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

[2] Thales and Université de Lorraine, Nancy

[3] University of Birmingham

[4] Adobe Inc. and Ruhr Universität Bochum

November 2nd, 2020

Definition and examples

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.

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

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

Need a *trusted setup* to choose N.

VDF based on RSA.

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

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

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

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

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

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

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	**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.

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.

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 E, we can choose $\mathbb{G}_1$ and $\mathbb{G}_2$ two groups of points of E, and $\mathbb{G}_3$ a multiplicative subgroup of a finite field.
A curve is *pairing-friendly* if the $\mathbb{G}_i$ are efficiently computable.

Pairing-friendly elliptic curves.

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 E, we can choose $\mathbb{G}_1$ and $\mathbb{G}_2$ two groups of points of E, and $\mathbb{G}_3$ a multiplicative subgroup of a finite field.

A curve is *pairing-friendly* if the $\mathbb{G}_i$ are efficiently computable.

Isogenies of elliptic curves.

An isogeny between two elliptic curves E and E' is an algebraic map

$$\varphi : E \longrightarrow E'$$

such that $\varphi(0_E) = 0_{E'}$.

Pairing-friendly elliptic curves.

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 E, we can choose $\mathbb{G}_1$ and $\mathbb{G}_2$ two groups of points of E, and $\mathbb{G}_3$ a multiplicative subgroup of a finite field.

A curve is *pairing-friendly* if the $\mathbb{G}_i$ are efficiently computable.

Isogenies of elliptic curves.

An isogeny between two elliptic curves E and E' is an algebraic map

$$\varphi : E \longrightarrow E'$$

such that $\varphi(0_E) = 0_{E'}$.

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

Pairing-friendly elliptic curves.

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 E, we can choose $\mathbb{G}_1$ and $\mathbb{G}_2$ two groups of points of E, and $\mathbb{G}_3$ a multiplicative subgroup of a finite field.
A curve is *pairing-friendly* if the $\mathbb{G}_i$ are efficiently computable.

Isogenies of elliptic curves.

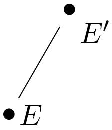
An isogeny between two elliptic curves E and E' is an algebraic map

$$\varphi : E \longrightarrow E'$$

such that $\varphi(0_E) = 0_{E'}$.
From $\varphi : E \to E'$, there always exists a dual isogeny $\hat{\varphi} : E' \to E$ such that $\varphi \circ \hat{\varphi} = [\deg \varphi]$. For $P \in \mathbb{G}_1$ on E and $Q \in \mathbb{G}_2$ on E',

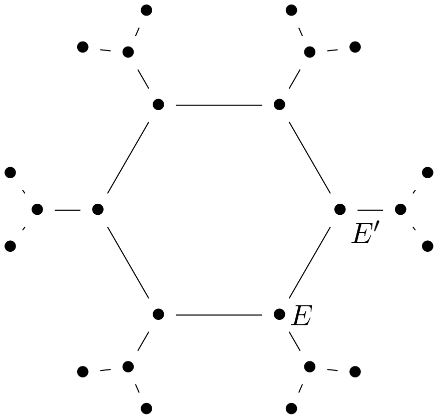
$$e(\varphi(P), Q) = e(P, \hat{\varphi}(Q))$$

Two types of elliptic curves:



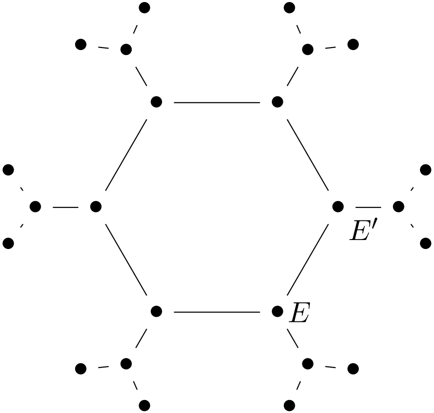
Two types of elliptic curves:

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



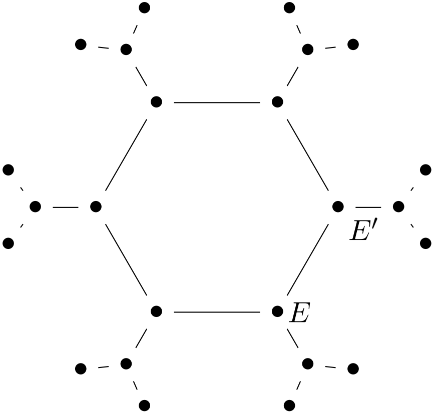
Two types of elliptic curves:

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



Two types of elliptic curves:

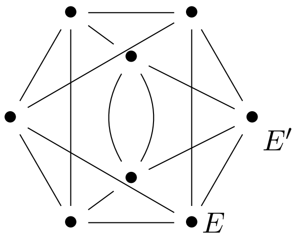
Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.



