Verifiable delay functions from elliptic curve cryptography

Simon Masson
Joint work with L. De Feo, C. Petit and A. Sanso

Thales – LORIA

July 4th, 2019

Definition

A *verifiable delay function* (VDF) is a function that

1. takes T steps to evaluate, even with unbounded parallelism
2. the output can be verified efficiently.

Definition

A *verifiable delay function* (VDF) is a function that

1. takes T steps to evaluate, even with unbounded parallelism
2. the output can be verified efficiently.

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

Definition

A *verifiable delay function* (VDF) is a function that

1. takes T steps to evaluate, even with unbounded parallelism
2. the output can be verified efficiently.

- $\mathrm{Setup}(\lambda, T) \longrightarrow$ public parameters pp
- $\mathrm{Eval}(pp, x) \longrightarrow$ output y, proof π (requires T steps)
- $\mathrm{Verify}(pp, x, y, \pi) \longrightarrow$ yes or no.

Uniqueness If $\mathrm{Verify}(pp, x, y, \pi) = \mathrm{Verify}(pp, x, y', \pi') =$ yes, then $y = y'$.

Correctness The verification will always succeed if Eval has been computed honestly.

Soundness A lying evaluator will always fail the verification.

Sequentiality It is impossible to correctly evaluate the VDF in time less than $T - o(T)$, even when using $\mathrm{poly}(T)$ parallel processors.

Application. How to generate randomness in the real world ?

Application. How to generate randomness in the real world ?

Fail1 *from a physical value.*

Application. How to generate randomness in the real world ?

Fail1 *from a physical value.*

Application. How to generate randomness in the real world ?

Fail1 *from a physical value.*

Application. How to generate randomness in the real world ?

Fail1 *from a physical value.*

Fail2 *Distributed generation.*

Application. How to generate randomness in the real world ?

Fail1 *from a physical value.*

Fail2 *Distributed generation.*

$r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$ seems random...

Application. How to generate randomness in the real world ?

Fail1 *from a physical value.*

Fail2 *Distributed generation.*

$r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$ seems random... but Eve controls the randomness !

Application. How to generate randomness in the real world ?

Fail1 *from a physical value.*

Fail2 *Distributed generation.*

$r_a \oplus r_b \oplus r_c \oplus r_d \oplus r_e$ seems random... but Eve controls the randomness !

Idea: slow things down by adding delay.

- VDF without "delay": public-key cryptography.

- VDF without "delay": public-key cryptography.

$$\forall x \in \langle g \rangle, \qquad f(x) = \log_g(x)$$

Verification is easy: $g^{f(x)} \stackrel{?}{=} x$.
You can parallelize to compute $f(x)$.

- VDF without "delay": public-key cryptography.

$$\forall x \in \langle g \rangle, \qquad f(x) = \log_g(x)$$

  Verification is easy: $g^{f(x)} \stackrel{?}{=} x$.
  You can parallelize to compute $f(x)$.
- VDF without "verifiability": composition of hash functions.

- VDF without "delay": public-key cryptography.

$$\forall x \in \langle g \rangle, \qquad f(x) = \log_g(x)$$

Verification is easy: $g^{f(x)} \stackrel{?}{=} x$.
You can parallelize to compute $f(x)$.

- VDF without "verifiability": composition of hash functions.

$$f(x) = h^{(T)}(x)$$

You need to recompute $f(x)$ to verify.

- VDF without "delay": public-key cryptography.

$$\forall x \in \langle g \rangle, \qquad f(x) = \log_g(x)$$

  Verification is easy: $g^{f(x)} \stackrel{?}{=} x$.
  You can parallelize to compute $f(x)$.

- VDF without "verifiability": composition of hash functions.

$$f(x) = h^{(T)}(x)$$

  You need to recompute $f(x)$ to verify.

- VDF without "no parallelization": pre-image of a hash function.

- VDF without "delay": public-key cryptography.

$$\forall x \in \langle g \rangle, \qquad f(x) = \log_g(x)$$

  Verification is easy: $g^{f(x)} \stackrel{?}{=} x$.
  You can parallelize to compute $f(x)$.
- VDF without "verifiability": composition of hash functions.

$$f(x) = h^{(T)}(x)$$

  You need to recompute $f(x)$ to verify.
- VDF without "no parallelization": pre-image of a hash function.

$$f(x) = h^{-1}(x)$$

  Verification is easy: $h(f(x)) \stackrel{?}{=} x$.
  Computation is faster as long as you parallelize.

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

Evaluation. $y = H(x)^{2^T} \mod N$, and π a proof.

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

Evaluation. $y = H(x)^{2^T} \mod N$, and π a proof.

Verification. Proof of a correct exponentiation.

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

Evaluation. $y = H(x)^{2^T} \mod N$, and π a proof.

Verification. Proof of a correct exponentiation.

Wesolowski proof.

- Verifier challenges with a small prime ℓ

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

Evaluation. $y = H(x)^{2^T} \mod N$, and π a proof.

Verification. Proof of a correct exponentiation.

Wesolowski proof.

- Verifier challenges with a small prime ℓ
- Evaluator computes q, r such that $2^T = q\ell + r$ and send $\pi = H(x)^q$.

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

Evaluation. $y = H(x)^{2^T} \mod N$, and π a proof.

Verification. Proof of a correct exponentiation.

Wesolowski proof.

- Verifier challenges with a small prime ℓ
- Evaluator computes q, r such that $2^T = q\ell + r$ and send $\pi = H(x)^q$.
- Verifier checks $y \stackrel{?}{=} \pi^\ell \cdot H(x)^r$.

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

Evaluation. $y = H(x)^{2^T} \mod N$, and π a proof.

Verification. Proof of a correct exponentiation.

Wesolowski proof.

- Verifier challenges with a small prime ℓ
- Evaluator computes q, r such that $2^T = q\ell + r$ and send $\pi = H(x)^q$.
- Verifier checks $y \stackrel{?}{=} \pi^\ell \cdot H(x)^r$.

Turned to non-interactive using Fiat-Shamir

π is short

Verification is fast.

VDF based on RSA.

Setup. N is a RSA modulus, public parameters: $(\mathbb{Z}/N\mathbb{Z}, H : \{0,1\}^* \to \mathbb{Z}/N\mathbb{Z})$.

Evaluation. $y = H(x)^{2^T} \mod N$, and π a proof.

Verification. Proof of a correct exponentiation.

Wesolowski proof.

- Verifier challenges with a small prime ℓ
- Evaluator computes q, r such that $2^T = q\ell + r$ and send $\pi = H(x)^q$.
- Verifier checks $y \stackrel{?}{=} \pi^\ell \cdot H(x)^r$.

Turned to non-interactive using Fiat-Shamir

π is short

Verification is fast.

- 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.

- If one can compute a root $\bmod N$, the VDF is **unsound**:
  Choose w and compute $\sqrt[\ell]{w}$. (y, π) and $(wy, \sqrt[\ell]{w}\pi)$ are two correct outputs !

