

# Effective Pairings

in Isogeny-based Cryptography

Krijn Reijnders  
LATINCRYPT 2023

**Pairings map  
elliptic curve problems  
to finite field problems**

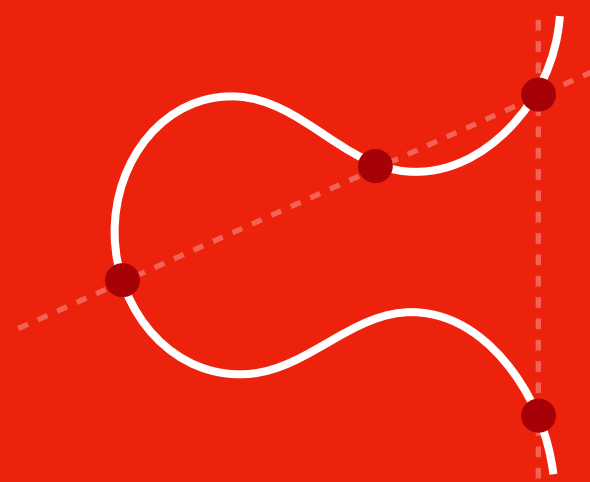
**Elliptic curve arithmetic  
is slow**

**Finite field arithmetic  
is (very) fast**

**Hence, fast pairings  
means fast solutions**

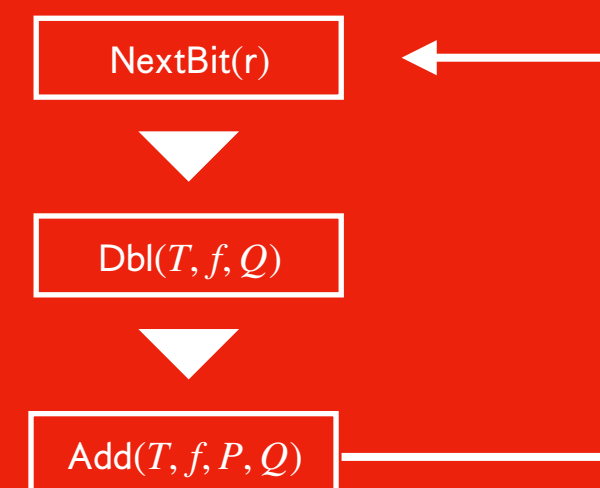
# Effective Pairings in Isogeny-based Cryptography

1



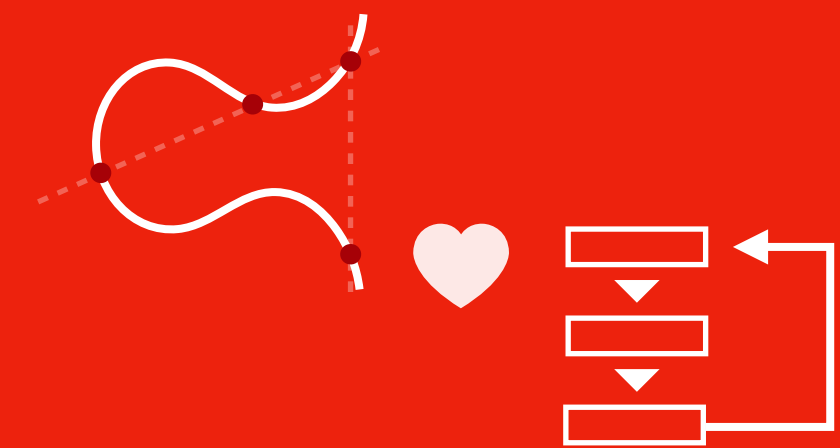
Isogenies  
& Pairings

2



Speeding-up  
general pairings

3



Applying pairings  
in isogeny crypto

# What are pairings and what are isogenies?

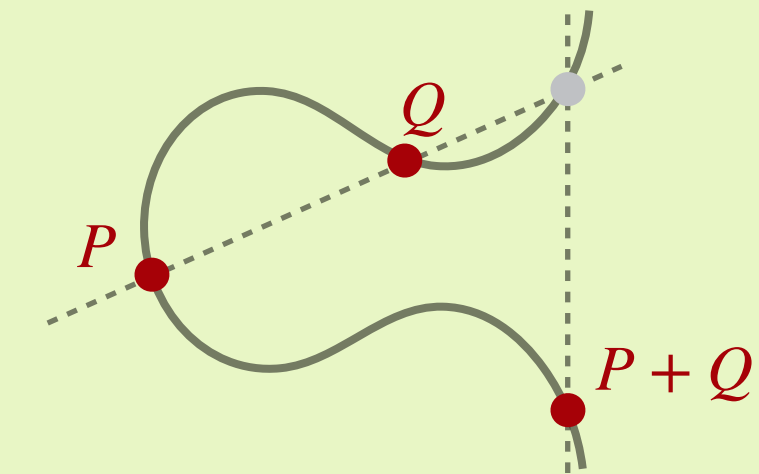
## Isogenies & Pairings



## elliptic curves in CSIDH

### supersingular elliptic curve

- has  $p + 1$  points in  $E(\mathbb{F}_p)$
- choose  $p$  so that  $p + 1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$
- this implies the rational points on  $E$  have orders that divide  $p + 1$



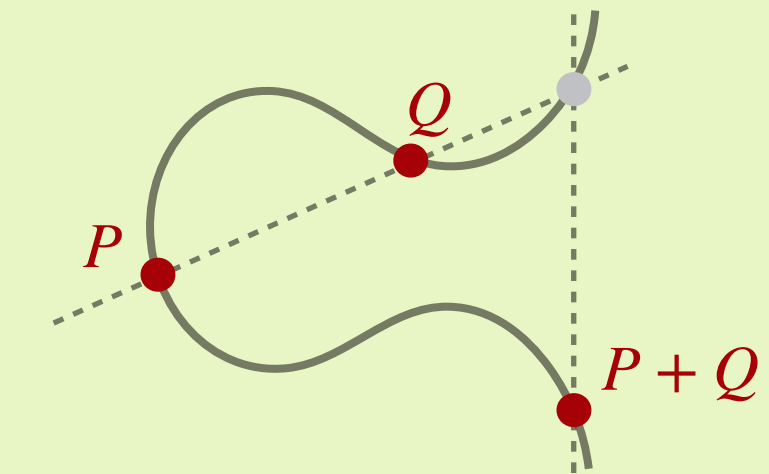
$$E : y^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_p$$

## Isogenies & Pairings

## elliptic curves in CSIDH

### supersingular elliptic curve

- has  $p + 1$  points in  $E(\mathbb{F}_p)$
- choose  $p$  so that  $p + 1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$
- this implies the rational points on  $E$  have orders that divide  $p + 1$



$$E : y^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_p$$

### points on such curves

We have that

$$E(\mathbb{F}_p) \cong \mathbb{Z}_4 \times \mathbb{Z}_{\ell_1} \times \mathbb{Z}_{\ell_2} \times \dots \times \mathbb{Z}_{\ell_n},$$

So think of a point  $P \in E(\mathbb{F}_p)$  as a sum of points  $P_i$  of order  $\ell_i$

$$P = P_0 + P_1 + P_2 + \dots + P_n$$

which shows how scalars  $[\lambda]$  with  $\lambda \in \mathbb{N}$  affect the torsion

$$\begin{aligned} [\ell_2]P &= [\ell_2]P_0 + [\ell_2]P_1 + [\ell_2]P_2 + \dots + [\ell_2]P_n \\ &= [\ell_2]P_0 + [\ell_2]P_1 + \mathcal{O} + \dots + [\ell_2]P_n \end{aligned}$$

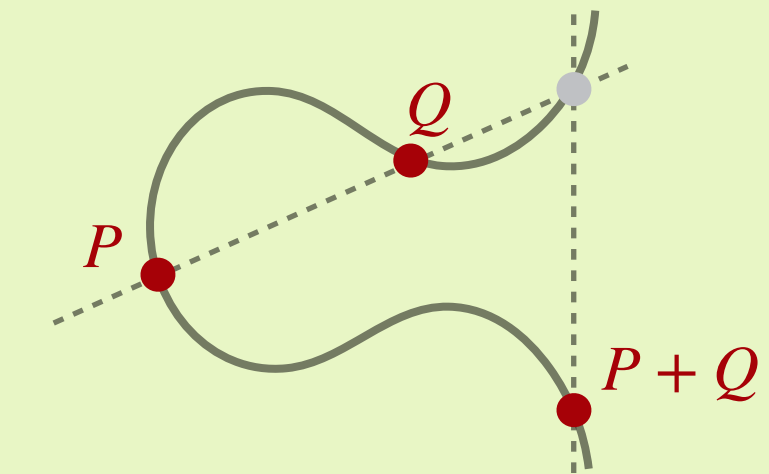


## Isogenies & Pairings

## elliptic curves in CSIDH

### supersingular elliptic curve

- has  $p + 1$  points in  $E(\mathbb{F}_p)$
- choose  $p$  so that  $p + 1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$
- this implies the rational points on  $E$  have orders that divide  $p + 1$



$$E : y^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_p$$

### points on such curves

We have that

$$E(\mathbb{F}_p) \cong \mathbb{Z}_4 \times \mathbb{Z}_{\ell_1} \times \mathbb{Z}_{\ell_2} \times \dots \times \mathbb{Z}_{\ell_n},$$

So think of a point  $P \in E(\mathbb{F}_p)$  as a sum of points  $P_i$  of order  $\ell_i$

$$P = P_0 + P_1 + P_2 + \dots + P_n$$

which shows how scalars  $[\lambda]$  with  $\lambda \in \mathbb{N}$  affect the torsion

$$\begin{aligned} [\ell_2]P &= [\ell_2]P_0 + [\ell_2]P_1 + [\ell_2]P_2 + \dots + [\ell_2]P_n \\ &= [\ell_2]P_0 + [\ell_2]P_1 + \mathcal{O} + \dots + [\ell_2]P_n \end{aligned}$$

the order of  $P$  is readable  
from the non-zero  $P_i$ 's

the torsion that  $P$  is *missing*  
are precisely the zero  $P_i$ 's

### full-torsion points

we call a point  $P \in E(\mathbb{F}_p)$  a **full-torsion point**  
if the order is  $p + 1$ , equivalently, all  $P_i$  are non-zero

## elliptic curves in CSIDH

supersingular elliptic curve

- has  $p + 1$  points in  $E(\mathbb{F}_p)$
- choose  $p$  so that  $p + 1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$
- this implies the rational points on  $E$  have orders that divide  $p + 1$

$$E : y^2 = x^3 + Ax^2 + x, \quad A \in \mathbb{F}_p$$

the order of  $\mathbf{P}$  is readable  
from the non-zero  $\mathbf{P}_i$ 's

the torsion that  $P$  is *missing*  
are precisely the zero  $P_i$ 's

## full-torsion points

we call a point  $P \in E(\mathbb{F}_p)$  a **full-torsion point**  
if the order is  $p + 1$ , equivalently, all  $P_i$  are non-zero

## torsion points and isogenies

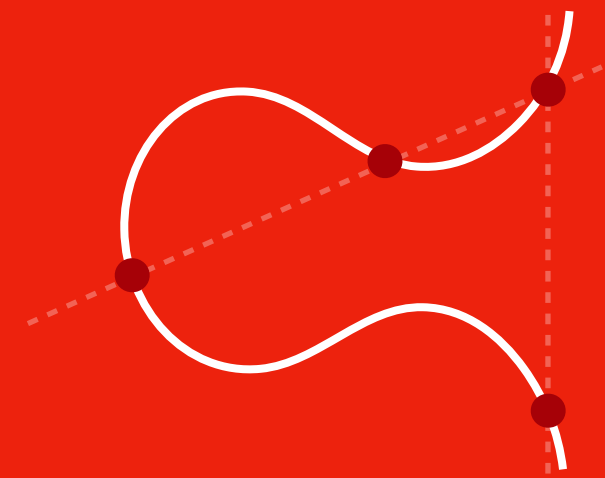
1. Any\* isogeny  $\varphi$  of degree  $N$ 
  - given by kernel of size  $N$
  - generated by point  $P$  of order  $N$

$$\text{Diagram} \xrightarrow[\deg 3 \cdot 5 \cdot 7]{\varphi} \text{Diagram}$$

2. Any\* isogeny  $\varphi$  of degree  $N = \prod \ell_i$ 
  - splits into sub-isogenies of degree  $\ell_i$
  - each generated by point  $P$  of order  $\ell_i$

3. Any\* isogeny  $\varphi$  of degree  $N = \prod \ell_i$ 
  - computed using one **full-torsion**  $P$
  - per  $\ell_i$  compute  $[\frac{p+1}{\ell_i}]P$  to get  $\ker(\varphi_i)$

$$\begin{aligned} P &= P_3 + P_5 + P_7 \in E(\mathbb{F}_p) \\ [5 \cdot 7]P &= P'_3 + \mathcal{O} + \mathcal{O} \in E(\mathbb{F}_p) \\ \varphi_1(P) &= \mathcal{O} + P'_5 + P'_7 \in E'(\mathbb{F}_p) \end{aligned}$$