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 a quaternion algebra.

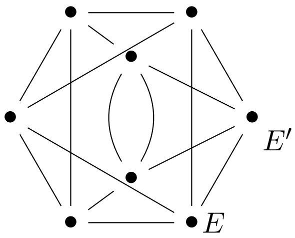


Two types of elliptic curves:

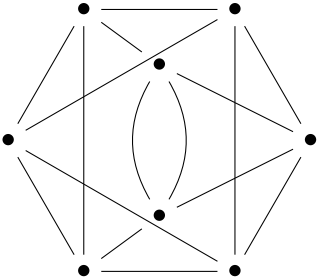
Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$.

Supersingular curves $End(E)$ is a maximal order in a quaternion algebra.

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

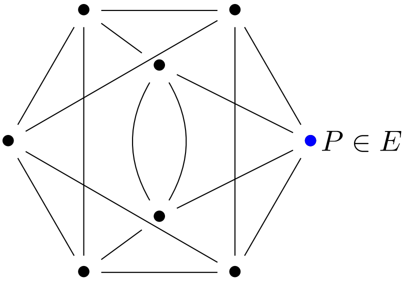


VDF over $\mathbb{F}_{p^2}$ supersingular curves.



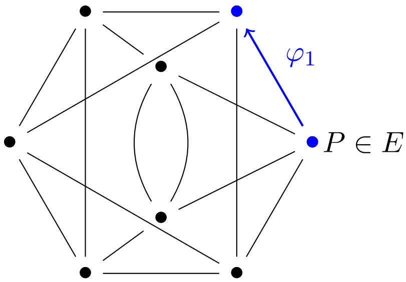
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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



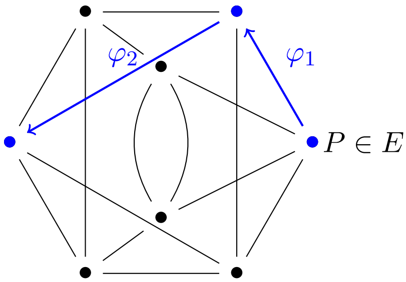
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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



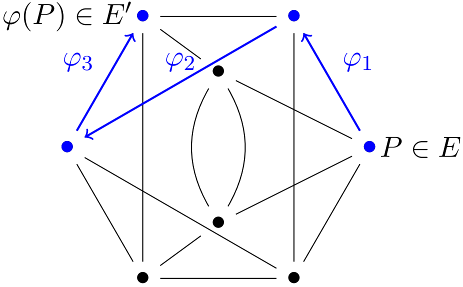
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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



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

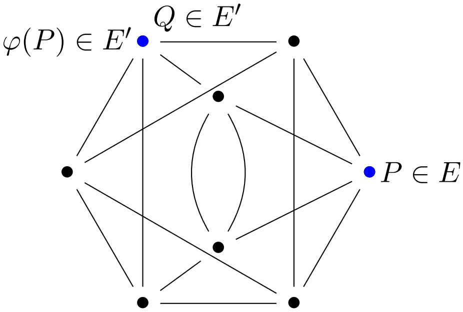
Setup A **public** walk in the isogeny graph.



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

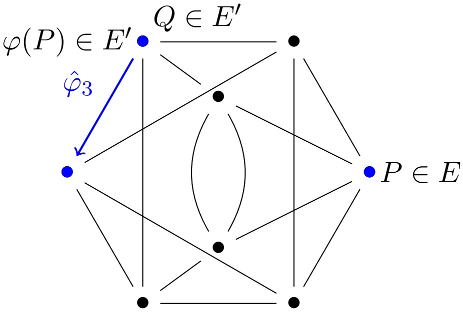
Setup A **public** walk in the isogeny graph.