- If one can compute a root $\bmod N$, the VDF is **unsound**:
  Choose w and compute $\sqrt[\ell]{w}$. (y, π) and $(wy, \sqrt[\ell]{w}\pi)$ are two correct outputs !
- We need the assumption that computing a root is hard. This holds in a RSA setup, as well as in another group of unknown order.

- If one can compute a root $\bmod N$, the VDF is **unsound**:
  Choose w and compute $\sqrt[\ell]{w}$. (y, π) and $(wy, \sqrt[\ell]{w}\pi)$ are two correct outputs !
- We need the assumption that computing a root is hard. This holds in a RSA setup, as well as in another group of unknown order.
- It works in 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 $\#\mathrm{ClassGroup}(D)$.

- If one can compute a root $\bmod N$, the VDF is **unsound**:
  Choose w and compute $\sqrt[\ell]{w}$. (y, π) and $(wy, \sqrt[\ell]{w}\pi)$ are two correct outputs !
- We need the assumption that computing a root is hard. This holds in a RSA setup, as well as in another group of unknown order.
- It works in 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 $\#\mathrm{ClassGroup}(D)$.
- It is not post-quantum...

Let E be an elliptic curve defined over $\mathbb{F}_p$.
Suppose that we have N a large prime integer and k a small integer such that

- $N \mid \#E(\mathbb{F}_p)$
- All the N-torsion points are defined over $\mathbb{F}_{p^k}$.

Let E be an elliptic curve defined over $\mathbb{F}_p$.
Suppose that we have N a large prime integer and k a small integer such that

- $N \mid \#E(\mathbb{F}_p)$
- All the N-torsion points are defined over $\mathbb{F}_{p^k}$.

The N-torsion points is a dimension 2 vector space $\mathbb{G}_1 \times \mathbb{G}_2$ where $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^k})$.

Let E be an elliptic curve defined over $\mathbb{F}_p$.
Suppose that we have N a large prime integer and k a small integer such that

- $N \mid \#E(\mathbb{F}_p)$
- All the N-torsion points are defined over $\mathbb{F}_{p^k}$.

The N-torsion points is a dimension 2 vector space $\mathbb{G}_1 \times \mathbb{G}_2$ where $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^k})$.

Definition

A pairing on E is a bilinear non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_{p^k}^\times$

Let E be an elliptic curve defined over $\mathbb{F}_p$.
Suppose that we have N a large prime integer and k a small integer such that

- $N \mid \#E(\mathbb{F}_p)$
- All the N-torsion points are defined over $\mathbb{F}_{p^k}$.

The N-torsion points is a dimension 2 vector space $\mathbb{G}_1 \times \mathbb{G}_2$ where $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^k})$.

Definition

A pairing on E is a bilinear non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_{p^k}^\times$

Application. The BLS signature.
Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

Let E be an elliptic curve defined over $\mathbb{F}_p$.
Suppose that we have N a large prime integer and k a small integer such that

- $N \mid \#E(\mathbb{F}_p)$
- All the N-torsion points are defined over $\mathbb{F}_{p^k}$.

The N-torsion points is a dimension 2 vector space $\mathbb{G}_1 \times \mathbb{G}_2$ where $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^k})$.

Definition

A pairing on E is a bilinear non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_{p^k}^\times$

Application. The BLS signature.
Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: s an integer
- Public key: $P_K = [s]P$.

Let E be an elliptic curve defined over $\mathbb{F}_p$.
Suppose that we have N a large prime integer and k a small integer such that

- $N \mid \#E(\mathbb{F}_p)$
- All the N-torsion points are defined over $\mathbb{F}_{p^k}$.

The N-torsion points is a dimension 2 vector space $\mathbb{G}_1 \times \mathbb{G}_2$ where $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^k})$.

Definition

A pairing on E is a bilinear non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_{p^k}^\times$

Application. The BLS signature.
Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: s an integer
- Public key: $P_K = [s]P$.

Sign Hash the message m into $\mathbb{G}_2$ and the signature is $\sigma = [s]H(m)$.

Verify Check that $e(P, \sigma) = e(P_K, H(m))$.

Let E be an elliptic curve defined over $\mathbb{F}_p$.
Suppose that we have N a large prime integer and k a small integer such that

- $N \mid \#E(\mathbb{F}_p)$
- All the N-torsion points are defined over $\mathbb{F}_{p^k}$.

The N-torsion points is a dimension 2 vector space $\mathbb{G}_1 \times \mathbb{G}_2$ where $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{p^k})$.

Definition

A pairing on E is a bilinear non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{F}_{p^k}^{\times}$

Application. The BLS signature.
Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: s an integer
- Public key: $P_K = [s]P$.

Sign Hash the message m into $\mathbb{G}_2$ and the signature is $\sigma = [s]H(m)$.

Verify Check that $e(P, \sigma) = e(P_K, H(m))$.
$e(P, \sigma) = e(P, [s]H(m)) = e([s]P, H(m)) = e(P_K, H(m))$.

Definition (Isogeny)

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

Definition (Isogeny)

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

Example (Frobenius)

For $A, B \in \bar{\mathbb{F}}_p$,

$$\begin{array}{rrcl} \pi_p : & E : y^2 = x^3 + Ax + B & \longrightarrow & E^{(p)} : y^2 = x^3 + A^p x + B^p \\ & (x, y) & \longmapsto & (x^p, y^p) \end{array}$$

Definition (Isogeny)

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

Example (Frobenius)

For $A, B \in \bar{\mathbb{F}}_p$,

$$\begin{array}{rrcl} \pi_p : & E : y^2 = x^3 + Ax + B & \longrightarrow & E^{(p)} : y^2 = x^3 + A^p x + B^p \\ & (x, y) & \longmapsto & (x^p, y^p) \end{array}$$

Vélu's formulas. For $P \in E(\bar{\mathbb{F}}_p)$ of order ℓ coprime with p, we have formulas for computing an isogeny φ of kernel $\langle P \rangle$. The degrees of the polynomials defining φ is $O(\ell)$.

Definition (Isogeny)

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

Example (Frobenius)

For $A, B \in \bar{\mathbb{F}}_p$,

$$\begin{array}{rrcl} \pi_p : & E : y^2 = x^3 + Ax + B & \longrightarrow & E^{(p)} : y^2 = x^3 + A^p x + B^p \\ & (x, y) & \longmapsto & (x^p, y^p) \end{array}$$

Vélu's formulas. For $P \in E(\bar{\mathbb{F}}_p)$ of order ℓ coprime with p, we have formulas for computing an isogeny φ of kernel $\langle P \rangle$. The degrees of the polynomials defining φ is $O(\ell)$.

In practice, Vélu's formulas are efficient for very small kernel.