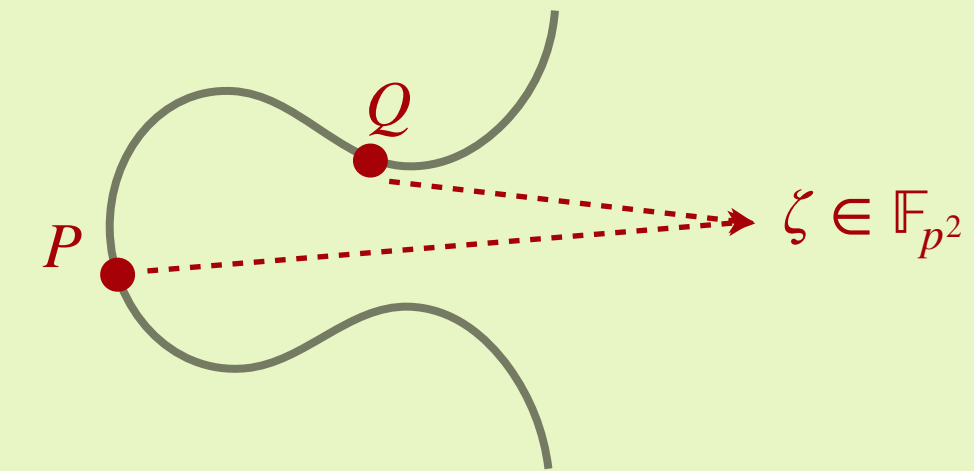


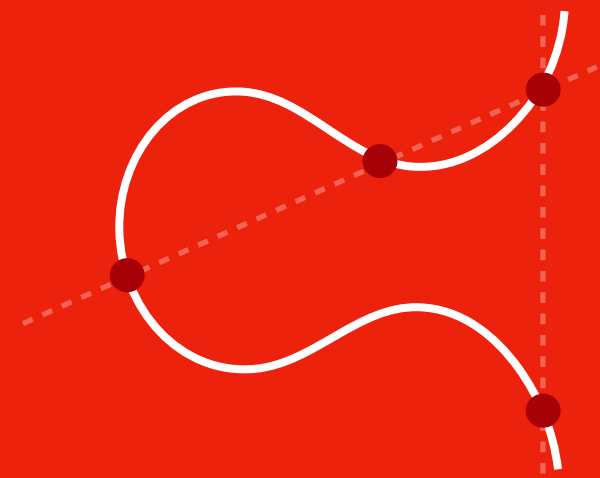
## Isogenies & Pairings

### the Tate pairing\*

#### bilinear pairing from torsion groups to fields

- choose a degree  $r$
- take point  $P$  of order  $r$  on  $E$ , that is  $P \in E(\mathbb{F}_{p^2})[r]$
- take point  $Q$  on  $E$  such that  $Q \in E(\mathbb{F}_{p^2})/rE(\mathbb{F}_{p^2})$
- then  $e_r(P, Q) = \zeta \in \mu_r$



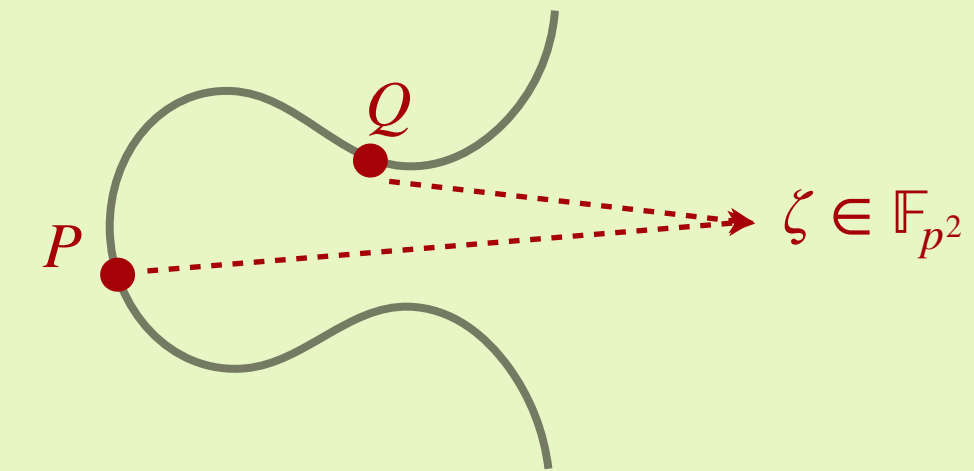


## Isogenies & Pairings

### the Tate pairing\*

#### bilinear pairing from torsion groups to fields

- choose a degree  $r$
- take point  $P$  of order  $r$  on  $E$ , that is  $P \in E(\mathbb{F}_{p^2})[r]$
- take point  $Q$  on  $E$  such that  $Q \in E(\mathbb{F}_{p^2})/rE(\mathbb{F}_{p^2})$
- then  $e_r(P, Q) = \zeta \in \mu_r$

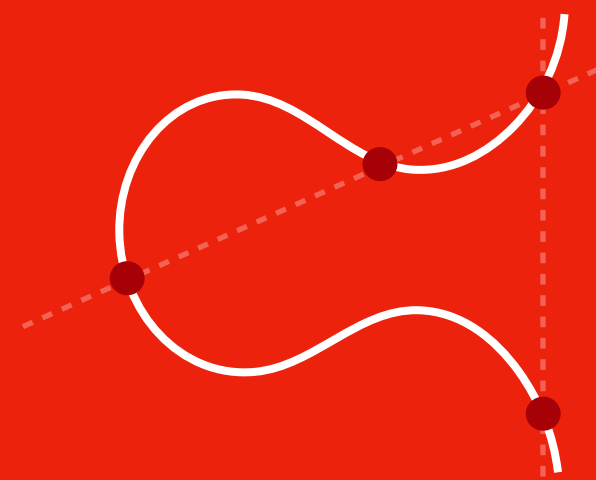


#### in our specific case

Formally, this pairing is abstract. Specifically in our case,  $p + 1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$  there is a nice interpretation of this pairing.

Choose  $r$  dividing  $p + 1$ , say  $r = \prod \ell_i = \frac{p+1}{4}$  then for  $P \in E(\mathbb{F}_p)$  we get

$$P = \mathcal{O} + P_1 + P_2 + \dots + P_n.$$

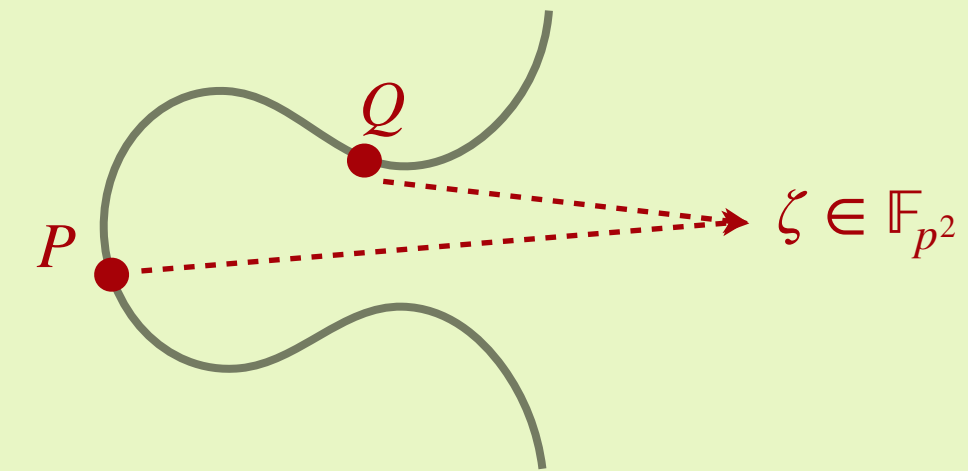


## Isogenies & Pairings

### the Tate pairing\*

#### bilinear pairing from torsion groups to fields

- choose a degree  $r$
- take point  $P$  of order  $r$  on  $E$ , that is  $P \in E(\mathbb{F}_{p^2})[r]$
- take point  $Q$  on  $E$  such that  $Q \in E(\mathbb{F}_{p^2})/rE(\mathbb{F}_{p^2})$
- then  $e_r(P, Q) = \zeta \in \mu_r$



#### in our specific case

Formally, this pairing is abstract. Specifically in our case,  $p+1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$  there is a nice interpretation of this pairing.

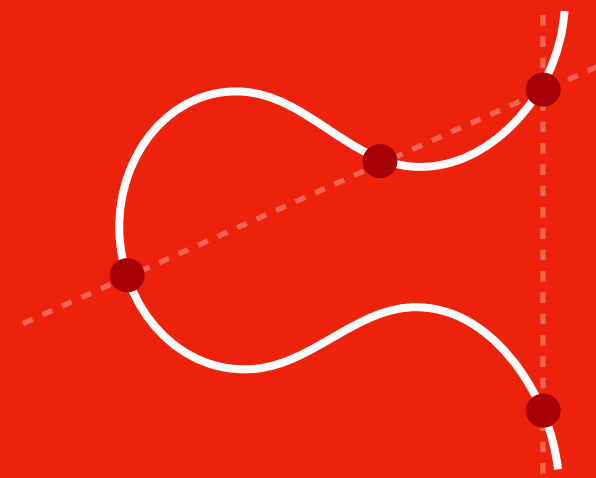
Choose  $r$  dividing  $p+1$ , say  $r = \prod \ell_i = \frac{p+1}{4}$  then for  $P \in E(\mathbb{F}_p)$  we get

$$P = \mathcal{O} + P_1 + P_2 + \dots + P_n.$$

For  $Q \in E(\mathbb{F}_p)$ , we have equivalence by elements  $R$  in  $rE(\mathbb{F}_{p^2})$ . In this scenario, we can think of such elements  $R$  as  $R_0 + \mathcal{O} + \dots + \mathcal{O}$ , which implies  $Q \sim Q'$  whenever

$$Q = Q_0 + Q_1 + Q_2 + \dots + Q_n \sim Q' = Q'_0 + Q_1 + Q_2 + \dots + Q_n$$

In this specific scenario, we can think of  $Q$  as the elements  $\mathcal{O} + Q_1 + \dots + Q_n$

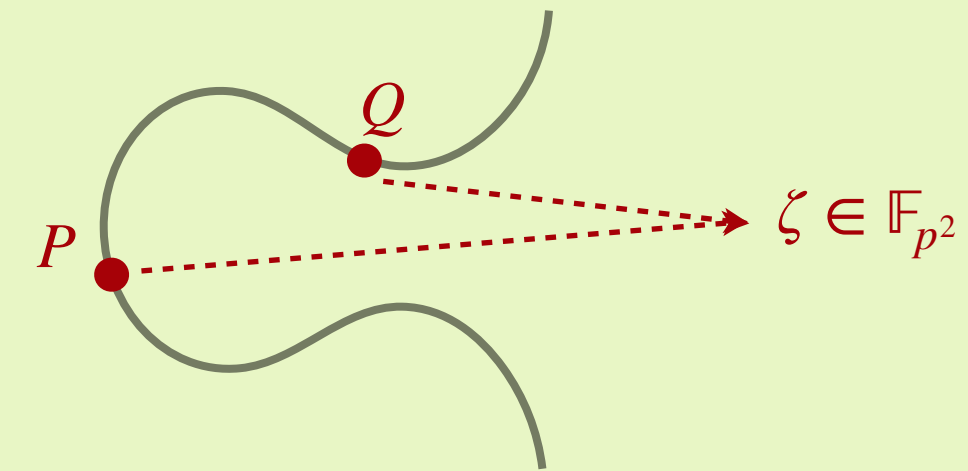


## Isogenies & Pairings

### the Tate pairing\*

#### bilinear pairing from torsion groups to fields

- choose a degree  $r$
- take point  $P$  of order  $r$  on  $E$ , that is  $P \in E(\mathbb{F}_{p^2})[r]$
- take point  $Q$  on  $E$  such that  $Q \in E(\mathbb{F}_{p^2})/rE(\mathbb{F}_{p^2})$
- then  $e_r(P, Q) = \zeta \in \mu_r$



#### in our specific case

Formally, this pairing is abstract. Specifically in our case,  $p+1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$  there is a nice interpretation of this pairing.