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



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

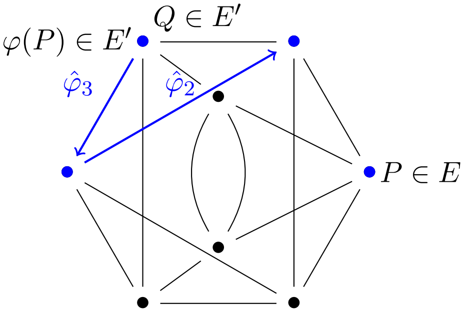
Setup	A **public** walk in the isogeny graph.
Evaluation	For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).



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

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

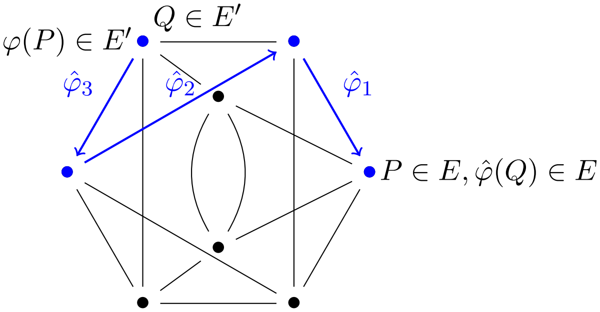
Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtracking walk).



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).

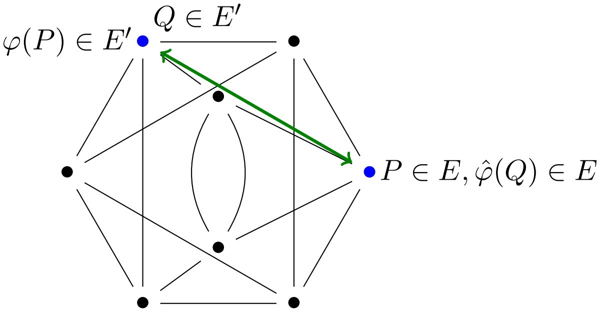


VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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

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

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

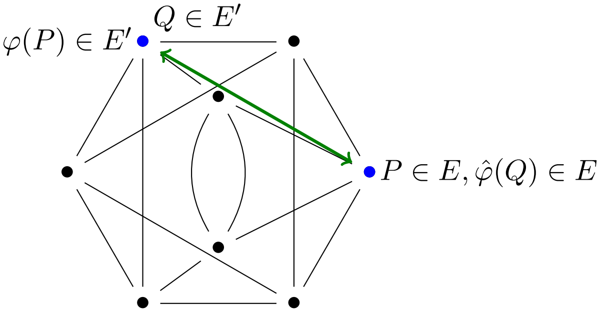


VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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

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

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



Another version with isogenies defined over $\mathbb{F}_p$ in the paper.

Security considerations

Definition and examples

VDF based on isogenies and pairings

Security considerations

Implementation and comparison

Security.
What means the VDF is secure ?

Security.
What means the VDF is secure ?
One cannot evaluate in less than T steps.

Security.

What means the VDF is secure ?

One cannot evaluate in less than T steps.

- Attacking the DLP in $\mathbb{F}_{p^2}$.
  Writing $\mathbb{G}_2 = \langle G \rangle$, find x such that $e(P, G)^x = e(\varphi(P), Q)$.
  Solution: choose a large prime p (1500 bits) such that DLP is hard in $\mathbb{F}_{p^2}$.

Security.

What means the VDF is secure ?

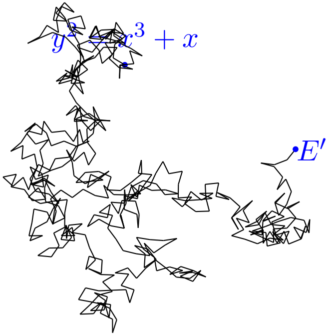
One cannot evaluate in less than T steps.

- Attacking the DLP in $\mathbb{F}_{p^2}$.
  Writing $\mathbb{G}_2 = \langle G \rangle$, find x such that $e(P, G)^x = e(\varphi(P), Q)$.
  Solution: choose a large prime p (1500 bits) such that DLP is hard in $\mathbb{F}_{p^2}$.
- Find a shortcut.
  Find a way to compute the isogeny in less than T steps.

Isogeny shortcut.

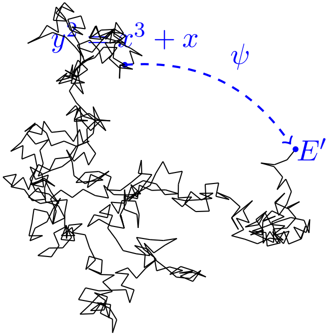
$$y^2 = x^3 + x$$

Isogeny shortcut.