Definition (Isogeny)

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

Example (Frobenius)

For $A, B \in \bar{\mathbb{F}}_p$,

$$\begin{array}{rrcl} \pi_p : & E : y^2 = x^3 + Ax + B & \longrightarrow & E^{(p)} : y^2 = x^3 + A^p x + B^p \\ & (x, y) & \longmapsto & (x^p, y^p) \end{array}$$

Vélu's formulas. For $P \in E(\bar{\mathbb{F}}_p)$ of order ℓ coprime with p, we have formulas for computing an isogeny φ of kernel $\langle P \rangle$. The degrees of the polynomials defining φ is $O(\ell)$.
In practice, Vélu's formulas are efficient for very small kernel.
From $\varphi : E \to E'$, there exists $\hat{\varphi} : E' \to E$ such that $\varphi \circ \hat{\varphi} = \hat{\varphi} \circ \varphi = [\deg \varphi]$.

Definition (Isogeny)

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

Example (Frobenius)

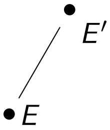
For $A, B \in \bar{\mathbb{F}}_p$,

$$\begin{array}{rrcl} \pi_p : & E : y^2 = x^3 + Ax + B & \longrightarrow & E^{(p)} : y^2 = x^3 + A^p x + B^p \\ & (x, y) & \longmapsto & (x^p, y^p) \end{array}$$

Vélu's formulas. For $P \in E(\bar{\mathbb{F}}_p)$ of order ℓ coprime with p, we have formulas for computing an isogeny φ of kernel $\langle P \rangle$. The degrees of the polynomials defining φ is $O(\ell)$.
In practice, Vélu's formulas are efficient for very small kernel.
From $\varphi : E \to E'$, there exists $\hat{\varphi} : E' \to E$ such that $\varphi \circ \hat{\varphi} = \hat{\varphi} \circ \varphi = [\deg \varphi]$.

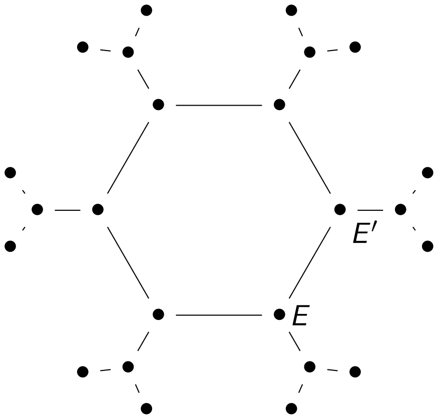
$$e(\varphi(P), \varphi(Q)) = e(P, Q)^{\deg(\varphi)}$$

Two types of elliptic curves:



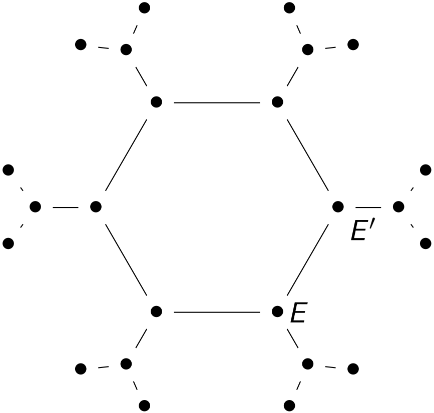
Two types of elliptic curves:

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



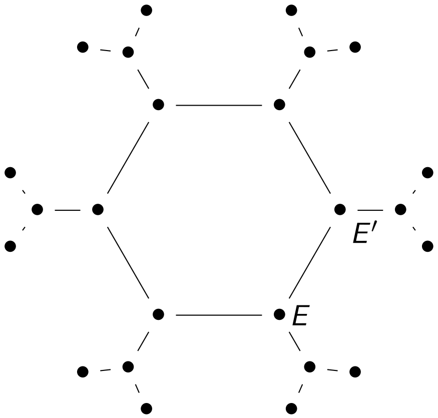
Two types of elliptic curves:

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



Two types of elliptic curves:

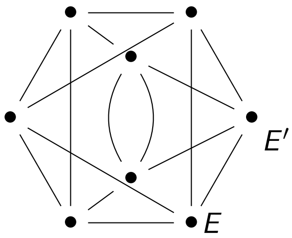
Ordinary curves $End(E)$ is an order in $\mathbb{Q}(\sqrt{-D})$. Isogeny graph is a volcano.



Two types of elliptic curves:

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

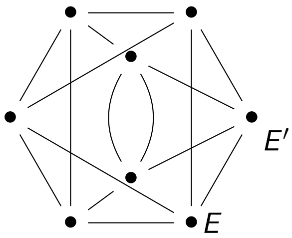
Supersingular curves $End(E)$ is a maximal order in the quaternion algebra $\mathbb{Q}_{p,\infty}$. Isogeny graph is expander.



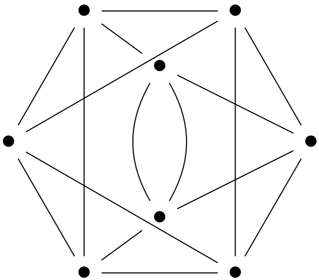
Two types of elliptic curves:

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

Supersingular curves $End(E)$ is a maximal order in the quaternion algebra $\mathbb{Q}_{p,\infty}$. Isogeny graph is expander. Supersingular curves are defined over $\mathbb{F}_{p^2}$.

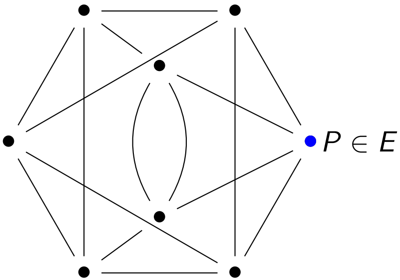


VDF over $\mathbb{F}_{p^2}$ supersingular curves.



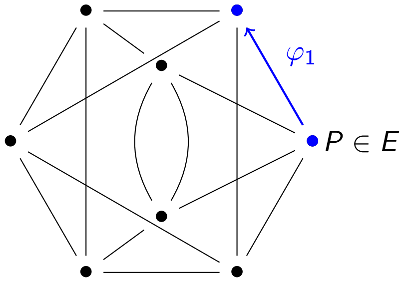
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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



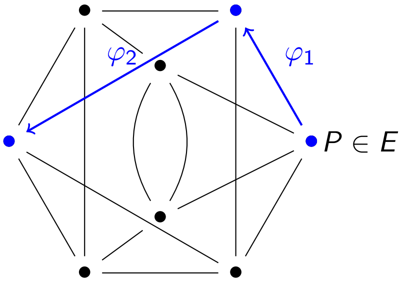
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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



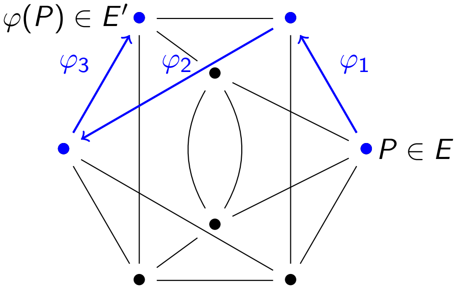
VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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



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

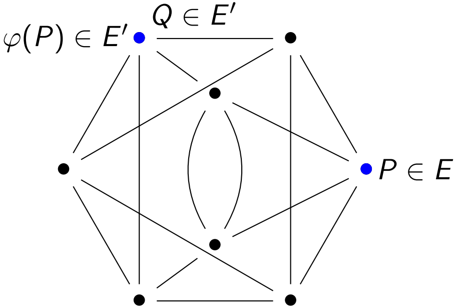
Setup A **public** walk in the isogeny graph.



