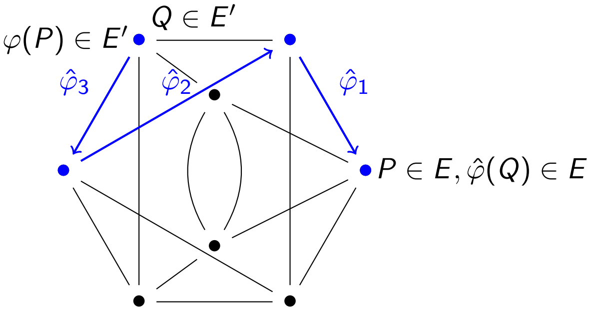
Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).



VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

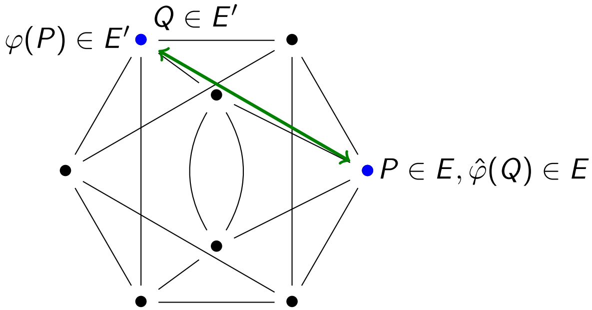


VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

Verification Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q)$.

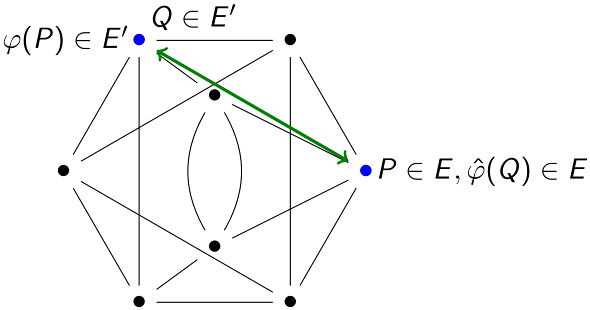


VDF over $\mathbb{F}_{p^2}$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

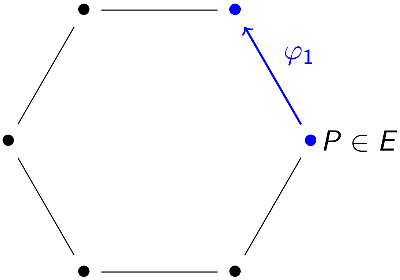
Verification Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q)$.



Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

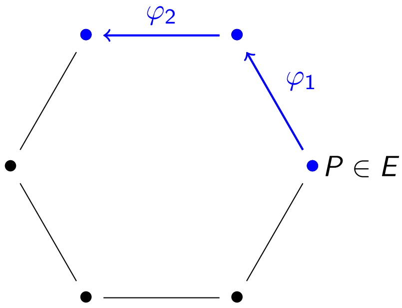
Setup A **public** walk in the isogeny graph.



Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

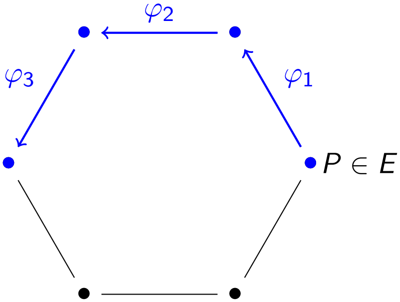
Setup A **public** walk in the isogeny graph.



Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

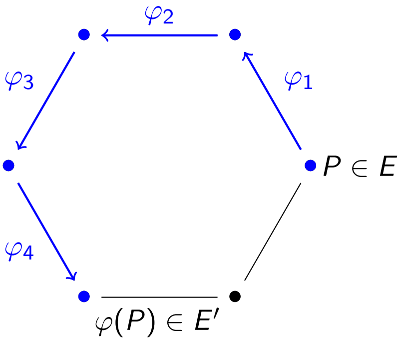
Setup A **public** walk in the isogeny graph.



Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

Setup A **public** walk in the isogeny graph.