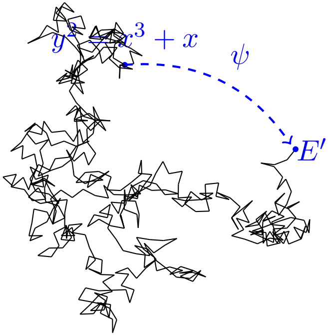
Isogeny shortcut.

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.

- Convert the isogeny into an ideal of $\mathrm{End}(E)$;
- Find an equivalent ideal J of different (smaller) norm;
- Convert J into another isogeny ψ of smaller degree.



Example. $p = 3 \bmod 4$.

$$\begin{tikzcd} y^2 = x^3 + x : \ E_0 \arrow[rr, bend left, "{\ker\varphi = \langle P \rangle, \deg\varphi = 2}"] & \varphi : (x,y) \mapsto \left(\frac{x^2-1}{x}, y\frac{x^2+1}{x^2}\right) & E \end{tikzcd}$$

Example. $p = 3 \bmod 4$.

$$\begin{tikzcd}[column sep=huge]
y^2 = x^3 + x : \; E_0 \arrow[rr, bend left, "\ker\varphi = \langle P\rangle, \deg\varphi = 2"] & \varphi : (x, y) \mapsto \left(\frac{x^2-1}{x}, y\frac{x^2+1}{x^2}\right) & E \\
\mathcal{O}_0 & & \mathcal{O}
\end{tikzcd}$$

Example. $p = 3 \bmod 4$.

$$\begin{tikzcd}
y^2 = x^3 + x :\ E_0 \arrow[rr, bend left, "\ker\varphi = \langle P\rangle, \deg\varphi = 2"] & \varphi : (x,y) \mapsto \left(\frac{x^2-1}{x}, y\frac{x^2+1}{x^2}\right) & E \\
\mathbb{Z}\langle \mathbf{1}, \mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}, \frac{\mathbf{i}+\mathbf{k}}{2}\rangle =\ \mathcal{O}_0 & & \mathcal{O}
\end{tikzcd}$$

Example. $p = 3 \bmod 4$.

$$\begin{tikzcd}
y^2 = x^3 + x :\ E_0 \arrow[rr, bend left, "{\ker\varphi = \langle P\rangle, \deg\varphi = 2}"] & \varphi : (x,y) \mapsto \left(\frac{x^2-1}{x}, y\frac{x^2+1}{x^2}\right) & E \\
\mathbb{Z}\langle \mathbf{1}, \mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}, \frac{\mathbf{i}+\mathbf{k}}{2}\rangle = \mathcal{O}_0 \arrow[rr, leftrightarrow, "{\mathcal{I} = \mathcal{O}_0 \cdot 2 + \mathcal{O}_0 \cdot \alpha}"'] & & \mathcal{O}
\end{tikzcd}$$

Example. $p = 3 \bmod 4$.

$$\begin{tikzcd}[column sep=huge]
y^2 = x^3 + x : \; E_0 \arrow[r, bend left, "{\ker\varphi = \langle P \rangle, \deg\varphi = 2}", "{\varphi : (x,y) \mapsto \left(\frac{x^2-1}{x}, y\frac{x^2+1}{x^2}\right)}"'] & E \\
\mathbb{Z}\langle \mathbf{1}, \mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}, \frac{\mathbf{i}+\mathbf{k}}{2} \rangle = \; \mathcal{O}_0 \arrow[r, leftrightarrow, "{\mathcal{I} = \mathcal{O}_0 \cdot 2 + \mathcal{O}_0 \cdot \alpha}"'] & \mathcal{O}
\end{tikzcd}$$

The endomorphism α can be written $\alpha = n_1\mathbf{1} + n_2\mathbf{i} + n_3\frac{\mathbf{1}+\mathbf{j}}{2} + n_4\frac{\mathbf{i}+\mathbf{k}}{2}$.