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

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

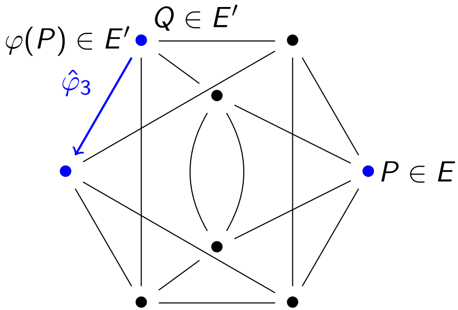
Evaluation: For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtrack walk).



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

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

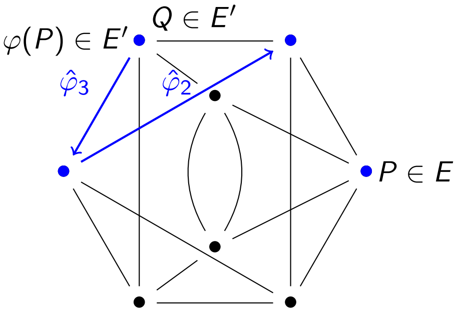
Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtrack walk).



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

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

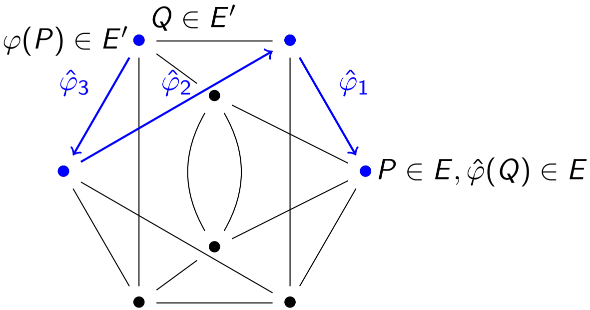
Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtrack walk).



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 backtrack walk).

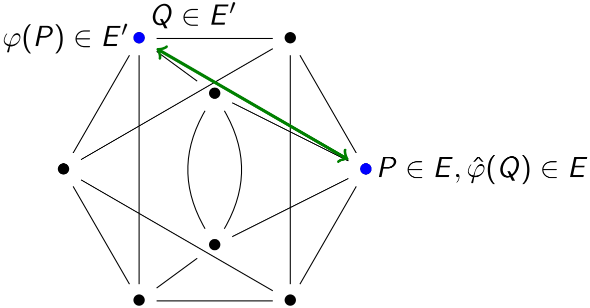


VDF over $\mathbb{F}_{p^2}$ supersingular curves.

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

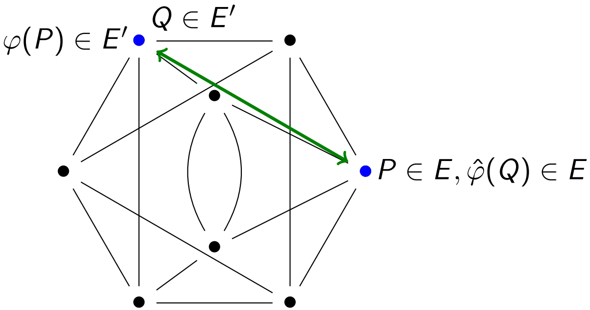
Evaluation For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtrack walk).

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



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

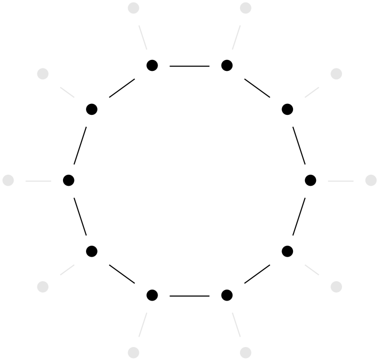
Setup	A **public** walk in the isogeny graph.
Evaluation	For $Q \in E'$, compute $\hat{\varphi}(Q)$ (the backtrack walk).
Verification	Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q)$.



Not post-quantum, but also no proof needed!

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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

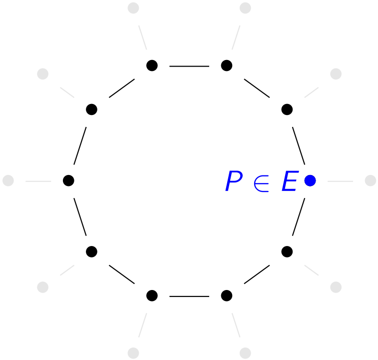


VDF over $\mathbb{F}_p$ supersingular curves.

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

Setup Choose a curve E on the crater.

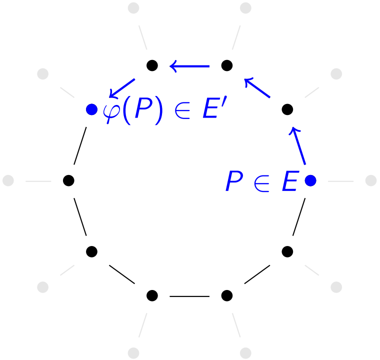
Choose $P \in E(\mathbb{F}_p)[N]$.



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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

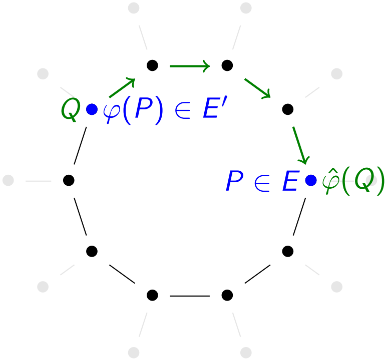
Setup Choose a curve E on the crater.
Choose $P \in E(\mathbb{F}_p)[N]$.
Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.



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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

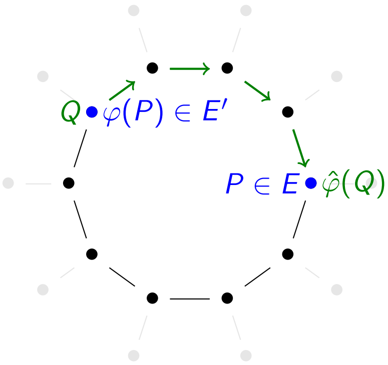
Setup	Choose a curve E on the crater. Choose $P \in E(\mathbb{F}_p)[N]$. Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.
Evaluation	Compute $\hat{\varphi}(Q)$ for a given $Q \in E'(\mathbb{F}_{p^2})[N]$.



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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

Setup	Choose a curve E on the crater. Choose $P \in E(\mathbb{F}_p)[N]$. Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.
Evaluation	Compute $\hat{\varphi}(Q)$ for a given $Q \in E'(\mathbb{F}_{p^2})[N]$.
Verification	Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q) \neq 1$.



