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

Uniqueness If Verify($pp, x, y, \pi$) = 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 poly($T$) parallel processors.
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($\lambda, T$) $\rightarrow$ public parameters $pp$
- Eval($pp, x$) $\rightarrow$ output $y$, proof $\pi$ (requires $T$ steps)
- Verify($pp, x, y, \pi$) $\rightarrow$ yes or no.

**Uniqueness** If Verify($pp, x, y, \pi$) = 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 poly($T$) parallel processors.
Application. How to generate randomness in the real world?

Idea: slow things down by adding delay.
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?

Fail 1 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.

\[ \forall x \in \langle g \rangle, f(x) = \log_g(x) \]

Verification is easy: \( g(f(x)) = x \).

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: \( h(f(x)) = x \).

Computation is faster as long as you parallelize.
VDF without "delay": public-key cryptography.

\[ \forall x \in \langle g \rangle, \quad f(x) = \log_g(x) \]

Verification is easy: \( g^{f(x)} \stackrel{?}{=} x \).
You can parallelize to compute \( f(x) \).
VDF without "delay": public-key cryptography.

\[ \forall x \in \langle g \rangle, \quad f(x) = \log_g(x) \]

Verification is easy: \( g^{f(x)} \equiv x \).
You can parallelize to compute \( f(x) \).

VDF without "verifiability": composition of hash functions.

\[ f(x) = h^{-1}(x) \]
Verification is easy: \( h(f(x)) \equiv x \).
Computation is faster as long as you parallelize.
VDF without “delay”: public-key cryptography.

\[ \forall x \in \langle g \rangle, \quad f(x) = \log_g(x) \]

Verification is easy: \( g^{f(x)} = x \).
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.

\[ \forall x \in \langle g \rangle, \quad f(x) = \log_g(x) \]

Verification is easy: \( g^{f(x)} \overset{?}{=} x \).
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.

\[ \forall x \in \langle g \rangle, \quad f(x) = \log_g(x) \]

Verification is easy: \( g^{f(x)} \overset{?}{=} x \).
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: \( h(f(x)) \overset{?}{=} x \).
Computation is faster as long as you parallelize.
VDF based on RSA.

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

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

Verification. Proof of a correct exponentiation. If one knows the factorization of $N$, the evaluation can be computed using $H(x)^{2^T \equiv H(x)^{2^T \mod \phi(N) \mod N}}$. Need a trusted setup to choose $N$. 

Turned to non-interactive using Fiat-Shamir. $\pi$ is short. Verification is fast.
VDF based on RSA.

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

Evaluation. $y = H(x)^{2^T} \text{ mod } N$, and $\pi$ a proof.
VDF based on RSA.

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

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

Verification. Proof of a correct exponentiation.
VDF based on RSA.

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

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

Verification. Proof of a correct exponentiation.

Wesolowski proof.

- Verifier challenges with a small prime $\ell$
VDF based on RSA.

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

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

Verification. Proof of a correct exponentiation.

Wesolowski proof.

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

Turned to non-interactive using Fiat-Shamir

Verification is fast.

If one knows the factorization of $N$, the evaluation can be computed using $H(x)^{2^T} \equiv H(x)^{2^T} \mod \phi(N) \mod N$

Need a trusted setup to choose $N$. 

VDF based on RSA.

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

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

**Verification.** Proof of a correct exponentiation.

**Wesolowski proof.**

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

Turned to non-interactive using Fiat-Shamir. $\pi$ is short. Verification is fast.

If one knows the factorization of $N$, the evaluation can be computed using $H(x)^{2^T} \equiv H(x)^{2^T} \mod \phi(N) \mod N$.

Need a trusted setup to choose $N$. 
VDF based on RSA.

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

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

Verification. Proof of a correct exponentiation.

Wesolowski proof.

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

Turned to non-interactive using Fiat-Shamir

$\pi$ is short

Verification is fast.
**VDF based on RSA.**

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

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

**Verification.** Proof of a correct exponentiation.

**Wesolowski proof.**

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

Turned to non-interactive using Fiat-Shamir

$\pi$ is short

Verification is fast.