Choose  $r$  dividing  $p+1$ , say  $r = \prod \ell_i = \frac{p+1}{4}$  then for  $P \in E(\mathbb{F}_p)$  we get

$$P = \mathcal{O} + P_1 + P_2 + \dots + P_n.$$

For  $Q \in E(\mathbb{F}_p)$ , we have equivalence by elements  $R$  in  $rE(\mathbb{F}_{p^2})$ . In this scenario, we can think of such elements  $R$  as  $R_0 + \mathcal{O} + \dots + \mathcal{O}$ , which implies  $Q \sim Q'$  whenever

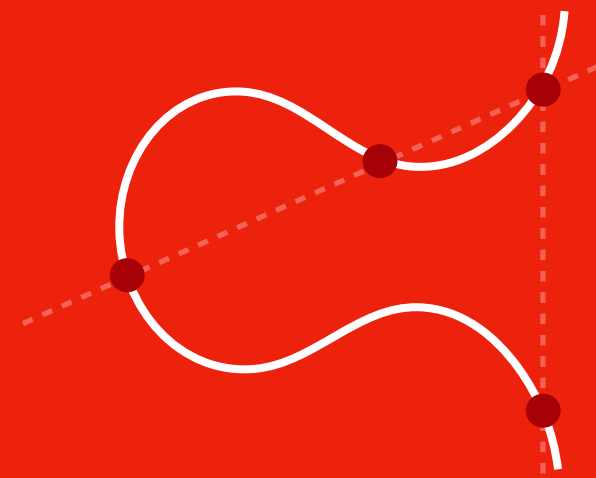
$$Q = Q_0 + Q_1 + Q_2 + \dots + Q_n \sim Q' = Q'_0 + Q_1 + Q_2 + \dots + Q_n$$

In this specific scenario, we can think of  $Q$  as the elements  $\mathcal{O} + Q_1 + \dots + Q_n$



#### problem

If we pick  $P, Q \in E(\mathbb{F}_p)$ ,  
then  $Q$  is a multiple of  $P$ .  
Then  $e_r(P, Q) = 1$

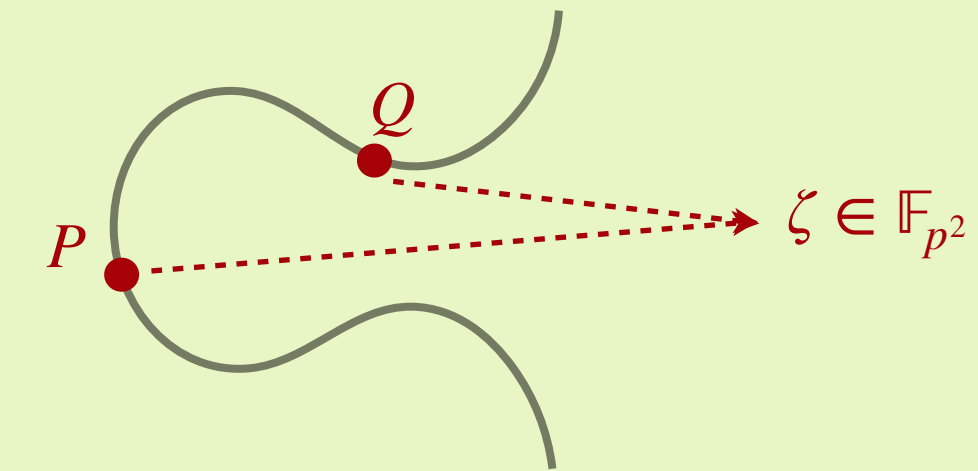


## Isogenies & Pairings

### the Tate pairing\*

#### bilinear pairing from torsion groups to fields

- choose a degree  $r$
- take point  $P$  of order  $r$  on  $E$ , that is  $P \in E(\mathbb{F}_{p^2})[r]$
- take point  $Q$  on  $E$  such that  $Q \in E(\mathbb{F}_{p^2})/rE(\mathbb{F}_{p^2})$
- then  $e_r(P, Q) = \zeta \in \mu_r$



#### in our specific case

Formally, this pairing is abstract. Specifically in our case,  $p+1 = 4 \cdot \ell_1 \cdot \ell_2 \cdot \dots \cdot \ell_n$  there is a nice interpretation of this pairing.

Choose  $r$  dividing  $p+1$ , say  $r = \prod \ell_i = \frac{p+1}{4}$  then for  $P \in E(\mathbb{F}_p)$  we get

$$P = \mathcal{O} + P_1 + P_2 + \dots + P_n.$$

For  $Q \in E(\mathbb{F}_p)$ , we have equivalence by elements  $R$  in  $rE(\mathbb{F}_{p^2})$ . In this scenario, we can think of such elements  $R$  as  $R_0 + \mathcal{O} + \dots + \mathcal{O}$ , which implies  $Q \sim Q'$  whenever

$$Q = Q_0 + Q_1 + Q_2 + \dots + Q_n \sim Q' = Q'_0 + Q_1 + Q_2 + \dots + Q_n$$

In this specific scenario, we can think of  $Q$  as the elements  $\mathcal{O} + Q_1 + \dots + Q_n$



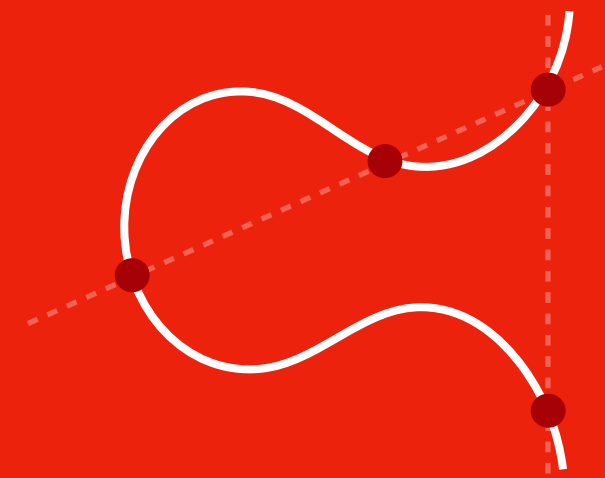
#### problem

If we pick  $P, Q \in E(\mathbb{F}_p)$ , then  $Q$  is a multiple of  $P$ . Then  $e_r(P, Q) = 1$



#### solution

Work over  $E[r] \subseteq E(\mathbb{F}_{p^2})$ . In our specific case, just use  $Q$  on the *twist*

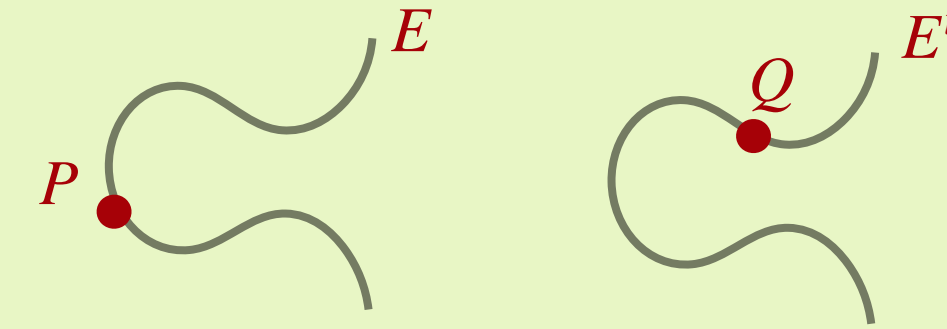


## Isogenies & Pairings

### the twist of $E$

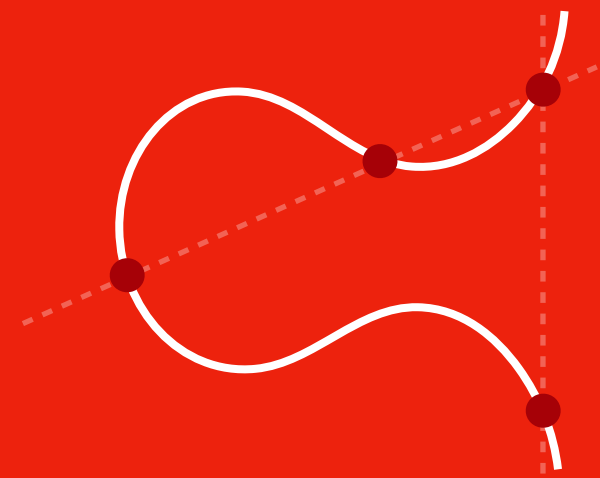
#### Twist over $\mathbb{F}_p$ of supersingular curve $E$

- a curve  $E^t$  with  $p + 1$  points over  $\mathbb{F}_p$
- isomorphic to a specific subset of  $E(\mathbb{F}_{p^2})$
- used in CSIDH to “move backwards” in graph
- want  $P \in E(\mathbb{F}_p)$  and  $Q \in E^t(\mathbb{F}_p)$ , both full order





1

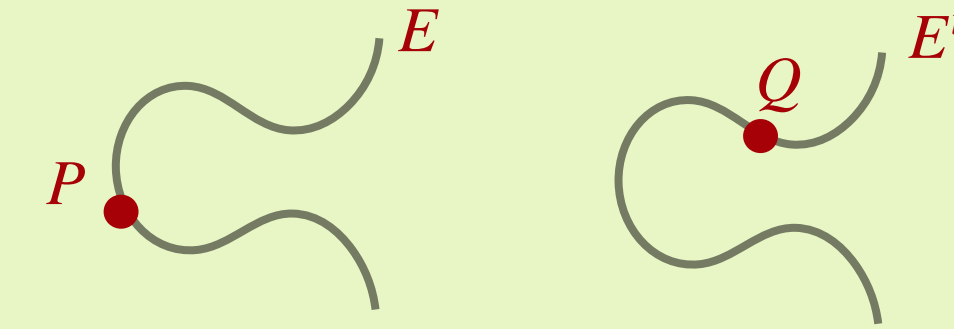


## Isogenies & Pairings

### the twist of $E$

#### Twist over $\mathbb{F}_p$ of supersingular curve $E$

- a curve  $E^t$  with  $p + 1$  points over  $\mathbb{F}_p$
- isomorphic to a specific subset of  $E(\mathbb{F}_{p^2})$
- used in CSIDH to “move backwards” in graph
- want  $P \in E(\mathbb{F}_p)$  and  $Q \in E^t(\mathbb{F}_p)$ , both full order



1

consider  $P$  and  $Q$  as

$$P = P_0 + P_1 + \dots + P_n$$

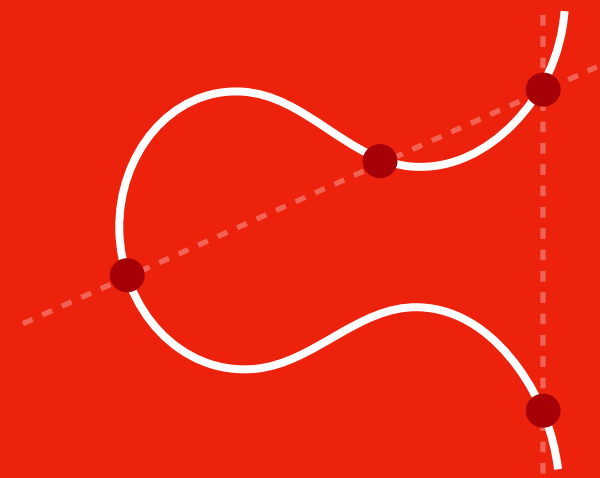
$$Q = Q_0 + Q_1 + \dots + Q_n$$

2

let  $r = p + 1$

Tate pairing  $e_r(P, Q)$  captures  
where **both**  $P_i, Q_i \neq \mathcal{O}$

1

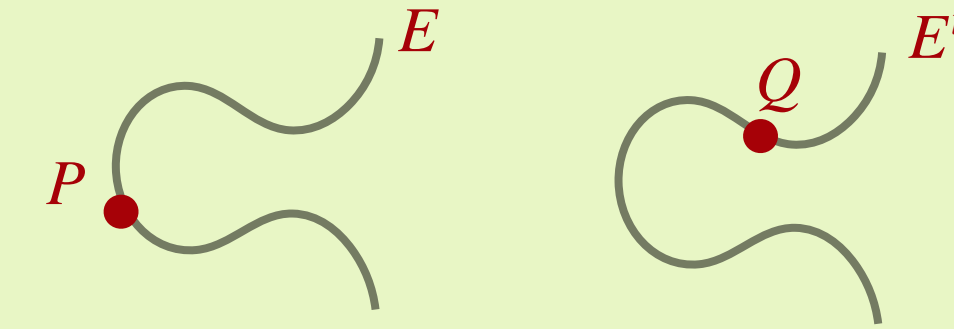


## Isogenies & Pairings

### the twist of $E$

#### Twist over $\mathbb{F}_p$ of supersingular curve $E$

- a curve  $E'$  with  $p + 1$  points over  $\mathbb{F}_p$
- isomorphic to a specific subset of  $E(\mathbb{F}_{p^2})$
- used in CSIDH to “move backwards” in graph
- want  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , both full order



1

consider  $P$  and  $Q$  as

$$P = P_0 + P_1 + \dots + P_n$$

$$Q = Q_0 + Q_1 + \dots + Q_n$$

2

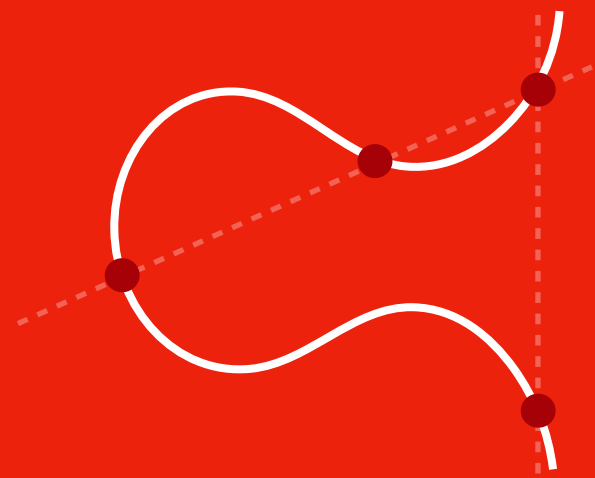
let  $r = p + 1$

Tate pairing  $e_r(P, Q)$  captures  
where **both**  $P_i, Q_i \neq \mathcal{O}$

### crucial lemma

Let  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$ , and  $r = p + 1$ . Let  $\zeta = e_r(P, Q) \in \mathbb{F}_{p^2}$ .