Example. $p = 3 \bmod 4$.

$$
\begin{tikzcd}
y^2 = x^3 + x : \; E_0 \arrow[rr, bend left, "{\ker\varphi = \langle P\rangle, \deg\varphi = 2}"] & \varphi : (x,y) \mapsto \left(\frac{x^2-1}{x}, y\frac{x^2+1}{x^2}\right) & E \\
\mathbb{Z}\langle \mathbf{1}, \mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}, \frac{\mathbf{i}+\mathbf{k}}{2}\rangle = \mathcal{O}_0 \arrow[rr, leftrightarrow, "{\mathcal{I} = \mathcal{O}_0 \cdot 2 + \mathcal{O}_0 \cdot \alpha}"'] & & \mathcal{O}
\end{tikzcd}
$$

The endomorphism α can be written $\alpha = n_1\mathbf{1} + n_2\mathbf{i} + n_3\frac{\mathbf{1}+\mathbf{j}}{2} + n_4\frac{\mathbf{i}+\mathbf{k}}{2}$.
We solve

$$n_1\mathbf{1}(P) + n_2\mathbf{i}(P) + n_3\left(\frac{\mathbf{1}+\mathbf{j}}{2}\right)(P) + n_4\left(\frac{\mathbf{i}+\mathbf{k}}{2}\right)(P) = 0_{E_0}.$$

Example. $p = 3 \bmod 4$.

$$\begin{tikzcd}
y^2 = x^3 + x : \ E_0 \arrow[rr, bend left, "{\ker\varphi = \langle P\rangle, \deg\varphi = 2}", "{\varphi : (x,y) \mapsto \left(\frac{x^2-1}{x}, y\frac{x^2+1}{x^2}\right)}"'] & & E \\
\mathbb{Z}\langle \mathbf{1}, \mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}, \frac{\mathbf{i}+\mathbf{k}}{2} \rangle = \ \mathcal{O}_0 \arrow[rr, leftrightarrow, "{\mathcal{I} = \mathcal{O}_0 \cdot 2 + \mathcal{O}_0 \cdot \alpha}"'] & & \mathcal{O}
\end{tikzcd}$$

The endomorphism α can be written $\alpha = n_1\mathbf{1} + n_2\mathbf{i} + n_3\frac{\mathbf{1}+\mathbf{j}}{2} + n_4\frac{\mathbf{i}+\mathbf{k}}{2}$. We solve

$$n_1\mathbf{1}(P) + n_2\mathbf{i}(P) + n_3\left(\frac{\mathbf{1}+\mathbf{j}}{2}\right)(P) + n_4\left(\frac{\mathbf{i}+\mathbf{k}}{2}\right)(P) = 0_{E_0}.$$

$\alpha = u_1 + u_3 + u_4 = \frac{3+\mathbf{i}+\mathbf{j}+\mathbf{k}}{2}$ and

$$\mathcal{I} = \mathcal{O}_0 \cdot 2 + \mathcal{O}_0 \cdot \alpha = \mathbb{Z}\left\langle \frac{\mathbf{1}+\mathbf{i}+\mathbf{j}+3\mathbf{k}}{2}, \mathbf{i}+\mathbf{k}, \mathbf{j}+\mathbf{k}, 2\mathbf{k} \right\rangle.$$

$$\begin{tikzcd}
E_0 \arrow[rr, bend left, "\varphi"'] & & E \\
\mathcal{O}_0 \arrow[rr, leftrightarrow, "\mathcal{I}"'] & & \mathcal{O}
\end{tikzcd}$$

Algorithms based on quaternion computations provide an equivalent ideal $J = I\beta$ of norm coprime to 2.

$$\begin{tikzcd}
E_0 \arrow[rr, bend left, "\varphi"'] & & E \\
\mathcal{O}_0 \arrow[rr, leftrightarrow, "\mathcal{I}"'] \arrow[rr, leftrightarrow, bend right, "\mathcal{J}"'] & & \mathcal{O}
\end{tikzcd}$$

Algorithms based on quaternion computations provide an equivalent ideal $J = I\beta$ of norm coprime to 2.
It corresponds to an isogeny of degree $N(J)$.

$$
\begin{tikzcd}[column sep=huge]
E_0 \arrow[r, bend left, "\varphi"] \arrow[r, bend right=10, "\tilde{\varphi}"'] & E \\
\mathcal{O}_0 \arrow[r, leftrightarrow, "\mathcal{I}"'] \arrow[r, leftrightarrow, bend right, "\mathcal{J}"'] & \mathcal{O}
\end{tikzcd}
$$

Algorithms based on quaternion computations provide an equivalent ideal $J = I\beta$ of norm coprime to 2.
It corresponds to an isogeny of degree $N(J)$.

$$
\begin{tikzcd}
E_0 \arrow[r, bend left, "\varphi"] \arrow[r, bend right=10, "\tilde{\varphi}"'] & E
\end{tikzcd}
$$

$$
\begin{tikzcd}
\mathcal{O}_0 \arrow[r, leftrightarrow, "\mathcal{I}"'] \arrow[r, leftrightarrow, bend right, "\mathcal{J}"'] & \mathcal{O}
\end{tikzcd}
$$

Implementation in Magma
https://gitlab.inria.fr/smasson/endomorphismsthroughisogenies.