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

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

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

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

This group is finite and it is hard to compute $\#\text{ClassGroup}(D)$.
If one can compute a root mod $N$, the VDF is **unsound**:
Choose $w$ and compute $\sqrt[ℓ]{w}$. $(y, π)$ and $(wy, \sqrt[ℓ]{wπ})$ 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.

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

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

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 $G_1 \times G_2$ where $G_1 \subset E(\mathbb{F}_p)$ and $G_2 \subset E(\mathbb{F}_{p^k})$. 

**Definition**

A pairing on $E$ is a bilinear non-degenerate application $e : G_1 \times G_2 \rightarrow \mathbb{F}_k^\times$.

**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:** $PK = [s]P$.

Sign Hash the message $m$ into $G_2$ and the signature is $\sigma = [s]H(m)$.

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

$$e(P, \sigma) = e([s]P, H(m)) = e(PK, H(m)).$$
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 \( G_1 \times G_2 \) where \( G_1 \subset E(\mathbb{F}_p) \) and \( G_2 \subset E(\mathbb{F}_{p^k}) \).

**Definition**
A pairing on \( E \) is a bilinear non-degenerate application \( e : G_1 \times G_2 \rightarrow \mathbb{F}_{p^k}^\times \).
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 $G_1 \times G_2$ where $G_1 \subset E(\mathbb{F}_p)$ and $G_2 \subset E(\mathbb{F}_{p^k})$.

**Definition**
A pairing on $E$ is a bilinear non-degenerate application $e : G_1 \times G_2 \rightarrow \mathbb{F}_{p^k}^\times$

**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 $G_1 \times G_2$ where $G_1 \subset E(\mathbb{F}_p)$ and $G_2 \subset E(\mathbb{F}_{p^k})$.

**Definition**

A pairing on $E$ is a bilinear non-degenerate application $e : G_1 \times G_2 \to \mathbb{F}_{p^k}^\times$

**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 $G_1 \times G_2$ where $G_1 \subset E(\mathbb{F}_p)$ and $G_2 \subset E(\mathbb{F}_{p^k})$.

**Definition**

A pairing on $E$ is a bilinear non-degenerate application $e : G_1 \times G_2 \longrightarrow \mathbb{F}_{p^k}^\times$.

**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 $G_2$ 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 $G_1 \times G_2$ where $G_1 \subset E(\mathbb{F}_p)$ and $G_2 \subset E(\mathbb{F}_{p^k})$.

**Definition**

A pairing on $E$ is a bilinear non-degenerate application $e : G_1 \times G_2 \rightarrow \mathbb{F}_{p^k}^\times$.

**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 $G_2$ 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 $\varphi$ such that $\varphi(0_E) = 0_{E'}$. 

Example (Frobenius)

For $A, B \in \overline{\mathbb{F}}_p$, $\pi_p: E: y^2 = x^3 + Ax + B \rightarrow E(p): y^2 = x^3 + A_p x + B_p$, $V'\text{elu}'s$ formulas. 

For $P \in E(\overline{\mathbb{F}}_p)$ of order $\ell$ coprime with $p$, we have formulas for computing an isogeny $\varphi$ of kernel $\langle P \rangle$. The degrees of the polynomials defining $\varphi$ is $O(\ell)$. In practice, V'\text{elu}'s formulas are efficient for very small kernel.

From $\varphi: E \rightarrow E'$, there exists $\hat{\varphi}: E' \rightarrow E$ such that $\varphi \circ \hat{\varphi} = \hat{\varphi} \circ \varphi = [\deg \varphi]$. 

$e(\varphi(P), \varphi(Q)) = e(P, Q) \deg(\varphi)$
Definition (Isogeny)
An isogeny between two elliptic curves $E$ and $E'$ is an algebraic map $\varphi$ such that $\varphi(0_E) = 0_{E'}$.