Then  $\zeta$  is an  $r$ -th root of unity, whose order is precisely  
gcd of order of  $P$ , order of  $Q$

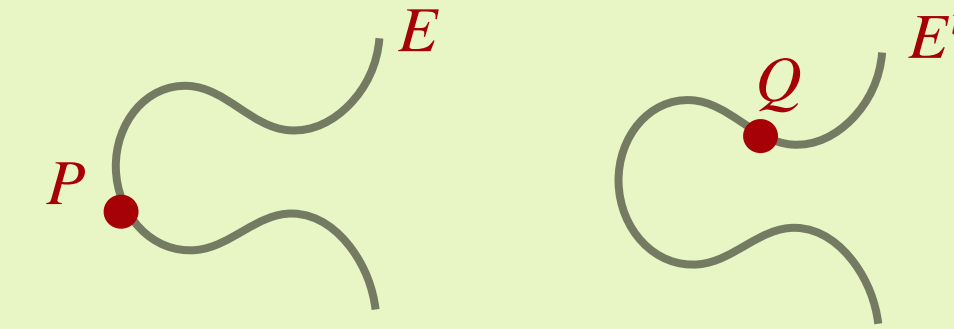


## Isogenies & Pairings

### the twist of $E$

#### Twist over $\mathbb{F}_p$ of supersingular curve $E$

- a curve  $E'$  with  $p + 1$  points over  $\mathbb{F}_p$
- isomorphic to a specific subset of  $E(\mathbb{F}_{p^2})$
- used in CSIDH to “move backwards” in graph
- want  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , both full order



1

consider  $P$  and  $Q$  as

$$P = P_0 + P_1 + \dots + P_n$$

$$Q = Q_0 + Q_1 + \dots + Q_n$$

2

let  $r = p + 1$

Tate pairing  $e_r(P, Q)$  captures  
where **both**  $P_i, Q_i \neq \mathcal{O}$

### crucial lemma

Let  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$ , and  $r = p + 1$ . Let  $\zeta = e_r(P, Q) \in \mathbb{F}_{p^2}$ .

Then  $\zeta$  is an  $r$ -th root of unity, whose order is precisely  
gcd of order of  $P$ , order of  $Q$

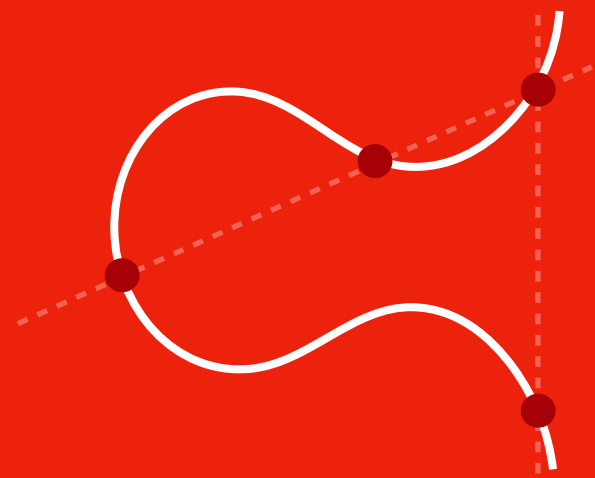
#### example

If  $P$  and  $Q$  both full torsion,  
then  $\zeta$  has order  $r = p + 1$

#### example

If  $P$  has order 5, and  $Q$  has  
order 15, then  $\zeta$  has order 5

1

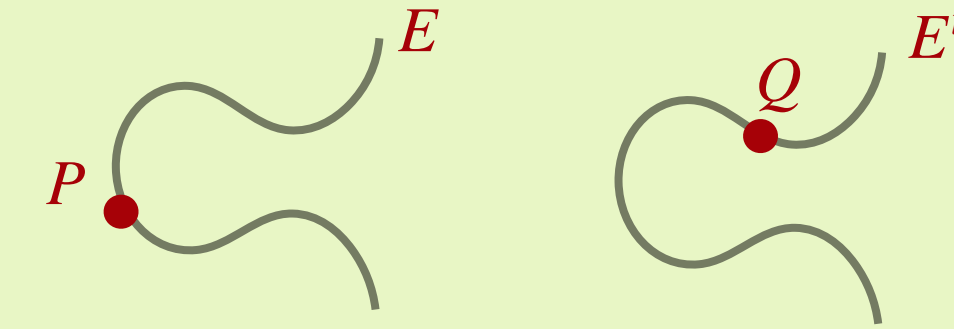


## Isogenies & Pairings

### the twist of $E$

#### Twist over $\mathbb{F}_p$ of supersingular curve $E$

- a curve  $E^t$  with  $p + 1$  points over  $\mathbb{F}_p$
- isomorphic to a specific subset of  $E(\mathbb{F}_{p^2})$
- used in CSIDH to “move backwards” in graph
- want  $P \in E(\mathbb{F}_p)$  and  $Q \in E^t(\mathbb{F}_p)$ , both full order



1

consider  $P$  and  $Q$  as

$$P = P_0 + P_1 + \dots + P_n$$

$$Q = Q_0 + Q_1 + \dots + Q_n$$

2

let  $r = p + 1$

Tate pairing  $e_r(P, Q)$  captures  
where **both**  $P_i, Q_i \neq \mathcal{O}$

### crucial lemma

Let  $P \in E(\mathbb{F}_p)$ ,  $Q \in E^t(\mathbb{F}_p)$ , and  $r = p + 1$ . Let  $\zeta = e_r(P, Q) \in \mathbb{F}_{p^2}$ .

Then  $\zeta$  is an  $r$ -th root of unity, whose order is precisely  
gcd of order of  $P$ , order of  $Q$

#### example

If  $P$  and  $Q$  both full torsion,  
then  $\zeta$  has order  $r = p + 1$

#### example

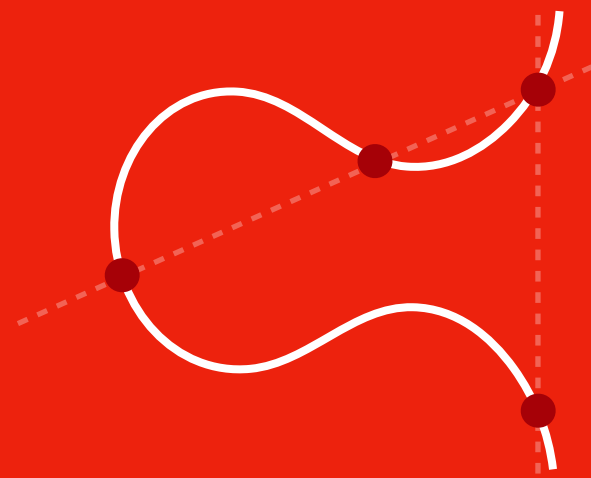
If  $P$  has order 5, and  $Q$  has  
order 15, then  $\zeta$  has order 5

!

#### notice

Curve arithmetic is slow!  
Field arithmetic is fast!!  
(more than factor 6)

1

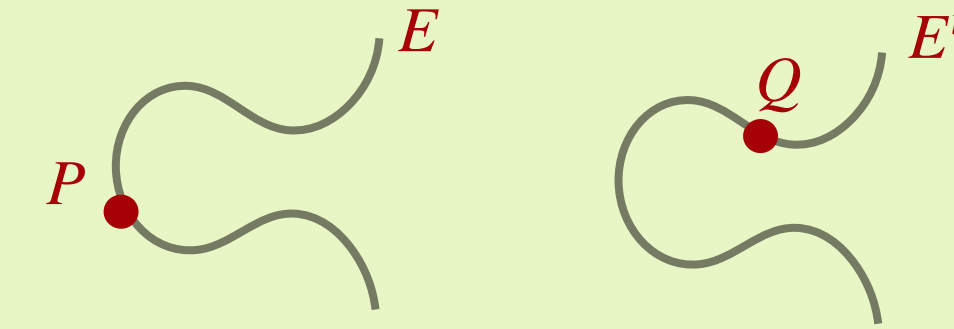


## Isogenies & Pairings

### the twist of $E$

#### Twist over $\mathbb{F}_p$ of supersingular curve $E$

- a curve  $E'$  with  $p + 1$  points over  $\mathbb{F}_p$
- isomorphic to a specific subset of  $E(\mathbb{F}_{p^2})$
- used in CSIDH to “move backwards” in graph
- want  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , both full order



1

consider  $P$  and  $Q$  as

$$P = P_0 + P_1 + \dots + P_n$$

$$Q = Q_0 + Q_1 + \dots + Q_n$$

2

let  $r = p + 1$

Tate pairing  $e_r(P, Q)$  captures  
where **both**  $P_i, Q_i \neq \mathcal{O}$

### crucial lemma

Let  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$ , and  $r = p + 1$ . Let  $\zeta = e_r(P, Q) \in \mathbb{F}_{p^2}$ .

Then  $\zeta$  is an  $r$ -th root of unity, whose order is precisely  
gcd of order of  $P$ , order of  $Q$

### example

If  $P$  and  $Q$  both full torsion,  
then  $\zeta$  has order  $r = p + 1$

### example

If  $P$  has order 5, and  $Q$  has  
order 15, then  $\zeta$  has order 5

!

### notice

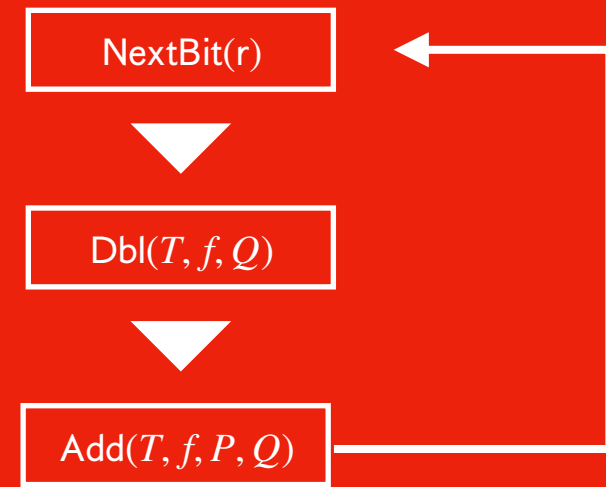
Curve arithmetic is slow!  
Field arithmetic is fast!!  
(more than factor 6)

✓

### core idea

Pick random  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$   
Instead of using curve arithmetic  
to compute their orders, use  $\zeta$   
to compute the overlap in orders!

**Pairings are quite slow**

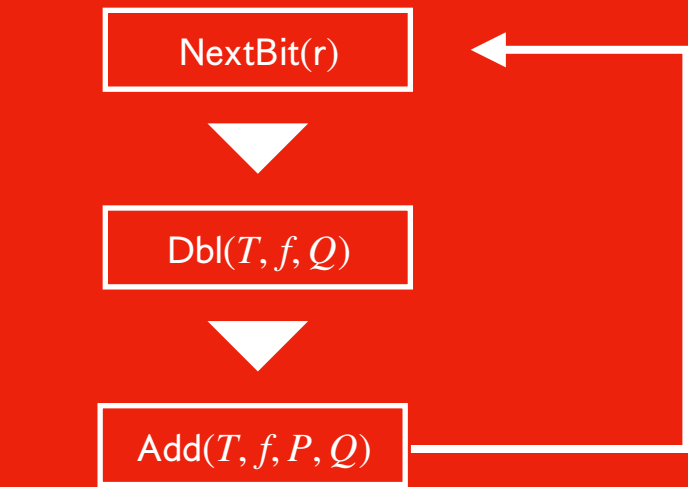


**Speeding-up  
general pairings**



**core idea**

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



**Speeding-up  
general pairings**

## pairing crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **pairings** with

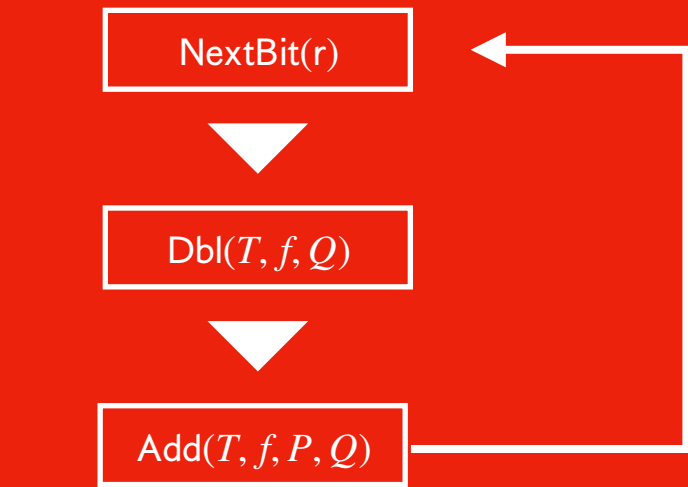
Computing  $e(P, Q)$   
is quite **fast**!



## core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!





**Speeding-up  
general pairings**

### pairing crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **pairings** with

Computing  $e(P, Q)$   
is quite **fast**!

### isogeny crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **isogenies** with

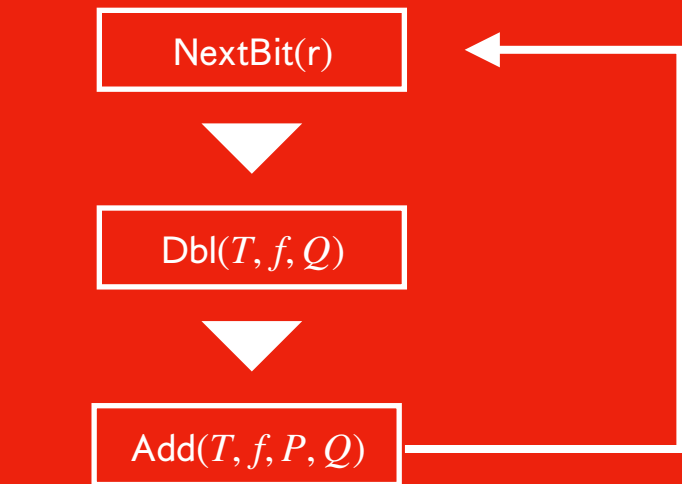
These are mediocre curves,  
and definitely bad primes,  
to do **pairings** with

Computing  $e(P, Q)$   
seems way too **slow**!