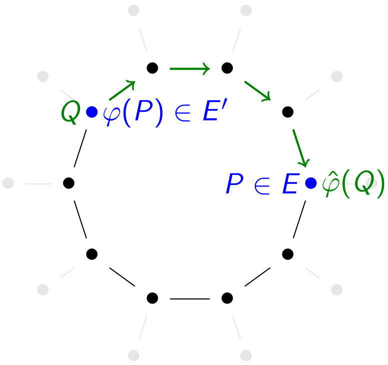
VDF over $\mathbb{F}_p$ supersingular curves.

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

Setup Choose a curve E on the crater.
Choose $P \in E(\mathbb{F}_p)[N]$.
Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.

Evaluation Compute $\hat{\varphi}(Q)$ for a given $Q \in E'(\mathbb{F}_{p^2})[N]$.

Verification Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q) \neq 1$.

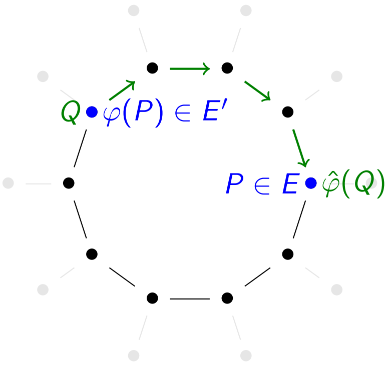


Similarity with the class group VDF:

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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

Setup	Choose a curve E on the crater. Choose $P \in E(\mathbb{F}_p)[N]$. Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.
Evaluation	Compute $\hat{\varphi}(Q)$ for a given $Q \in E'(\mathbb{F}_{p^2})[N]$.
Verification	Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q) \neq 1$.



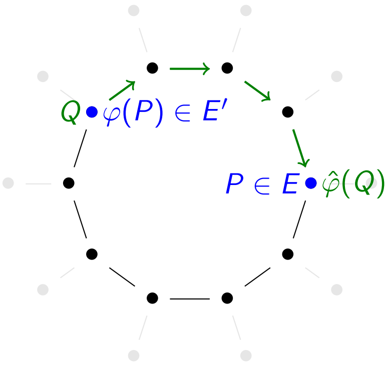
Similarity with the class group VDF:

$$E_1 \xrightarrow{f} E_2 \xrightarrow{g} E_3$$

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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

Setup	Choose a curve E on the crater. Choose $P \in E(\mathbb{F}_p)[N]$. Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.
Evaluation	Compute $\hat{\varphi}(Q)$ for a given $Q \in E'(\mathbb{F}_{p^2})[N]$.
Verification	Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q) \neq 1$.



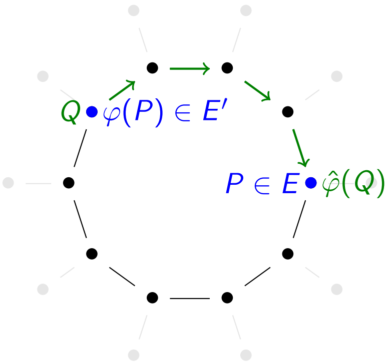
Similarity with the class group VDF:

$$\begin{array}{ccccc} E_1 & \xrightarrow{f} & E_2 & \xrightarrow{g} & E_3 \\ \mathrm{End}(E_1) & \xrightarrow{I} & \mathrm{End}(E_2) & \xrightarrow{J} & \mathrm{End}(E_3) \end{array}$$

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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

Setup	Choose a curve E on the crater.
	Choose $P \in E(\mathbb{F}_p)[N]$.
	Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.
Evaluation	Compute $\hat{\varphi}(Q)$ for a given $Q \in E'(\mathbb{F}_{p^2})[N]$.
Verification	Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q) \neq 1$.



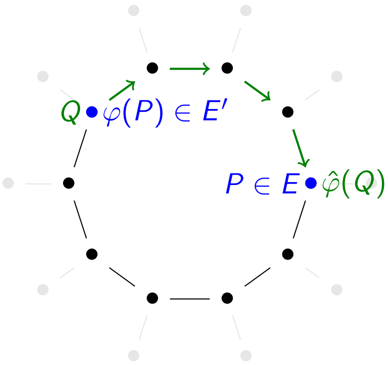
Similarity with the class group VDF:

$$\begin{array}{ccccc} E_1 & & \xrightarrow{g \circ f} & & E_3 \\ \mathrm{End}(E_1) & \xrightarrow{I} & \mathrm{End}(E_2) & \xrightarrow{J} & \mathrm{End}(E_3) \end{array}$$

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

Consider only the curves and isogenies defined over $\mathbb{F}_p$.

Setup	Choose a curve E on the crater. Choose $P \in E(\mathbb{F}_p)[N]$. Choose a direction for the isogeny and compute $\varphi(P) \in E'(\mathbb{F}_p)[N]$.
Evaluation	Compute $\hat{\varphi}(Q)$ for a given $Q \in E'(\mathbb{F}_{p^2})[N]$.
Verification	Check that $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q) \neq 1$.



Similarity with the class group VDF:

$$\begin{array}{ccc} E_1 & \xrightarrow{g \circ f} & E_3 \\ \mathrm{End}(E_1) & \xrightarrow{IJ} & \mathrm{End}(E_3) \end{array}$$

Attacks on the VDF.

- DLP over the curves.
  P and Q of order N with $\log_2(N) \approx 256$.
  $$\#E(\mathbb{F}_p) = p + 1$$
  so we set $p = hN - 1$ with h a cofactor.

Attacks on the VDF.

- DLP over the curves.
  P and Q of order N with $\log_2(N) \approx 256$.
  $$\#E(\mathbb{F}_p) = p + 1$$
  so we set $p = hN - 1$ with h 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 $\log_2(h) \approx 1250$.

Attacks on the VDF.

- DLP over the curves.
  P and Q of order N with $\log_2(N) \approx 256$.
  $$\#E(\mathbb{F}_p) = p + 1$$
  so we set $p = hN - 1$ with h 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 $\log_2(h) \approx 1250$.
- Isogeny shortcut.
  If E have a particular endomorphism ring, a shortcut can be found:
  $$E \quad \xrightarrow{\varphi} \quad E'$$

Attacks on the VDF.

- DLP over the curves.
  P and Q of order N with $\log_2(N) \approx 256$.
  $$\#E(\mathbb{F}_p) = p + 1$$
  so we set $p = hN - 1$ with h 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 $\log_2(h) \approx 1250$.
- Isogeny shortcut.
  If E have a particular endomorphism ring, a shortcut can be found:

$$\begin{array}{ccc} E & \xrightarrow{\varphi} & E' \\ \updownarrow & & \updownarrow \\ \mathrm{End}(E) = \mathcal{O} & \xrightarrow{I} & \mathcal{O}' = \mathrm{End}(E') \end{array}$$

Attacks on the VDF.