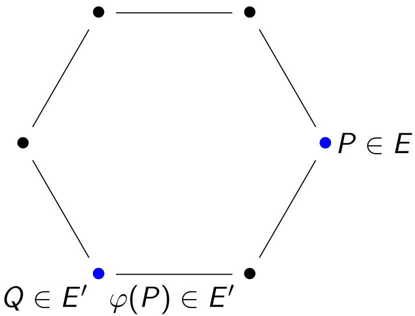


Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

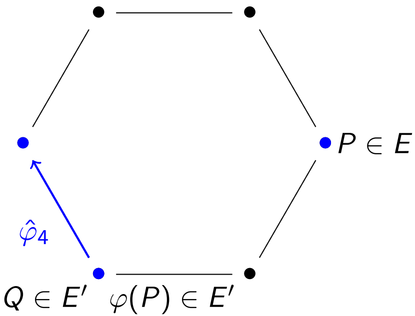


Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

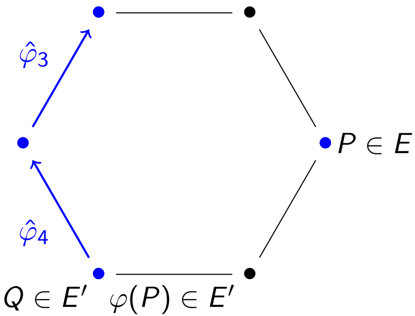


Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

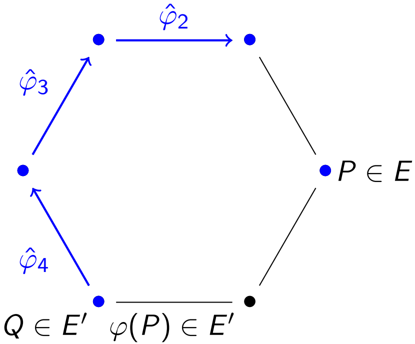


Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

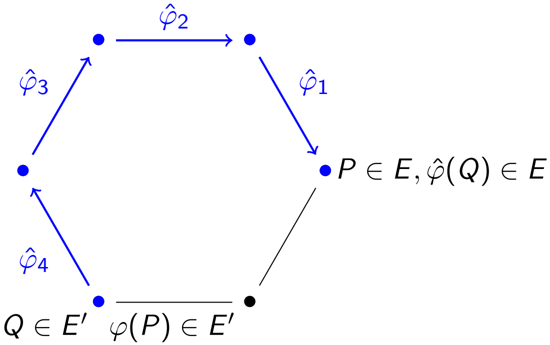


Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).



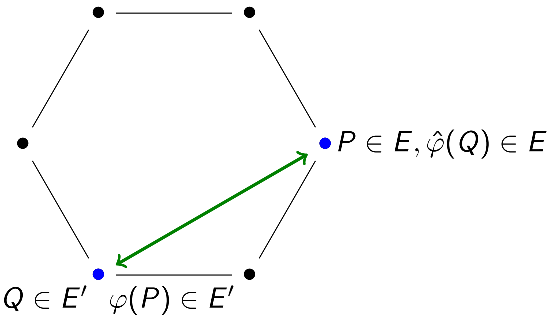
Not post-quantum, but also no proof of evaluation needed!

VDF over $\mathbb{F}_p$ supersingular curves.

Setup A **public** walk in the isogeny graph.

Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).

Verification Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q)$.

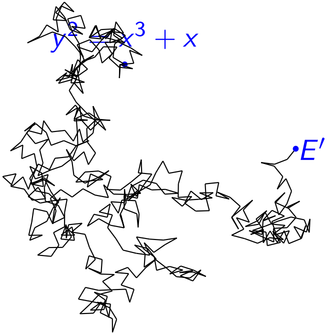


Not post-quantum, but also no proof of evaluation needed!

Isogeny shortcut.

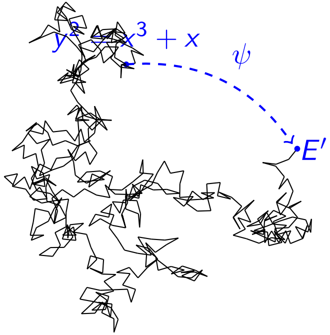
$$y^2 = x^3 + x$$

Isogeny shortcut.



Isogeny shortcut.