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



**Speeding-up  
general pairings**

### pairing crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **pairings** with

Computing  $e(P, Q)$   
is quite **fast**!



### isogeny crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **isogenies** with

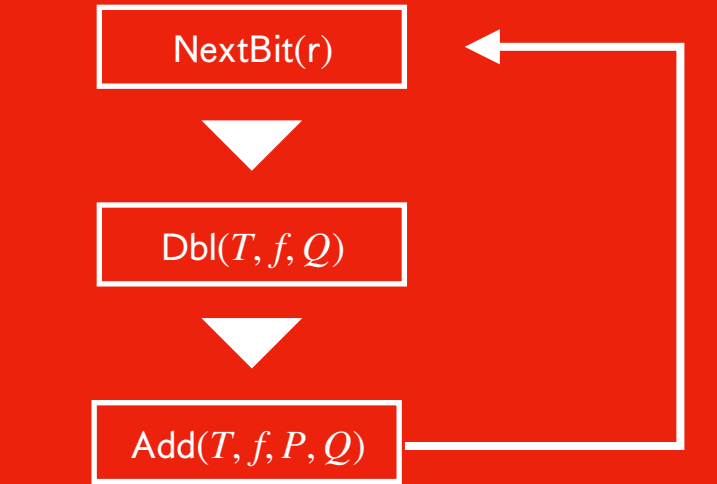
These are mediocre curves,  
and definitely bad primes,  
to do **pairings** with

Computing  $e(P, Q)$   
seems way too **slow**!



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



**Speeding-up  
general pairings**

### pairing crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **pairings** with

Computing  $e(P, Q)$   
is quite **fast**!



### isogeny crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **isogenies** with

These are mediocre curves,  
and definitely bad primes,  
to do **pairings** with

Computing  $e(P, Q)$   
seems way too **slow**!



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!

### MAIN RESULTS

1

make pairings  
great again



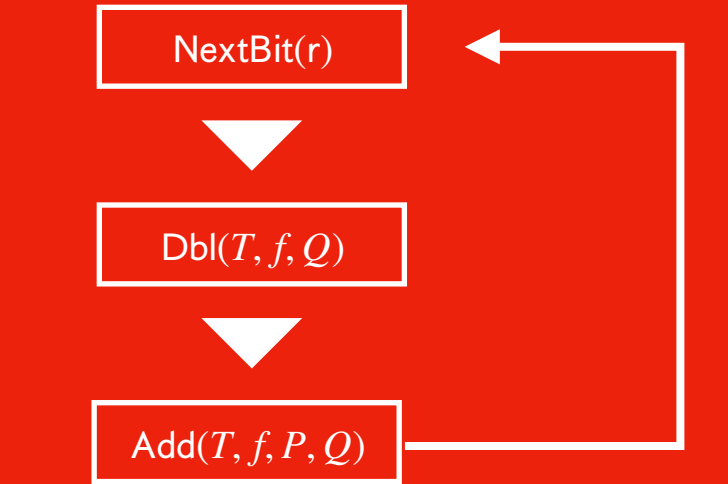
2

apply core idea



3

faster isogeny  
algorithms!



**Speeding-up  
general pairings**

### pairing crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **pairings** with

Computing  $e(P, Q)$   
is quite **fast**!



### isogeny crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **isogenies** with

These are mediocre curves,  
and definitely bad primes,  
to do **pairings** with

Computing  $e(P, Q)$   
seems way too **slow**!



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!

### MAIN RESULTS

1

make pairings  
great again

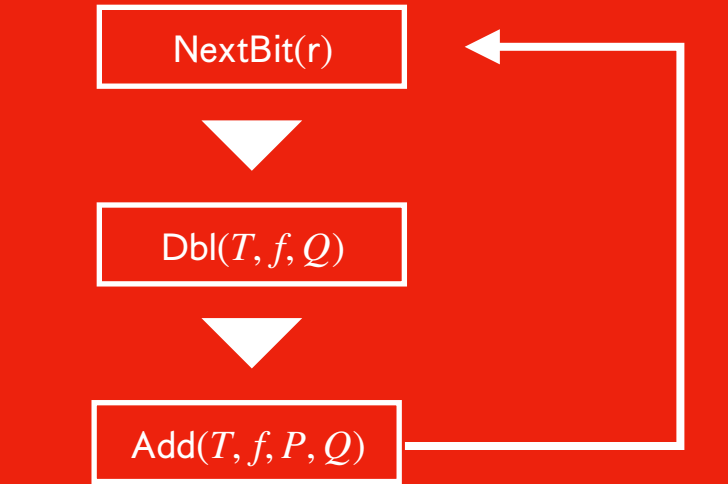
2

apply core idea

3

faster isogeny  
algorithms!

first  
this



**Speeding-up  
general pairings**

### pairing crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **pairings** with

Computing  $e(P, Q)$   
is quite **fast**!



### isogeny crypto

Choose a “nice” curve  $E$ ,  
Choose a “nice” prime  $p$ ,  
to do **isogenies** with

These are mediocre curves,  
and definitely bad primes,  
to do **pairings** with

Computing  $e(P, Q)$   
seems way too **slow**!



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!

### MAIN RESULTS

1

make pairings  
great again

2

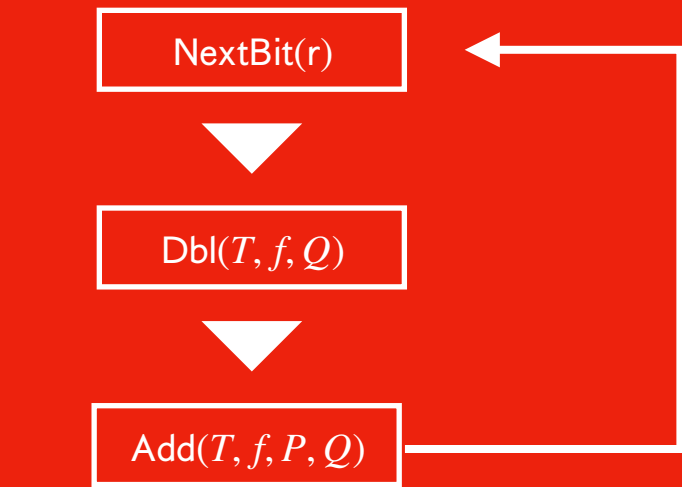
apply core idea

3

faster isogeny  
algorithms!

first  
this

then  
this



**Speeding-up  
general pairings**



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



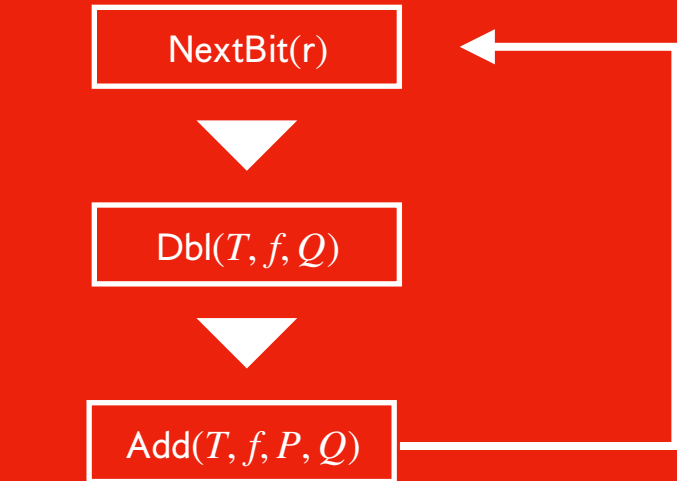
### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out



## Speeding-up general pairings



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out

0

take some literature

36 Chapter 3. Division

$Q$  with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $\ell' : y = \lambda_2 x + v_2$   
be the tangent at  $R$  with divisor  $(f') = 2(R) + (-2(O)) - 3(O)$ . The divisor of

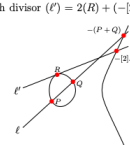


Figure 3.3. Two functions  $f$  and

the function  $\ell_{\text{con}} = \ell' - \ell$  is  $(\ell_{\text{con}}) = (f) + (f') - (P) + (Q) + (-P + Q) - 2(R) - (-2(O)) - 3(O)$ . Notes  
that  $\ell$  at  $O$ , projecting  $(f' - \ell_{\text{con}})$  gives points  
to any zeros or poles at  $Z = 0$ . Suppose we want  
on  $E$ , and we multiplied out  $(y - \lambda_2 x - v_2)(y - \lambda_2 x - v_2 + v_2)$   
 $x^2 + ax + b$  and wrote  $y = \lambda_2 x + v_2$ .  
try and depict this function over all the picture  
purposes also show how the function (on  $E$ ) looks  
 $E$ , where the substitution  $y^2 = x^2 + ax + b$  is not 1

man in the conventional set-up algorithm as it is given in Algorithm 3 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $\mathbb{F}_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

**Algorithm 3** Right-to-left version of Miller's algorithm with postponed addition steps for even  $k$  and ate-like pairings  
**Input:**  $Q \in G_2, P \in G_1, m = (m_1, m_2, \dots, m_k)$   
**Output:**  $f_{m,P,Q}$  representing a class in  $\mathbb{F}_p^*/\langle Q \rangle$   
1:  $R \leftarrow Q, f \leftarrow 1, j \leftarrow 0$   
2: **for** from 0 to  $k-1$  **do**  
3: **if**  $(m_j \neq 1)$  **then**  
4:  $A_j[1] \leftarrow R, A_j[2] \leftarrow f, j \leftarrow j+1$   
5: **end if**  
6:  $f \leftarrow f^2 \cdot \text{twisted}(P)$ ,  $R \leftarrow [2]R$   
7: **end for**  
8:  $R \leftarrow A_j[0], f \leftarrow A_j[0]$   
9: **for**  $(j = 1; j \leq M(m) - 1; j++)$  **do**  
10:  $f \leftarrow f \cdot A_j[1] \cdot \text{twisted}(P)$ ,  $R \leftarrow R$   
11: **end for**  
12: **return**  $f$

and so

$\ell(Q) = \theta_{RQ} - \eta_P \left( \frac{\alpha_{2k-1}}{\alpha_{2k} - x_P} (x_Q - x_P) + 1 \right)$

Writing this as  $\theta_{RQ} + \eta_P \ell_2$ , we have  $f_{m,P,Q} = \alpha_{2k+1,P} + \theta_{P,RQ} \beta_{2k+1,P}$  where

$\alpha_{2k+1} = (\ell_2^k + A_{2P} + B) \beta_{2k,P} \ell_2 - (\ell_2^k + A_{2Q} + B) \beta_{2k,P} / (\ell_2 Q - x_{2k+1,P})$

and

$\beta_{2k+1,P} = (\alpha_{2k,P} + \beta_{2k,P} \ell_2) / (\ell_2 Q - x_{2k+1,P})$

1

implement all tricks  
that apply

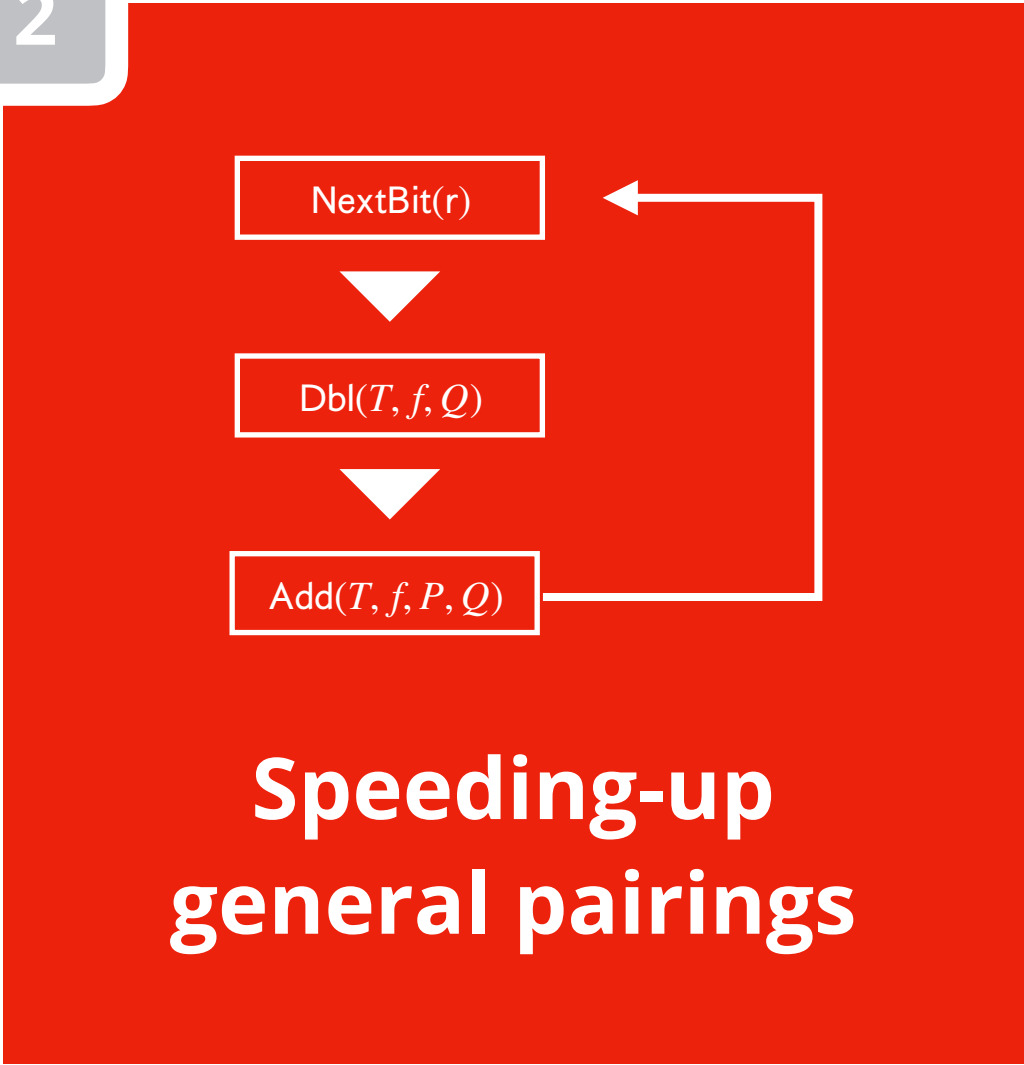
2

benchmark speed  
and finetune

3

fast pairings

2



**!**

**general notice**

Computing pairings fast is quite technical.  
Better suited for papers than slides

✓

**core idea**

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E^t(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!