- DLP over the curves.
  P and Q of order N with $\log_2(N) \approx 256$.

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

  so we set $p = hN - 1$ with h 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 $\log_2(h) \approx 1250$.
- Isogeny shortcut.
  If E have a particular endomorphism ring, a shortcut can be found:

$$\begin{array}{ccc} E & \xrightarrow{\varphi} & E' \\ \updownarrow & & \updownarrow \\ \mathrm{End}(E) = \mathcal{O} & \xrightarrow[J]{I} & \mathcal{O}' = \mathrm{End}(E') \end{array}$$

Attacks on the VDF.

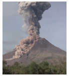
- DLP over the curves.
  P and Q of order N with $\log_2(N) \approx 256$.
  $$\#E(\mathbb{F}_p) = p + 1$$
  so we set $p = hN - 1$ with h a cofactor.
- DLP over the finite field $\mathbb{F}_{p^2}$.
  NFS over $\mathbb{F}_{p^2}$: $\log_2(p) \approx 1500$. We need a cofactor of size $\log_2(h) \approx 1250$.
- Isogeny shortcut.
  If E have a particular endomorphism ring, a shortcut can be found:

$$\begin{tikzcd}
E \arrow[r, "\varphi"] \arrow[d, leftrightarrow] & E' \arrow[d, leftrightarrow] \\
\mathrm{End}(E) = \mathcal{O} \arrow[r, "I", "J"'] \arrow[d, leftrightarrow] & \mathcal{O}' = \mathrm{End}(E') \arrow[d, leftrightarrow] \\
E \arrow[r, "\tilde{\varphi}", "\text{short deg}"'] & E'
\end{tikzcd}$$

Random endomorphism ring curves.

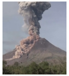
Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known $\mathrm{End}(E)$.



Random endomorphism ring curves.

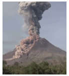
- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known $\mathrm{End}(E)$.



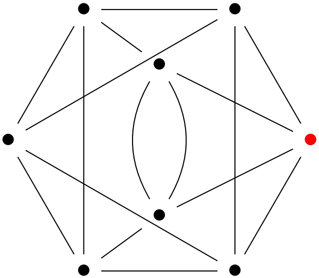
- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.

Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known End(E).

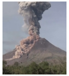


- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.
  Use a trusted setup to avoid isogeny shortcut:

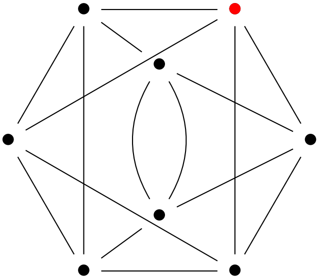


Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known $\mathrm{End}(E)$.

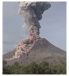


- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.
  Use a trusted setup to avoid isogeny shortcut:

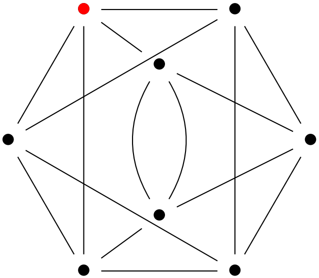


Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known End(E).

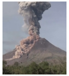


- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.
  Use a trusted setup to avoid isogeny shortcut:

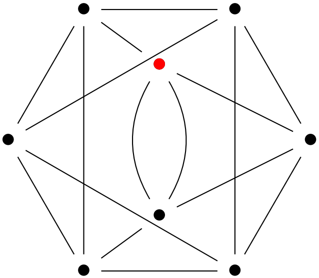


Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known $\mathrm{End}(E)$.

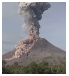


- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.
  Use a trusted setup to avoid isogeny shortcut:

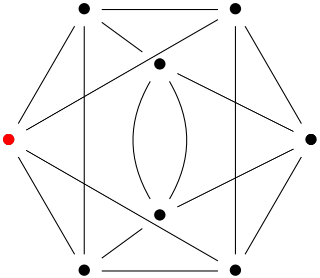


Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known $\mathrm{End}(E)$.

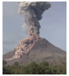


- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.
  Use a trusted setup to avoid isogeny shortcut:

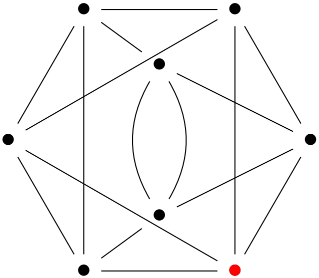


Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known End(E).

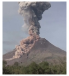


- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.
  Use a trusted setup to avoid isogeny shortcut:

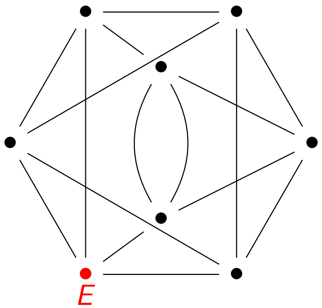


Random endomorphism ring curves.

- Ordinary curves. Pairing friendly $\longrightarrow$ small discrimnant $\longrightarrow$ known $\mathrm{End}(E)$.



- Supersingular curves.
  Open problem: compute a supersingular elliptic curve of unknown endomorphism ring.
  Use a trusted setup to avoid isogeny shortcut:

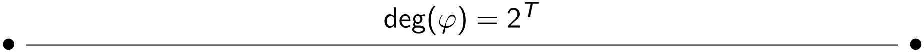


Efficient isogeny.

Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:

Efficient isogeny.

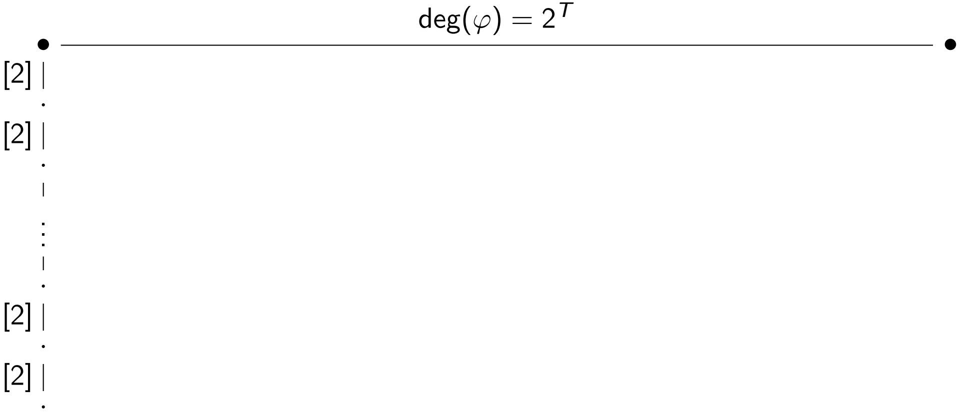
Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:



Efficient isogeny.