Example (Frobenius)
For $A, B \in \overline{\mathbb{F}}_p$,

$$\pi_p : E : y^2 = x^3 + Ax + B \rightarrow E^{(p)} : y^2 = x^3 + A^p x + B^p$$

$$(x, y) \mapsto (x^p, y^p)$$
Definition (Isogeny)
An isogeny between two elliptic curves $E$ and $E'$ is an algebraic map $\varphi$ such that $\varphi(0_E) = 0_{E'}$.

Example (Frobenius)
For $A, B \in \bar{\mathbb{F}}_p$, $\pi_p : E : y^2 = x^3 + Ax + B \mapsto E(p) : y^2 = x^3 + A^p x + B^p$

$(x, y) \mapsto (x^p, y^p)$

Vélu’s formulas. For $P \in E(\bar{\mathbb{F}}_p)$ of order $\ell$ coprime with $p$, we have formulas for computing an isogeny $\varphi$ of kernel $\langle P \rangle$. The degrees of the polynomials defining $\varphi$ is $O(\ell)$. 
Definition (Isogeny)
An isogeny between two elliptic curves $E$ and $E'$ is an algebraic map $\varphi$ such that $\varphi(0_E) = 0_{E'}$.

Example (Frobenius)
For $A, B \in \overline{\mathbb{F}}_p$,

$$\pi_p : E : y^2 = x^3 + Ax + B \quad \mapsto \quad E^{(p)} : y^2 = x^3 + A^px + B^p$$

$(x, y) \mapsto (x^p, y^p)$

Vélu’s formulas. For $P \in E(\overline{\mathbb{F}}_p)$ of order $\ell$ coprime with $p$, we have formulas for computing an isogeny $\varphi$ of kernel $\langle P \rangle$. The degrees of the polynomials defining $\varphi$ 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 $\varphi$ such that $\varphi(0_E) = 0_{E'}$.

**Example (Frobenius)**

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

$$
\pi_p : E : y^2 = x^3 + Ax + B \longrightarrow E^{(p)} : y^2 = x^3 + A^px + B^p
$$

$$(x, y) \mapsto (x^p, y^p)$$

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

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

From $\varphi : E \rightarrow E'$, there exists $\hat{\varphi} : E' \rightarrow E$ 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 $\varphi$ such that $\varphi(0_E) = 0_{E'}$.

Example (Frobenius)
For $A, B \in \overline{\mathbb{F}}_p$,

$$
\pi_p : E : y^2 = x^3 + Ax + B \longrightarrow E(p) : y^2 = x^3 + A^p x + B^p
$$

$(x, y) \mapsto (x^p, y^p)$

Vélu’s formulas. For $P \in E(\overline{\mathbb{F}}_p)$ of order $\ell$ coprime with $p$, we have formulas for computing an isogeny $\varphi$ of kernel $\langle P \rangle$. The degrees of the polynomials defining $\varphi$ 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:

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

- Supersingular curves
  \[ \text{End}(E) \text{ is a maximal order in the quaternion algebra } \mathbb{Q}_p, \infty \]
  - Supersingular curves are defined over \( \mathbb{F}_{p^2} \).
  - Isogeny graph is an expander.
Two types of elliptic curves:

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

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

Supersingular curves are defined over $\mathbb{F}_{p^2}$. 
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 an expander.

Supersingular curves are defined over $\mathbb{F}_{p^2}$. 
Two types of elliptic curves:

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

**Supersingular curves** $\text{End}(E')$ is a maximal order in the quaternion algebra $\mathbb{Q}_p$, $\infty$. Isogeny graph is an expander.

Supersingular curves are defined over $\mathbb{F}_{p^2}$. 
Two types of elliptic curves:

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

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

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

Supersingular curves $\text{End}(E)$ 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.

$$\varphi_1 \circ \varphi_2 \circ \ldots$$

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

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

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).

\[ \varphi(P) \in E' \quad Q \in E' \quad P \in E \]
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).

\[
\varphi(P) \in E' \\
\hat{\varphi}_3 \\
P \in E
\]

$Q \in E'$

$Q \in E'$

$P \in E$
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).

\[ \varphi(P) \in E' \rightarrow Q \in E' \rightarrow \hat{\varphi}_3 \rightarrow \hat{\varphi}_2 \rightarrow \hat{\varphi}_1 \rightarrow \hat{\varphi}(P) \in E' \rightarrow Q \in E' \rightarrow \hat{\varphi}_3 \rightarrow \hat{\varphi}_2 \rightarrow \hat{\varphi}_1 \rightarrow P \in E, \hat{\varphi}(Q) \in E \]
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 $e(P, \hat{\varphi}(Q)) = e(\varphi(P), Q)$.