✓

**general approach**

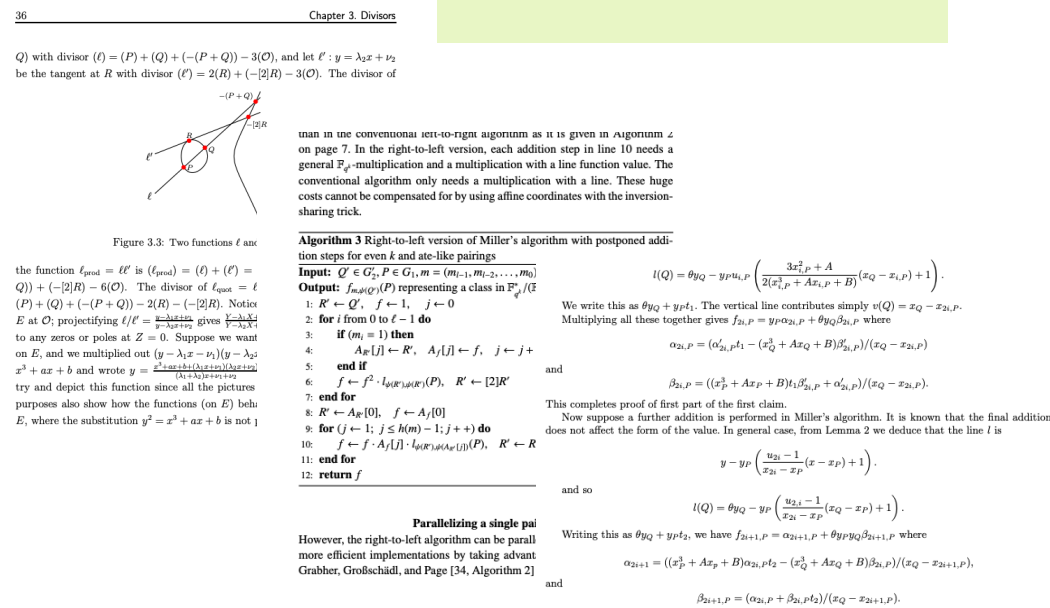
Instead I describe the general approach,  
and leave all details out

# 0

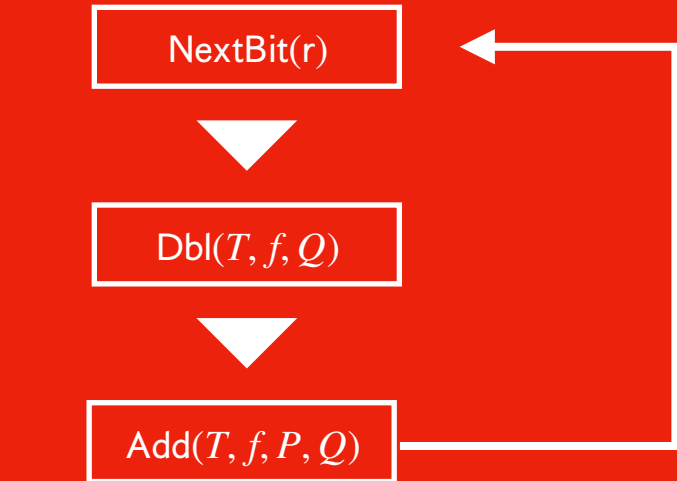
## take some literature

1

implement all tricks  
that apply







## Speeding-up general pairings



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out

0

take some literature

1

implement all tricks  
that apply

2

benchmark speed  
and finetune

3

fast pairings

36 Chapter 3. Division

$Q$  with divisor  $(f) = (P) + (Q) + (- (P + Q)) - 3(O)$ , and let  $\ell': y = \lambda_2 x + v_2$  be the tangent at  $R$  with divisor  $(f') = 2(R) + (-2(O)) - 3(O)$ . The divisor of  $-x^2 + a_2 x$

Figure 3.3. Two functions  $f$  and  $f'$  on the function field  $\mathbb{F} = \mathbb{F}_p(x, y)$  of the elliptic curve  $E: y^2 = x^3 + a_2 x + a_3$ . The divisor of  $f$  is  $(f) = (P) + (Q) + (- (P + Q)) - 3(O)$ . The divisor of  $f'$  is  $(f') = 2(R) + (-2(O)) - 3(O)$ . Notes:  $E$  at  $O$ , projecting  $(x, y) \mapsto (x, y/x)$  gives points  $P, Q, R$  on  $E$ , and we multiplied out  $(y - \lambda_2 x - v_2)(y - \lambda_2 x - v_2) = x^3 + a_2 x + a_3$  and wrote  $y = \lambda_2 x + v_2$ . The substitution  $y^2 = x^3 + a_2 x + a_3$  is not 1

man in the conventional set-up algorithm as it is given in Algorithm 3 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $\mathbb{F}_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

**Algorithm 3** Right-to-left version of Miller's algorithm with postponed addition steps for even  $k$  and ate-like pairings

**Input:**  $Q \in G_2, P \in G_1, m = (m_1, m_2, \dots, m_k)$

**Output:**  $f_{m,P,Q}$  representing a class in  $\mathbb{F}_p^*/\mathbb{Q}^*$

- $R \leftarrow Q, f \leftarrow 1, j \leftarrow 0$
- for** from 0 to  $k-1$  **do**
- if**  $(m_j \neq 1)$  **then**
- $A_j[1] \leftarrow R, A_j[2] \leftarrow f, j \leftarrow j+1$
- end if**
- $f \leftarrow f^2 \cdot \text{norm}_{\mathbb{F}_p}(P)$ ,  $R \leftarrow [2]R$
- end for**
- $R \leftarrow A_j[0], f \leftarrow A_j[0]$
- for**  $j = 1; j \leq M(m) - 1; j++$  **do**
- $f \leftarrow f \cdot A_j[1] \cdot \text{norm}_{\mathbb{F}_p}(P)$ ,  $R \leftarrow R$
- end for**
- return**  $f$

and so

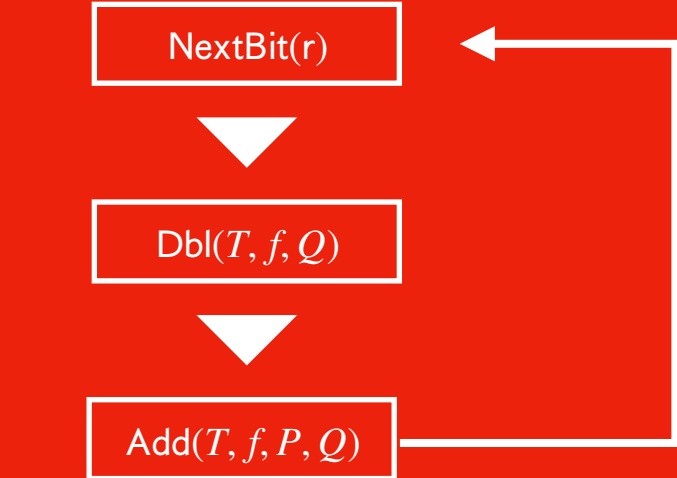
**Paralleling a single pair**

However, the right-to-left algorithm can be parallelized more efficiently by taking advantage of the fact that the line function value  $f_{m,P,Q}$  can be written as  $f_{m,P,Q} = \prod_{i=1}^k f_{m_i,P,Q}$  where

$$f_{m_i,P,Q} = ((\ell_i^2 + A_{\ell_i} + B)\beta_{m_i,P,Q} - (\ell_i^2 + A_{\ell_i} + B)\beta_{m_i,P,Q})/(\ell_i Q - x_{m_i,P,Q})$$

and

$$\beta_{m_i,P,Q} = (m_i P + \beta_{m_i,P,Q})/(\ell_i Q - x_{m_i,P,Q})$$



## Speeding-up general pairings



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out

0

take some literature

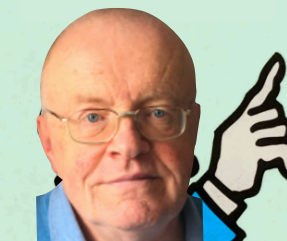
1

2

3

fast pairings

BACK TO



STEP 0

36 Chapter 3. Division

$Q$  with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $\ell' : y = \lambda_2 x + v_2$  be the tangent at  $R$  with divisor  $(\ell') = 2(R) + (-2(R) - 3(O))$ . The divisor of

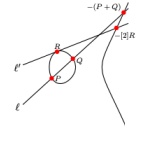


Figure 3.3.2: Two functions  $f$  and  $f'$

use in the conventional set-to-right algorithm as it is given in algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $\mathbb{F}_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

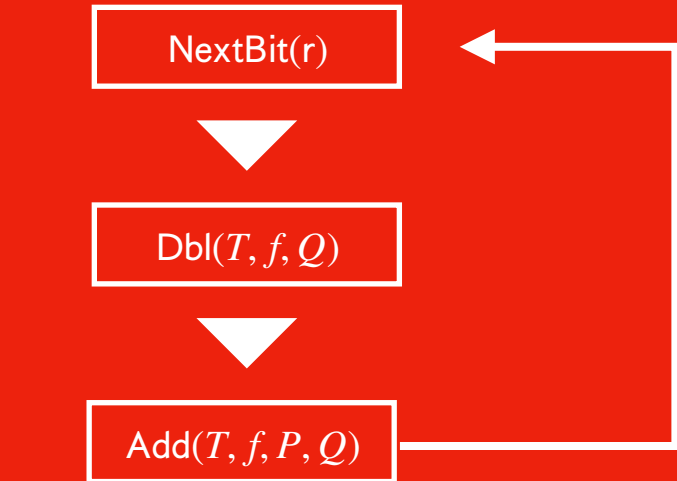
**Algorithm 3** Right-to-left version of Miller's algorithm with postponed additions

use in the conventional set-to-right algorithm as it is given in algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $\mathbb{F}_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

**Algorithm 3** Right-to-left version of Miller's algorithm with postponed additions

**Parallelizing a single pm**  
However, the right-to-left algorithm can be parallelized more efficiently by taking advantage of the fact that the line  $\ell$  is the same for all points  $P_i$ . This is shown in [14, Algorithm 2].

**Parallelizing a single pm**  
However, the right-to-left algorithm can be parallelized more efficiently by taking advantage of the fact that the line  $\ell$  is the same for all points  $P_i$ . This is shown in [14, Algorithm 2].



## Speeding-up general pairings



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out

0

take some literature

36 Chapter 3. Divisors

$Q$  with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $\ell' : y = \lambda_2 x + v_2$  be the tangent at  $R$  with divisor  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of

$-f' + Q_2$

is

$-P + Q_2$

use in the conventional set-to-right algorithm as it is given in algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $F_p$  multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Figure 3.3: Two functions  $\ell$  and

**Algorithm 3** Right-to-left version of Miller's algorithm with postponed additions

**Input:**  $Q \in G_p, P \in G_p, m = (m_{i-1}, \dots, m_1, m_0)$  class in  $\mathbb{F}_p^*/\langle Q \rangle$

**Output:**  $f_{m,Q}(P)$  representing a class in  $\mathbb{F}_p^*/\langle Q \rangle$

1:  $R \leftarrow Q, f \leftarrow 1, j \leftarrow 0$

2: **for**  $i$  from 0 to  $\ell - 1$  **do**

3:   **if**  $(m_i = 1)$  **then**

4:      $A_1[f] \leftarrow R, A_1[f] \leftarrow f, j \leftarrow j + 1$

5:   **end if**

6:    $f \leftarrow f^2 \cdot \ell_{m,P}(P), R \leftarrow [2]R$

7: **end for**

8:  $R \leftarrow A_1[0], f \leftarrow A_1[0]$

9: **for**  $j = 1, \dots, \ell - 1$  **do**

10:    $f \leftarrow f \cdot A_1[f] \cdot \ell_{m,P}(P), R \leftarrow R$

11: **end for**

12: **return**  $f$

and so

$l(Q) = \theta_{Q, P} - y_{m, P} \left( \frac{y_{Q, P} - 1}{x_{Q, P} - x_P} (x_Q - x_P) + 1 \right)$ .

Writing this as  $\theta_{Q, P} + y_{m, P}$ , we have  $\theta_{Q, P} = \alpha_{Q, P} + \beta_{Q, P} y_{m, P}$ , where

$\alpha_{Q, P} = ((x_P^2 + Ax_P + By_P)(x_Q - x_P) + \beta_{Q, P} y_{m, P}) / (x_Q - x_P)$ , and

$\beta_{Q, P} = (y_{m, P} + \beta_{Q, P} y_{m, P}) / (x_Q - x_P)$ .