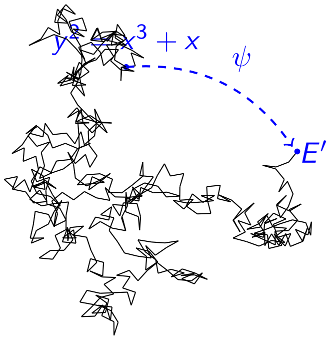
If E has a known endomorphism ring, a shortcut can be found!



Isogeny shortcut.
If E has a known endomorphism ring, a shortcut can be found!

Unknown endomorphism ring.

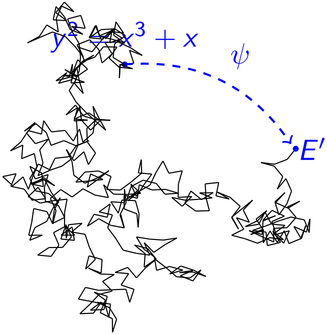
- Ordinary curves. Pairing friendly $\rightarrow$ small discriminant $\rightarrow$ known End(E).



Isogeny shortcut.
If E has a known endomorphism ring, a shortcut can be found!

Unknown endomorphism ring.

- Ordinary curves. Pairing friendly $\rightarrow$ small discriminant $\rightarrow$ known $\mathrm{End}(E)$.
- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.



Isogeny shortcut.
If E has a known endomorphism ring, a shortcut can be found!

Unknown endomorphism ring.

- Ordinary curves. Pairing friendly $\rightarrow$ small discriminant $\rightarrow$ known $\mathrm{End}(E)$.
- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.

Trusted setup.

Isogeny shortcut.
If E has a known endomorphism ring, a shortcut can be found!

Unknown endomorphism ring.

- Ordinary curves. Pairing friendly $\rightarrow$ small discriminant $\rightarrow$ known $\mathrm{End}(E)$.
- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.

Trusted setup.

- Start from a well known supersingular curve,

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

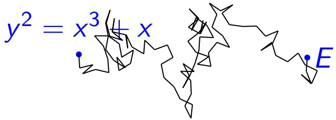
Isogeny shortcut.
If E has a known endomorphism ring, a shortcut can be found!

Unknown endomorphism ring.

- Ordinary curves. Pairing friendly $\to$ small discriminant $\to$ known $\mathrm{End}(E)$.
- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.

Trusted setup.

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



Isogeny shortcut.
If E has a known endomorphism ring, a shortcut can be found!

Unknown endomorphism ring.

- Ordinary curves. Pairing friendly $\rightarrow$ small discriminant $\rightarrow$ known $\mathrm{End}(E)$.
- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.

Trusted setup.

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

$\bullet E$

Isogeny shortcut.
If E has a known endomorphism ring, a shortcut can be found!

Unknown endomorphism ring.

- Ordinary curves. Pairing friendly $\rightarrow$ small discriminant $\rightarrow$ known $\mathrm{End}(E)$.
- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.

Trusted setup.

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

E has an unknown endomorphism ring.

$\bullet E$

Isogenies over $\mathbb{F}_p$ and class group.

$$E_1 \xrightarrow{\quad\varphi_1\quad} E_2$$

Isogenies over $\mathbb{F}_p$ and class group.

$$\begin{tikzcd}
E_1 \arrow[r, "\varphi_1"] \arrow[d] & E_2 \arrow[d] \\
\mathrm{End}_p(E_1) \arrow[r, "I_1"] & \mathrm{End}_p(E_2)
\end{tikzcd}$$

Isogenies over $\mathbb{F}_p$ and class group.

$$
\begin{tikzcd}
E_1 \arrow[r, "\varphi_1"] \arrow[d] & E_2 \arrow[r, "\varphi_2"] \arrow[d] & E_3 \arrow[d] \\
\mathrm{End}_p(E_1) \arrow[r, "I_1"] & \mathrm{End}_p(E_2) \arrow[r, "I_2"] & \mathrm{End}_p(E_3)
\end{tikzcd}
$$

Isogenies over $\mathbb{F}_p$ and class group.

$$\begin{tikzcd}
E_1 \arrow[r, "\varphi_2 \circ \varphi_1"] \arrow[d] & E_3 \arrow[d] \\
\mathrm{End}_p(E_1) \arrow[r, "I_1 I_2"] & \mathrm{End}_p(E_3)
\end{tikzcd}$$

Isogenies over $\mathbb{F}_p$ and class group.

$$\begin{tikzcd}
E_1 \arrow[rr, "\varphi_2 \circ \varphi_1"] \arrow[d] & & E_3 \arrow[d] \\
\mathrm{End}_p(E_1) \arrow[rr, "I_1 I_2"] & & \mathrm{End}_p(E_3)
\end{tikzcd}$$

Security of our VDF.