![Diagram of isogeny graph](image)

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

![Diagram of a graph with a marked point P in the center]
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 $\varphi(P) \in E'(\mathbb{F}_p)[N]$.
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.
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:

$$
\begin{align*}
E_1 & \xrightarrow{f} E_2 & \xrightarrow{g} & E_3 \\
\text{End}(E_1) & \xrightarrow{l} & \text{End}(E_2) & \xrightarrow{j} & \text{End}(E_3)
\end{align*}
$$
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:

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

$\text{End}(E_1) \xrightarrow{I} \text{End}(E_2) \xrightarrow{J} \text{End}(E_3)$
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:

$$
\begin{align*}
E_1 & \xrightarrow{g \circ f} E_3 \\
\text{End}(E_1) & \xrightarrow{IJ} \text{End}(E_3)
\end{align*}
$$
Attacks on the VDF.

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

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

  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 $\log_2(h) \approx 1250$. 

- Isogeny shortcut.

  If $E$ have a particular endomorphism ring, a shortcut can be found:

  $$E \xrightarrow{\phi} E'$$

  $$\text{End}(E) \xrightarrow{\text{Isogeny shortcut}} \text{End}(E')$$

  $$E \xrightarrow{\tilde{\phi}} \text{short deg}$$
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:
    \[
    E \xrightarrow{\phi} E'
    \]
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:

  \[
  E \xrightarrow{\varphi} E'
  \]

  \[
  \downarrow \quad \uparrow
  \]

  \[
  \text{End}(E) = \mathcal{O} \xrightarrow{I} \mathcal{O}' = \text{End}(E')
  \]
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:

  \[
  \begin{array}{ccc}
  E & \xrightarrow{\varphi} & E' \\
  \updownarrow & \quad & \updownarrow \\
  \text{End}(E) = \mathcal{O} & \xrightarrow{I} & \mathcal{O}' = \text{End}(E')
  \end{array}
  \]
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:

\[
\begin{array}{ccc}
E & \xrightarrow{\varphi} & E' \\
\downarrow & & \downarrow \\
\text{End}(E) = \mathcal{O} & \xrightarrow{I} & \mathcal{O'} = \text{End}(E') \\
\downarrow & & \downarrow \\
E & \xrightarrow{\tilde{\varphi}} & E' \\
& \text{short deg} & \\
\end{array}
\]
Random endomorphism ring curves.

Ordinary curves. Pairing friendly
-→ small discriminant
-→ known $\text{End}(E)$.

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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$. 

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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$.

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

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

- 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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$.

- 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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$.

- 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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$.

- 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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$.

- 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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$.

- 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 $\rightarrow$ small discriminant $\rightarrow$ known $\text{End}(E)$.

- 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 \( 2^T \):

\[
\phi_1 \phi_2 \phi_3 \phi_4 \cdots \phi_{2^T-1} \phi_{2^T}
\]

Complexity: \( O(T^2) \). It can be turned into \( O(T \log_2(T)) \) with a recursive strategy.
Efficient isogeny.
Suppose we have a point $P$ of order $2^T$ defined over $\mathbb{F}_p$.
It defines an isogeny of degree $2^T$:

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

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

$$\varphi_1 \quad \text{deg} = 2^{T-1}$$

Complexity: $O(T^2)$. It can be turned into $O(T \log_2 T)$ with a recursive strategy.
**Efficient isogeny.**
Suppose we have a point $P$ of order $2^T$ defined over $\mathbb{F}_p$. It defines an isogeny of degree $2^T$:

\[
\begin{array}{c}
\bullet & \varphi_1 & \bullet & \text{deg} = 2^{T-1} \\
[2] & \downarrow & & \\
[2] & \downarrow & & \\
[2] & \downarrow & & \\
\end{array}
\]