1

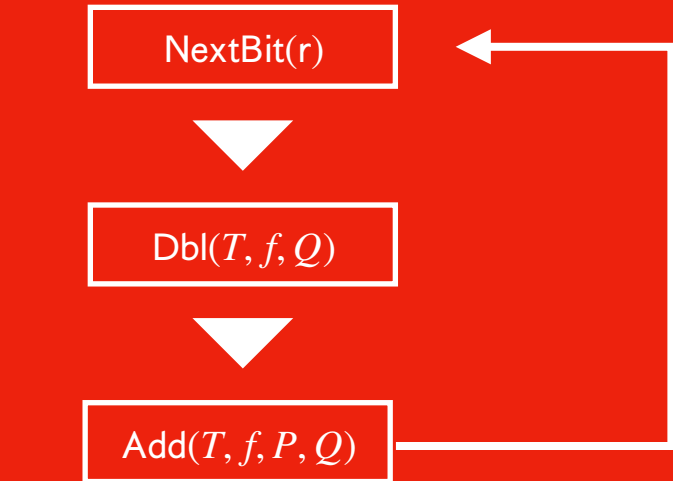
implement all tricks  
that apply

2

benchmark speed  
and finetune

3

fast pairings



## Speeding-up general pairings



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out

0

take some literature

1

implement all tricks  
that apply

2

benchmark speed  
and finetune

3

fast pairings

$Q$  with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $\ell' : y = \lambda_2 x + v_2$  be the tangent at  $R$  with divisor  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of

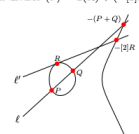


Figure 3.3.2: Two functions  $f$  and  $f'$

use in the conventional set-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $F_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

**Algorithm 3** Right-to-left version of Miller's algorithm with postponed additions

**Input:**  $Q \in G, P \in E, m \in \mathbb{Z}$ ,  $m = (m_{k-1}, \dots, m_0)$  in base  $\ell$

**Output:**  $f_{m,Q}(P)$  representing a class in  $\mathbb{F}_p^*/\langle \ell \rangle$

1:  $R \leftarrow Q, f \leftarrow 1, j \leftarrow 0$

2: **for**  $i$  from 0 to  $k-1$  **do**

3:   **if**  $m_i \neq 0$  **then**

4:      $A_1[f] \leftarrow R, A_1[j] \leftarrow f, j \leftarrow j + 1$

5:   **end if**

6:    $f \leftarrow f^{\ell^j} \cdot \ell_{m_i, \ell}(P)$ ,  $R \leftarrow [2]R$

7: **end for**

8:  $R \leftarrow A_1[0], f \leftarrow A_1[0]$

9: **for**  $j = 1, \dots, k$  **do**

10:    $f \leftarrow f \cdot A_1[j] \cdot \ell_{m_j, \ell}(P)$ ,  $R \leftarrow R$

11: **end for**

12: **return**  $f$

and so

$l(Q) = \theta_{Q, P} - y_{P, P} \left( \frac{y_{Q, P} - 1}{x_{Q, P} - x_P} (x_Q - x_P) + 1 \right)$ .

Writing this as  $\theta_{Q, P} + y_{P, P}$ , we have  $\theta_{Q, P} = \alpha_{Q, P} + \beta_{Q, P} y_{P, P}$ , where

$\alpha_{Q, P} = ((x_P^2 + Ax_P + B)y_{Q, P} - (x_Q^2 + Ax_Q + B)y_{P, P})/(x_Q - x_{P, P})$ , and

$\beta_{Q, P} = (\alpha_{Q, P} + \beta_{Q, P} y_{P, P})/(x_Q - x_{P, P})$ .

Paralleling a single pm

However, the right-to-left algorithm can be parallelized more efficiently by taking advantage of the fact that the line  $\ell$  is

the same for all  $i$ . In general, from Lemma 2 we deduce that the line  $\ell$  is

$y - y_P \left( \frac{y_{Q, P} - 1}{x_{Q, P} - x_P} (x_Q - x_P) + 1 \right)$ .

$Q$  with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $\ell' : y = \lambda_2 x + v_2$  be the tangent at  $R$  with divisor  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of

$f$  is  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ . The divisor of  $f'$  is  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

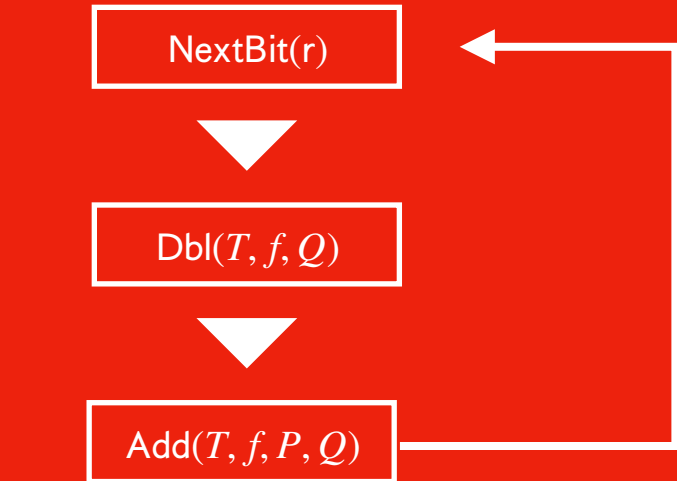
$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of

$f - f'$  is  $(f - f') = (P) + (Q) + (-P + Q) - 2(R) + (3(O))$ . The divisor of



## Speeding-up general pairings



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out

0

take some literature

1

implement all tricks  
that apply

2

benchmark speed  
and finetune

3

fast pairings

$Q$  with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $\ell' : y = \lambda_2 x + v_2$  be the tangent at  $R$  with divisor  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of



Figure 3.3.2: Two functions  $f$  and  $f'$

use in the conventional set-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $F_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

**Algorithm 3** Right-to-left version of Miller's algorithm with postponed additions

**Input:**  $Q \in G_p, P \in G_p, m = (m_{i-1}, \dots, m_1, m_0)$  class in  $\mathbb{F}_p^*/\langle G \rangle$

**Output:**  $f_{m,Q}(P)$  representing a class in  $\mathbb{F}_p^*/\langle G \rangle$

1:  $R \leftarrow Q, f \leftarrow 1, j \leftarrow 0$

2: **for**  $i$  from 0 to  $\ell - 1$  **do**

3:   **if**  $m_i = 1$  **then**

4:      $A_1[f] \leftarrow R, A_1[f] \leftarrow f, j \leftarrow j + 1$

5:   **end if**

6:    $f \leftarrow f^{2^i} \cdot \ell_{m_i, R}(P), R \leftarrow [2^i]R$

7: **end for**

8:  $R \leftarrow A_1[0], f \leftarrow A_1[0]$

9: **for**  $j = 1, j \leq \log_2(\ell - 1) + 1$  **do**

10:    $f \leftarrow f \cdot A_1[f] \cdot \ell_{m_j, R}(P), R \leftarrow R$

11: **end for**

12: **return**  $f$

and so

$l(Q) = \theta_{R,Q} - y_{R,Q} \left( \frac{y_{R,Q} - 1}{x_{R,Q} - x_P} (x_Q - x_P) + 1 \right).$

Writing this as  $\theta_{R,Q} + y_{R,Q}$ , we have  $\theta_{R,Q} = \alpha_{R,Q} + \beta_{R,Q} y_{R,Q}$ , where

$\alpha_{R,Q} = ((x_P^2 + Ax_P + By_P)(x_Q - x_P) + \beta_{R,Q} y_{R,Q}) / (x_Q - x_P),$

and

$\beta_{R,Q} = (\alpha_{R,Q} + \beta_{R,Q} y_{R,Q}) / (x_Q - x_P).$

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line  $l$  is

$y - y_P \left( \frac{y_{R,Q} - 1}{x_{R,Q} - x_P} (x - x_P) + 1 \right).$

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line  $l$  is

$y - y_P \left( \frac{y_{R,Q} - 1}{x_{R,Q} - x_P} (x - x_P) + 1 \right).$

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line  $l$  is

$y - y_P \left( \frac{y_{R,Q} - 1}{x_{R,Q} - x_P} (x - x_P) + 1 \right).$





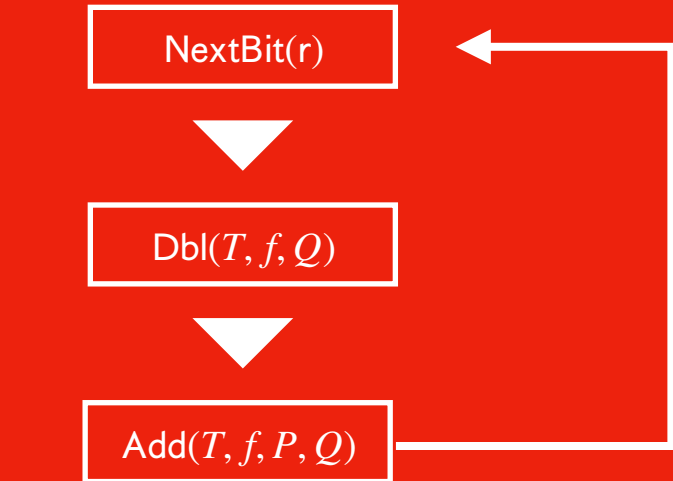
For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



Instead I describe the general approach,  
and leave all details out

## STEP 0

fast pairings



## Speeding-up general pairings



### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



### general approach

Instead I describe the general approach,  
and leave all details out

0

## take some literature

1

2

3

BACK TO



STEP 0

## fast pairings

Chapter 3. Divisors

Q) with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $f' : y = \lambda x + v_2$  be the tangent at R with divisor  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of

use in the conventional set-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $F_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed additions

Input:  $Q \in G, P \in G, m = (m_1, \dots, m_\ell)$  representing a class in  $\mathbb{F}_p^*/\mathbb{G}$

Output:  $f_{\text{pairing}}(P)$  representing a class in  $\mathbb{F}_p^*/\mathbb{G}$

1.  $K \leftarrow Q, f \leftarrow 1, j \leftarrow 0$

2. for  $i$  from 0 to  $\ell - 1$  do

3. if  $m_i = 1$  then

4.  $A_j[j] \leftarrow K, A_j[j] \leftarrow f, j \leftarrow j + 1$

5. end if

6.  $f \leftarrow f^2 \cdot \text{Lap}_{P, A_j[j]}(P), K \leftarrow [2]K$

7. end for

8.  $K \leftarrow A_j[j], f \leftarrow A_j[j]$

9. for  $i = 1, j \leq \ell(m) - 1, j \leftarrow j + 1$  do

10. return  $f$

Chapter 3. Divisors

Q) with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $f' : y = \lambda x + v_2$  be the tangent at R with divisor  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of

use in the conventional set-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $F_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed additions

Input:  $Q \in G, P \in G, m = (m_1, \dots, m_\ell)$  representing a class in  $\mathbb{F}_p^*/\mathbb{G}$

Output:  $f_{\text{pairing}}(P)$  representing a class in  $\mathbb{F}_p^*/\mathbb{G}$

1.  $K \leftarrow Q, f \leftarrow 1, j \leftarrow 0$

2. for  $i$  from 0 to  $\ell - 1$  do

3. if  $m_i = 1$  then

4.  $A_j[j] \leftarrow K, A_j[j] \leftarrow f, j \leftarrow j + 1$

5. end if

6.  $f \leftarrow f^2 \cdot \text{Lap}_{P, A_j[j]}(P), K \leftarrow [2]K$

7. end for

8.  $K \leftarrow A_j[j], f \leftarrow A_j[j]$

9. for  $i = 1, j \leq \ell(m) - 1, j \leftarrow j + 1$  do

10. return  $f$

Chapter 3. Divisors

Q) with divisor  $(f) = (P) + (Q) + (-P + Q) - 3(O)$ , and let  $f' : y = \lambda x + v_2$  be the tangent at R with divisor  $(f') = 2(R) + (-2(R) - 3(O))$ . The divisor of

use in the conventional set-to-right algorithm as it is given in Algorithm 2 on page 7. In the right-to-left version, each addition step in line 10 needs a general  $F_p$ -multiplication and a multiplication with a line function value. The conventional algorithm only needs a multiplication with a line. These huge costs cannot be compensated for by using affine coordinates with the inversion-sharing trick.