- Our VDFs are secure on a classical computer.
- The $\mathbb{F}_p$ VDF is insecure on a quantum computer:
  - Once the setup is done, compute $\#\mathrm{Cl}(D)$.
  - Evaluate the $\mathbb{F}_p$ VDF with ideal multiplications faster than isogenies.
- The $\mathbb{F}_{p^2}$ VDF is insecure on a quantum computer.
  It is *quantum-annoying* in the sense that you need to run Shor's algorithm for each evaluation of the VDF.

Implementation and comparison

Definition and examples

VDF based on isogenies and pairings

Implementation and comparison

Computing isogenies.

- Method 1. Degree 2 isogenies using 2-torsion points:

$$\underset{E}{\bullet} - \underset{E_1}{\bullet} - \underset{E_2}{\bullet} - \underset{E_3}{\bullet} - \underset{E_4}{\bullet} - \underset{E_5}{\bullet} - \underset{E_6}{\bullet} - \underset{E_7}{\bullet} - \underset{E_8}{\bullet} - \cdots - \bullet - \bullet - \bullet - \underset{E_{T-1}}{\bullet} - \underset{E'}{\bullet}$$

Computing isogenies.

- Method 1. Degree 2 isogenies using 2-torsion points:

$$\overset{E}{\bullet} - \overset{E_1}{\bullet} - \overset{E_2}{\bullet} - \overset{E_3}{\bullet} - \overset{E_4}{\bullet} - \overset{E_5}{\bullet} - \overset{E_6}{\bullet} - \overset{E_7}{\bullet} - \overset{E_8}{\bullet} - \cdots - \bullet - \bullet - \bullet - \overset{E_{T-1}}{\bullet} - \overset{E'}{\bullet}$$

  Time complexity: T isogenies of degree 2.
  Storage complexity: $O(T)$.

Computing isogenies.

- Method 1. Degree 2 isogenies using 2-torsion points:

$$\underset{\bullet}{E} - \underset{\bullet}{E_1} - \underset{\bullet}{E_2} - \underset{\bullet}{E_3} - \underset{\bullet}{E_4} - \underset{\bullet}{E_5} - \underset{\bullet}{E_6} - \underset{\bullet}{E_7} - \underset{\bullet}{E_8} - \cdots - \bullet - \bullet - \bullet - \underset{\bullet}{E_{T-1}} - \underset{\bullet}{E'}$$

  Time complexity: T isogenies of degree 2.
  Storage complexity: $O(T)$.

- Method 2. Degree 2^n isogenies using 2^n-torsion point:

$$\underset{\bullet}{E} \longrightarrow\!\!\!\longrightarrow \underset{\bullet}{E_4} \longrightarrow\!\!\!\longrightarrow \underset{\bullet}{E_8} - \cdots - \bullet \longrightarrow\!\!\!\longrightarrow \bullet$$

Computing isogenies.

- Method 1. Degree 2 isogenies using 2-torsion points:

$$E - E_1 - E_2 - E_3 - E_4 - E_5 - E_6 - E_7 - E_8 - \cdots - \bullet - \bullet - \bullet - E_{T-1} - E'$$

  Time complexity: T isogenies of degree 2.
  Storage complexity: $O(T)$.

- Method 2. Degree 2^n isogenies using 2^n-torsion point:

$$E \longrightarrow E_4 \longrightarrow E_8 - \cdots - \bullet \longrightarrow \bullet$$

  Time complexity: T/n isogenies of degree $2^n \approx T \log_2(n)$ degree 2 isogenies.
  Storage complexity: $O(T/n)$.
  In practice (for large T), $log_2(n)$ is small and it can be useful to reduce the storage.