Algorithms based on quaternion computations provide an equivalent ideal $J = I\beta$ of norm coprime to 2.
It corresponds to an isogeny of degree $N(J)$.

$$\begin{tikzcd}
E_0 \arrow[r, bend left, "\varphi"] \arrow[r, bend right, "\tilde{\varphi}"'] & E
\end{tikzcd}$$

$$\begin{tikzcd}
\mathcal{O}_0 \arrow[r, leftrightarrow, "\mathcal{I}"'] \arrow[r, leftrightarrow, bend right, "\mathcal{J}"'] & \mathcal{O}
\end{tikzcd}$$

Implementation in Magma
https://gitlab.inria.fr/smasson/endomorphismsthroughisogenies.
Conclusion: **do not use a curve with a known endomorphism ring!**

Do we known curves with an unknown endomorphism ring?

Do we known curves with an unknown endomorphism ring?

Pairing-friendly
ordinary curves
no

Do we known curves with an unknown endomorphism ring?

Pairing-friendly ordinary curves	Special supersingular curves
no	**no**

Do we known curves with an unknown endomorphism ring?

Pairing-friendly ordinary curves	Special supersingular curves
no	**no**

Trusted setup (supersingular case).

Do we known curves with an unknown endomorphism ring?

Pairing-friendly ordinary curves	Special supersingular curves
no	**no**

$E_0 \bullet$

Trusted setup (supersingular case)**.**

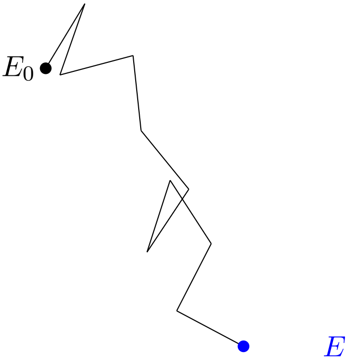
- Start from a well known supersingular curve,

Do we known curves with an unknown endomorphism ring?

Pairing-friendly ordinary curves	Special supersingular curves
no	**no**

Trusted setup (supersingular case)**.**

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



Do we known curves with an unknown endomorphism ring?

Pairing-friendly ordinary curves	Special supersingular curves
no	**no**

Trusted setup (supersingular case).

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

Do we known curves with an unknown endomorphism ring?

Pairing-friendly ordinary curves	Special supersingular curves
no	**no**

Trusted setup (supersingular case).

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

E has an unknown endomorphism ring.

Implementation and comparison

Definition and examples

VDF based on isogenies and pairings

Security considerations

Implementation and comparison

Implementation of the VDF.

Implementation of the VDF.

- Proof of concept in SageMath : https://github.com/isogenies-vdf.

Implementation of the VDF.

- Proof of concept in SageMath : https://github.com/isogenies-vdf.
- Parameters chosen for 128 bits of security

Implementation of the VDF.

- Proof of concept in SageMath : https://github.com/isogenies-vdf.
- Parameters chosen for 128 bits of security
- Arithmetic of Montgomery curves

Implementation of the VDF.

- 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

Implementation of the VDF.

- 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 of the VDF.

- 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.

	Step	e_k size	Time	Throughput
$\mathbb{F}_p$ graph	Setup	238 kb	–	0.75isog/ms
	Evaluation	–	–	0.75isog/ms
	Verification	–	0.3 s	–
$\mathbb{F}_{p^2}$ VDF	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, $T \approx 2^{16}$

Comparison.

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 long setup
Isogenies over $\mathbb{F}_{p^2}$	Quantum-annoying Fast verification	trusted setup long setup

Comparison.

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 long setup
Isogenies over $\mathbb{F}_{p^2}$	Quantum-annoying Fast verification	trusted setup long setup

Open problems.

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

Comparison.

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 long setup
Isogenies over $\mathbb{F}_{p^2}$	Quantum-annoying Fast verification	trusted setup long setup

Open problems.

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

Comparison.

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 long setup
Isogenies over $\mathbb{F}_{p^2}$	Quantum-annoying Fast verification	trusted setup long 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.