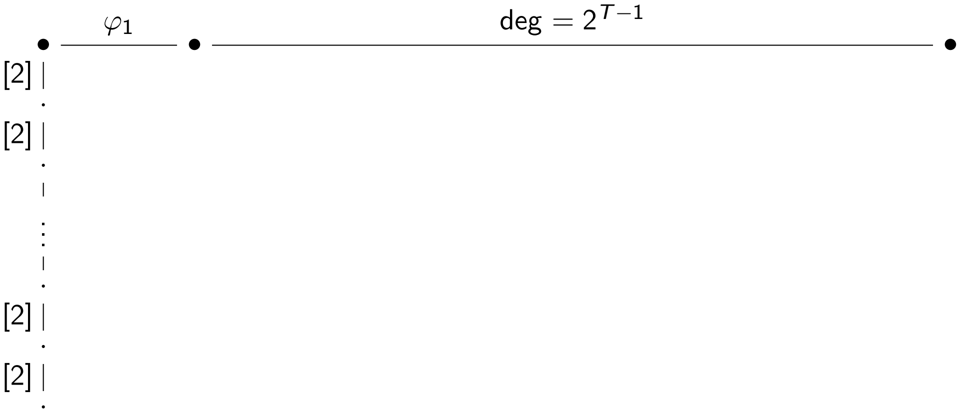
Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.

It defines an isogeny of degree 2^T:



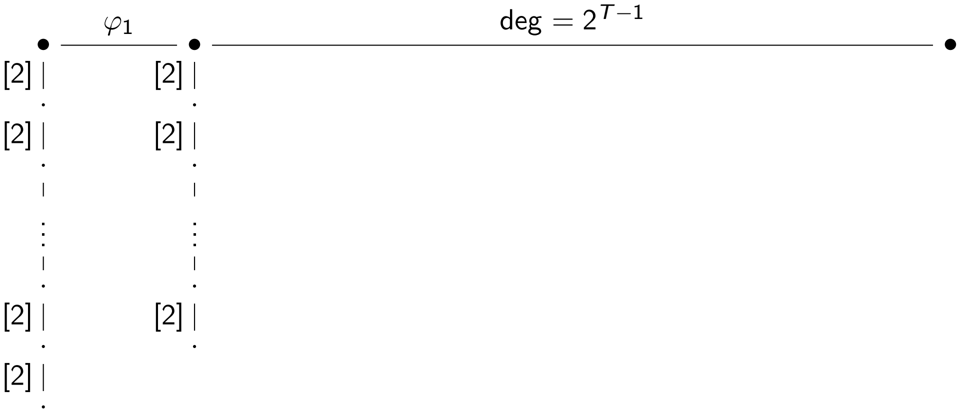
Efficient isogeny.

Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:



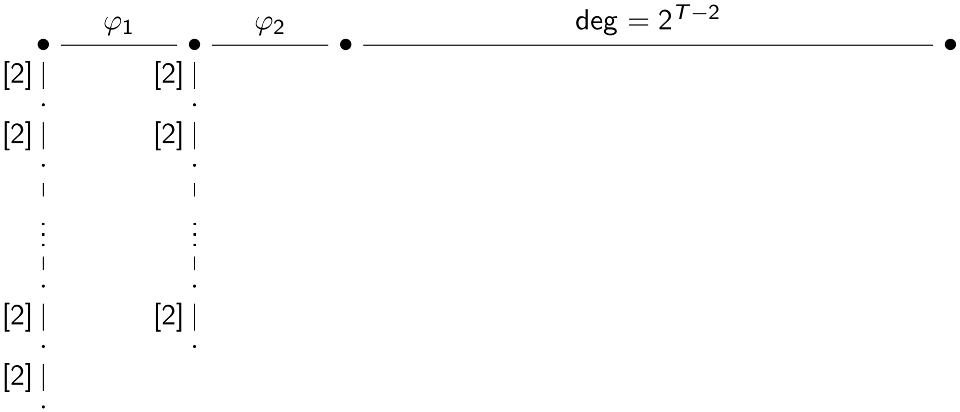
Efficient isogeny.

Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:



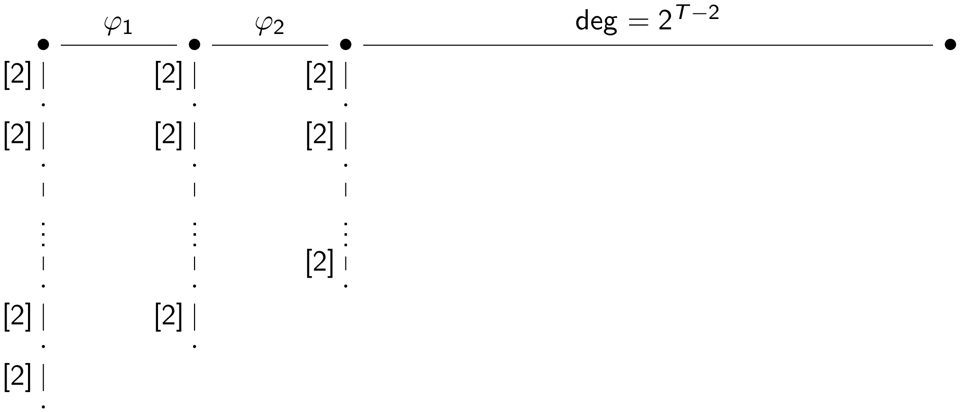
Efficient isogeny.

Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:



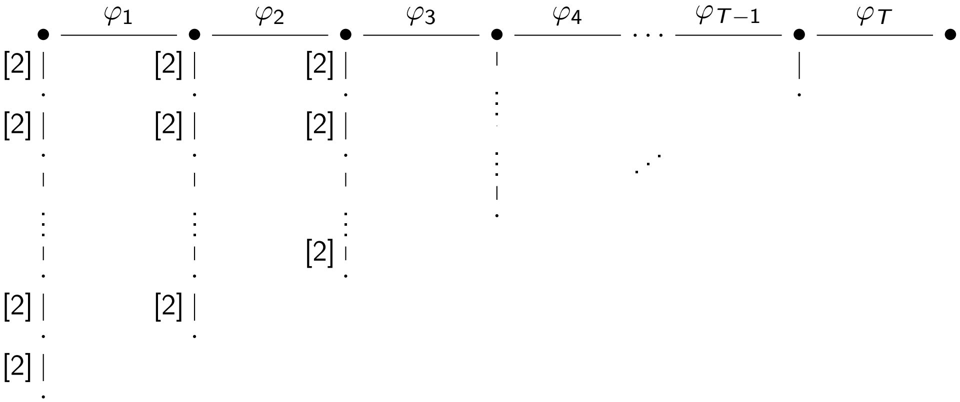
Efficient isogeny.

Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:



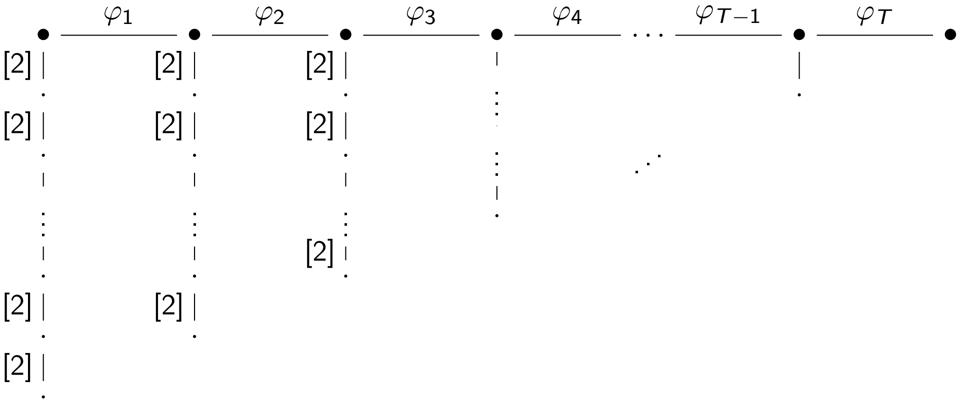
Efficient isogeny.

Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:



Efficient isogeny.

Suppose we have a point P of order 2^T defined over $\mathbb{F}_p$.
It defines an isogeny of degree 2^T:



Complexity: $O(T^2)$. It can be turned into $O(T\log_2(T))$ with a recursive strategy.

In practice, we cannot find a point of order 2^T (it is too large).
We choose curves such that $2^n \mid \#E(\mathbb{F}_p)$ with n as large as possible.

In practice, we cannot find a point of order 2^T (it is too large).
We choose curves such that $2^n \mid \#E(\mathbb{F}_p)$ with n as large as possible.

$$\bullet \;\overset{2^n}{\rule{2cm}{0.4pt}}\; \bullet \;\overset{2^n}{\rule{2cm}{0.4pt}}\; \cdots \;\overset{2^n}{\rule{2cm}{0.4pt}}\; \bullet \;\overset{2^n}{\rule{2cm}{0.4pt}}\; \bullet$$

In practice, we cannot find a point of order 2^T (it is too large).
We choose curves such that $2^n \mid \#E(\mathbb{F}_p)$ with n as large as possible.

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

$$p = 2^n fN - 1$$

$$\log_2(p) = 1500$$

Proof-of-concept available on https://github.com/isogeny-vdf.

In practice, we cannot find a point of order 2^T (it is too large).
We choose curves such that $2^n \mid \#E(\mathbb{F}_p)$ with n as large as possible.

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

$$p = 2^n f N - 1$$

$$\log_2(p) = 1500$$

Proof-of-concept available on https://github.com/isogeny-vdf.

In practice, we cannot find a point of order 2^T (it is too large).
We choose curves such that $2^n \mid \#E(\mathbb{F}_p)$ with n as large as possible.

$$\bullet \overset{2^n}{\text{———}} \bullet \overset{2^n}{\text{———}} \cdots \overset{2^n}{\text{———}} \bullet \overset{2^n}{\text{———}} \bullet$$

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

$$p = 2^n f N - 1$$

$$\log_2(p) = 1500$$

Proof-of-concept available on https://github.com/isogeny-vdf.

In practice, we cannot find a point of order 2^T (it is too large).
We choose curves such that $2^n \mid \#E(\mathbb{F}_p)$ with n as large as possible.

$$\bullet \overset{2^n}{\text{————}} \bullet \overset{2^n}{\text{————}} \cdots \overset{2^n}{\text{————}} \bullet \overset{2^n}{\text{————}} \bullet$$

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

$$p = 2^n f N - 1$$

$$\log_2(p) = 1500$$

$$\bullet \overset{2^{1244}}{\text{————}} \bullet \overset{2^{1244}}{\text{————}} \cdots \overset{2^{1244}}{\text{————}} \bullet \overset{2^{1244}}{\text{————}} \bullet$$

Proof-of-concept available on https://github.com/isogeny-vdf.

Post-quantum security.

- Our VDF is **not** post-quantum (discrete log problem).

Post-quantum security.

- Our VDF is **not** post-quantum (discrete log problem).
- Our VDF over $\mathbb{F}_{p^2}$ is quantum-annoying: once the setup is done, a quantum computer need to break the DLP for each evaluation of the VDF.

Post-quantum security.

- Our VDF is **not** post-quantum (discrete log problem).
- Our VDF over $\mathbb{F}_{p^2}$ is quantum-annoying: once the setup is done, a quantum computer need to break the DLP for each evaluation of the VDF.
- Our VDF over $\mathbb{F}_p$ is **not** quantum-annoying: once the setup is done, a quantum computer can compute the class number $\mathrm{Cl}(-D)$ and then find a faster isogeny (similar to Wesolowski group-class VDF).

A generalization of the BLS signature.

A generalization of the BLS signature.

Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: s an integer
- Public key: $P_K = \varphi(P)$.

Sign Hash the message m into $\mathbb{G}_2$ and the signature is $\sigma = [s]H(m)$.

Verify Check that $e(P, \sigma) = e(P_K, H(m))$.

A generalization of the BLS signature.

Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: φ an isogeny $E \to E'$
- Public key: $P_K = \varphi(P)$.

Sign Hash the message m into $\mathbb{G}_2$ (on E') and the signature is $\sigma = \hat{\varphi}(H(m))$.

Verify Check that $e(P, \sigma) = \tilde{e}(P_K, H(m))$.

A generalization of the BLS signature.
Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: φ an isogeny $E \to E'$
- Public key: $P_K = \varphi(P)$.

Sign Hash the message m into $\mathbb{G}_2$ (on E') and the signature is $\sigma = \hat{\varphi}(H(m))$.

Verify Check that $e(P, \sigma) = \tilde{e}(P_K, H(m))$.

Patented by Broker, Charles, and Lauter in 2012 (different implementation, not efficient).

A generalization of the BLS signature.

Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: φ an isogeny $E \to E'$
- Public key: $P_K = \varphi(P)$.

Sign Hash the message m into $\mathbb{G}_2$ (on E') and the signature is $\sigma = \hat{\varphi}(H(m))$.

Verify Check that $e(P, \sigma) = \tilde{e}(P_K, H(m))$.

Patented by Broker, Charles, and Lauter in 2012 (different implementation, not efficient).

We obtain an identification protocol where the secret can be sub-exponentially larger than the proof. But it is not zero-knowledge.

A generalization of the BLS signature.

Let E an elliptic curve and $P \in E(\mathbb{F}_p)$ a point of order N.

- Secret key: φ an isogeny $E \to E'$
- Public key: $P_K = \varphi(P)$.

Sign Hash the message m into $\mathbb{G}_2$ (on E') and the signature is $\sigma = \hat{\varphi}(H(m))$.

Verify Check that $e(P, \sigma) = \tilde{e}(P_K, H(m))$.

Patented by Broker, Charles, and Lauter in 2012 (different implementation, not efficient).

We obtain an identification protocol where the secret can be sub-exponentially larger than the proof. But it is not zero-knowledge.

Now looking for an accumulator... But we failed!

Thank you for your attention.