Complexity: $O(T^2)$. It can be turned into $O(T \log_2(T))$ with a recursive strategy.
Efficient isogeny.
Suppose we have a point $P$ of order $2^T$ defined over $\mathbb{F}_p$.
It defines an isogeny of degree $2^T$:

\[
\begin{array}{c}
\bullet & \varphi_1 & \bullet & \varphi_2 & \bullet \\
| & \cdot & | & \cdot & | \\
| & \cdot & | & \cdot & | \\
| & \cdot & | & \cdot & | \\
| & \cdot & | & \cdot & | \\
| & \cdot & | & \cdot & | \\
\end{array}
\]

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

$\varphi_1 \varphi_2$ $\varphi_3 \varphi_4 \cdots$ $\deg = 2^{T-2}$

Complexity: $O(T^2)$. It can be turned into $O(T \log_2(T))$ with a recursive strategy.
**Efficient isogeny.**
Suppose we have a point $P$ of order $2^T$ defined over $\mathbb{F}_p$. It defines an isogeny of degree $2^T$:

```
\[\bullet \phi_1 \bullet \phi_2 \bullet \phi_3 \bullet \phi_4 \ldots \phi_{T-1} \bullet \phi_T \bullet\]
```

Complexity: $O(T^2)$. It can be turned into $O(T \log_2(T))$ with a recursive strategy.
Efficient isogeny.
Suppose we have a point $P$ of order $2^T$ defined over $\mathbb{F}_p$. It defines an isogeny of degree $2^T$:

\[
\begin{array}{ccccccc}
\bullet & \varphi_1 & \bullet & \varphi_2 & \bullet & \varphi_3 & \bullet & \varphi_4 & \ldots & \varphi_{T-1} & \bullet & \varphi_T & \bullet \\
[2] & \boxed{[2]} & [2] & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} & \boxed{[2]} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

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.
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.

\[
\begin{align*}
\bullet & \quad 2^n \quad \bullet \quad 2^n \quad \ldots \quad 2^n \quad \bullet \quad 2^n \\
\end{align*}
\]

\[
\begin{align*}
\#E(\mathbb{F}_p) &= p + 1 \\
p &= 2^n f N - 1 \\
\log_2(p) &= 1500
\end{align*}
\]

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.

\[
\begin{align*}
&2^n \quad 2^n \quad \ldots \quad 2^n \quad 2^n \\
\end{align*}
\]

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

$p = 2^n fN - 1$

$log_2(p) = 1500$

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

\[ \bullet \quad 2^n \quad \bullet \quad 2^n \quad \ldots \quad 2^n \quad \bullet \quad 2^n \quad \bullet \]

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

\[ p = 2^n f N - 1 \]

\[ \log_2(p) = 1500 \]

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.

\[
\begin{align*}
\bullet & \quad 2^n \quad \bullet \quad 2^n \quad \ldots \quad 2^n \quad \bullet \quad 2^n \\
\end{align*}
\]

\[
\#E(\mathbb{F}_p) = p + 1 \\
p = 2^n f N - 1 \\
\log_2(p) = 1500
\]

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 $\text{Cl}(-D)$ and then find a faster isogeny (similar to Wesolowski group-class VDF).
A generalization of the BLS signature.

Let $E$ an elliptic curve and $P \in E(F_p)$ a point of order $N$.

- **Secret key:**
- **Public key:** $PK = \phi(P)$.

Sign Hash the message $m$ into $G$ and the signature is $\sigma$.

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

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!
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 $P \in E(F_p)$ a point of order $N$.

- Secret key: $\varphi$ 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!
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: $\varphi$ 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!
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:** $\varphi$ an isogeny $E \rightarrow E'$
- **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:** $\varphi$ an isogeny $E \rightarrow E'$
- **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.