Choice of parameters.

$$\#E(\mathbb{F}_p) = p + 1$$

Choice of parameters.

$$\#E(\mathbb{F}_p) = p + 1$$

- DLP over the curves.
  P and Q of order r with $\log_2(r) \approx 256$. so we set $p = fr - 1$ with f a cofactor.

bits of p : $\boxed{\quad r \quad}\boxed{f}$

Choice of parameters.

$$\#E(\mathbb{F}_p) = p + 1$$

- DLP over the curves.
  P and Q of order r with $\log_2(r) \approx 256$. so we set $p = fr - 1$ with f a cofactor.
- DLP over the finite field $\mathbb{F}_{p^2}$.
  NFS over $\mathbb{F}_{p^2}$: $\log_2(p) \approx 1500$. We need a cofactor of size $\log_2(f) \approx 1250$.

bits of p :

Choice of parameters.

$$\#E(\mathbb{F}_p) = p + 1$$

- DLP over the curves.
  P and Q of order r with $\log_2(r) \approx 256$. so we set $p = fr - 1$ with f a cofactor.
- DLP over the finite field $\mathbb{F}_{p^2}$.
  NFS over $\mathbb{F}_{p^2}$: $\log_2(p) \approx 1500$. We need a cofactor of size $\log_2(f) \approx 1250$.
- From a point of order 2^n, we can compute a degree 2^n isogeny in $O(n \log_2(n))$ degree 2 isogenies.

bits of p :

Choice of parameters.

$$\#E(\mathbb{F}_p) = p + 1$$

- DLP over the curves.
  P and Q of order r with $\log_2(r) \approx 256$. so we set $p = fr - 1$ with f a cofactor.
- DLP over the finite field $\mathbb{F}_{p^2}$.
  NFS over $\mathbb{F}_{p^2}$: $\log_2(p) \approx 1500$. We need a cofactor of size $\log_2(f) \approx 1250$.
- From a point of order 2^n, we can compute a degree 2^n isogeny in $O(n \log_2(n))$ degree 2 isogenies.

bits of p :

$$p = r \cdot 2^{1244} \cdot f - 1$$

Implementation.

- Proof of concept in SageMath : https://github.com/isogenies-vdf.
- Parameters chosen for 128 bits of security
- Arithmetic of Montgomery curves
- Isogeny computation with recursive strategy
- Tate pairing computation.

Implementation.

- Proof of concept in SageMath : https://github.com/isogenies-vdf.
- Parameters chosen for 128 bits of security
- Arithmetic of Montgomery curves
- Isogeny computation with recursive strategy
- Tate pairing computation.

Protocol	Step	e_k size	Time	Throughput
$\mathbb{F}_p$ graph	Setup	238 kb	–	0.75isog/ms
	Evaluation	–	–	0.75isog/ms
	Verification	–	0.3 s	–
$\mathbb{F}_{p^2}$ graph	Setup	491 kb	–	0.35isog/ms
	Evaluation	–	–	0.23isog/ms
	Verification	–	4 s	–

Table: Benchmarks for our VDFs, on a Intel Core i7-8700 @ 3.20GHz, $T \approx 2^{16}$

VDF	**pro**	**con**
RSA	fast verification	trusted setup
Class group	no trusted setup small parameters	slow verification
Isogenies over $\mathbb{F}_p$	Fast verification	trusted setup
Isogenies over $\mathbb{F}_{p^2}$	Quantum-annoying Fast verification	trusted setup

VDF	pro	con
RSA	fast verification	trusted setup
Class group	no trusted setup small parameters	slow verification
Isogenies over $\mathbb{F}_p$	Fast verification	trusted setup
Isogenies over $\mathbb{F}_{p^2}$	Quantum-annoying Fast verification	trusted setup

Open problems.

- Hash to the supersingular set (in order to remove the trusted setup)

VDF	pro	con
RSA	fast verification	trusted setup
Class group	no trusted setup small parameters	slow verification
Isogenies over $\mathbb{F}_p$	Fast verification	trusted setup
Isogenies over $\mathbb{F}_{p^2}$	Quantum-annoying Fast verification	trusted setup

Open problems.

- Hash to the supersingular set (in order to remove the trusted setup)
- Find a fully post-quantum VDF

Thank you for your attention.