Algorithm 3 Right-to-left version of Miller's algorithm with postponed additions

Input:  $Q \in G, P \in G, m = (m_1, \dots, m_\ell)$  representing a class in  $\mathbb{F}_p^*/\mathbb{G}$

Output:  $f_{\text{pairing}}(P)$  representing a class in  $\mathbb{F}_p^*/\mathbb{G}$

1.  $K \leftarrow Q, f \leftarrow 1, j \leftarrow 0$

2. for  $i$  from 0 to  $\ell - 1$  do

3. if  $m_i = 1$  then

4.  $A_j[j] \leftarrow K, A_j[j] \leftarrow f, j \leftarrow j + 1$

5. end if

6.  $f \leftarrow f^2 \cdot \text{Lap}_{P, A_j[j]}(P), K \leftarrow [2]K$

7. end for

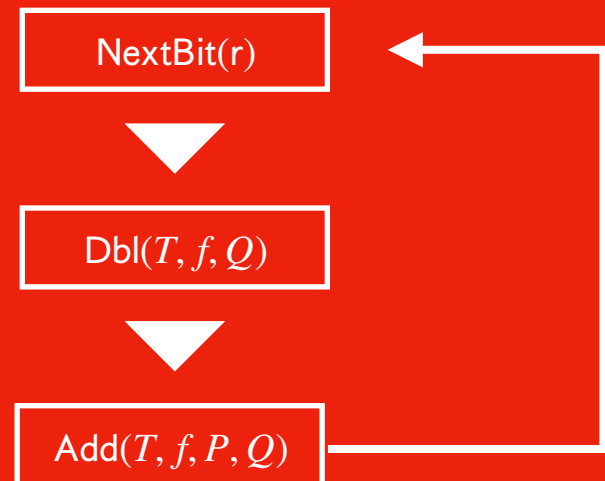
8.  $K \leftarrow A_j[j], f \leftarrow A_j[j]$

9. for  $i = 1, j \leq \ell(m) - 1, j \leftarrow j + 1$  do

10. return  $f$



For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



## Speeding-up general pairings

## general notice


Computing pairings fast is quite technical.  
Better suited for papers than slides

## general approach

Instead I describe the general approach,  
and leave all details out

take some literature

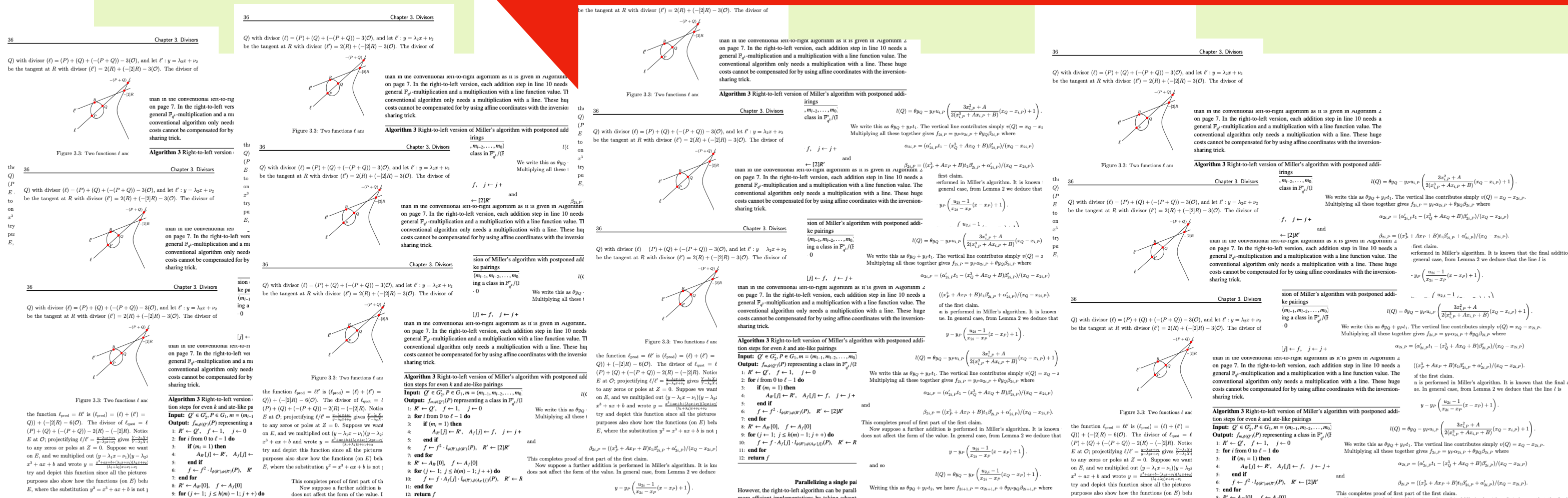
**BACK TO**



## STEP 0

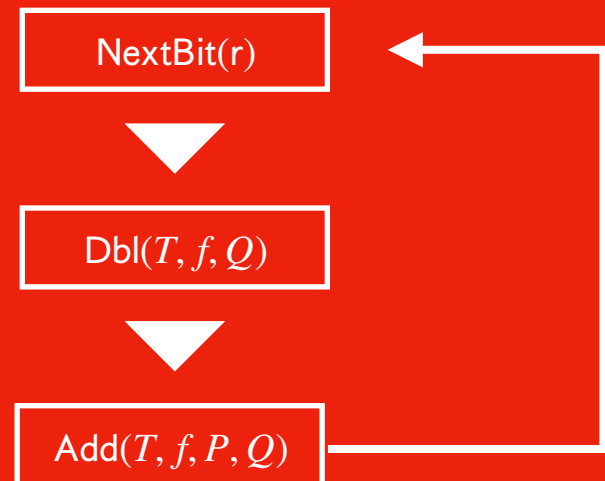
3

fast pairings





For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



## Speeding-up general pairings

## general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides

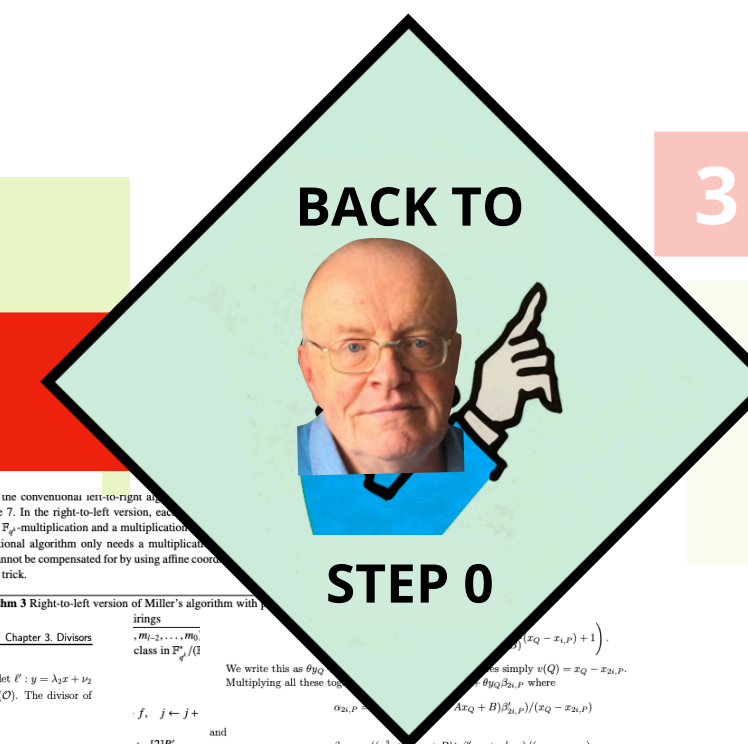


## general approach

Instead I describe the general approach,  
and leave all details out

take some literature

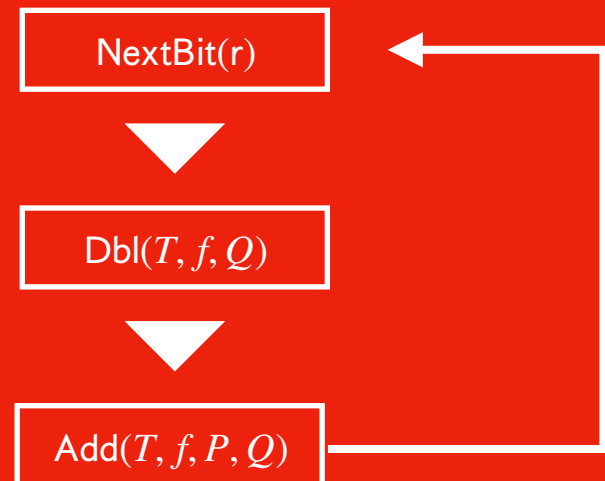
fast pairings

[illegible][illegible][illegible][illegible]

at the final addition  
the line  $i$  is



For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!



## Speeding-up general pairings

## general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides



Instead I describe the general approach,  
and leave all details out

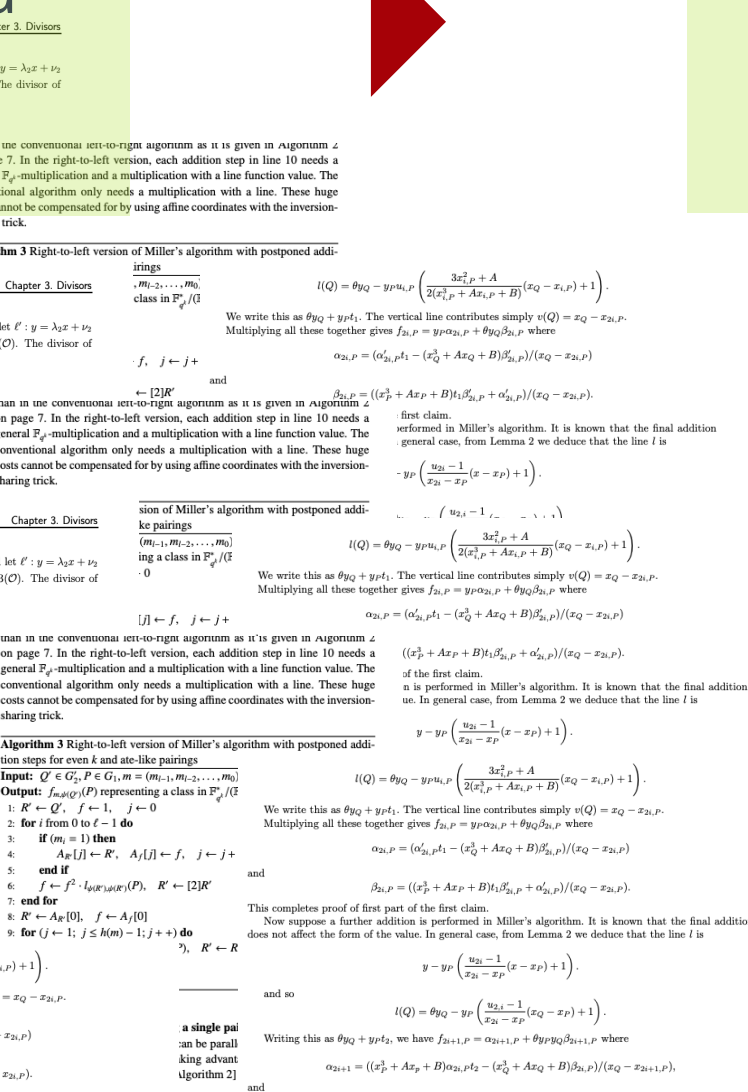
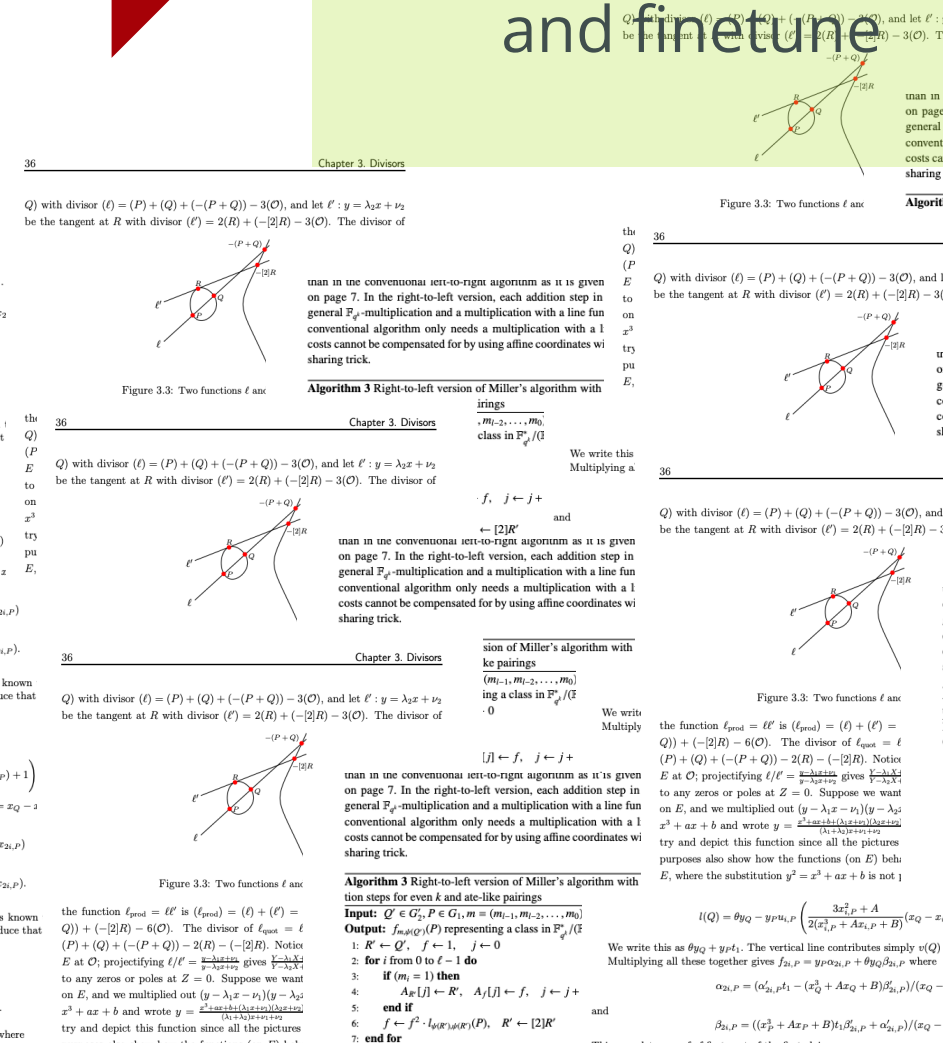
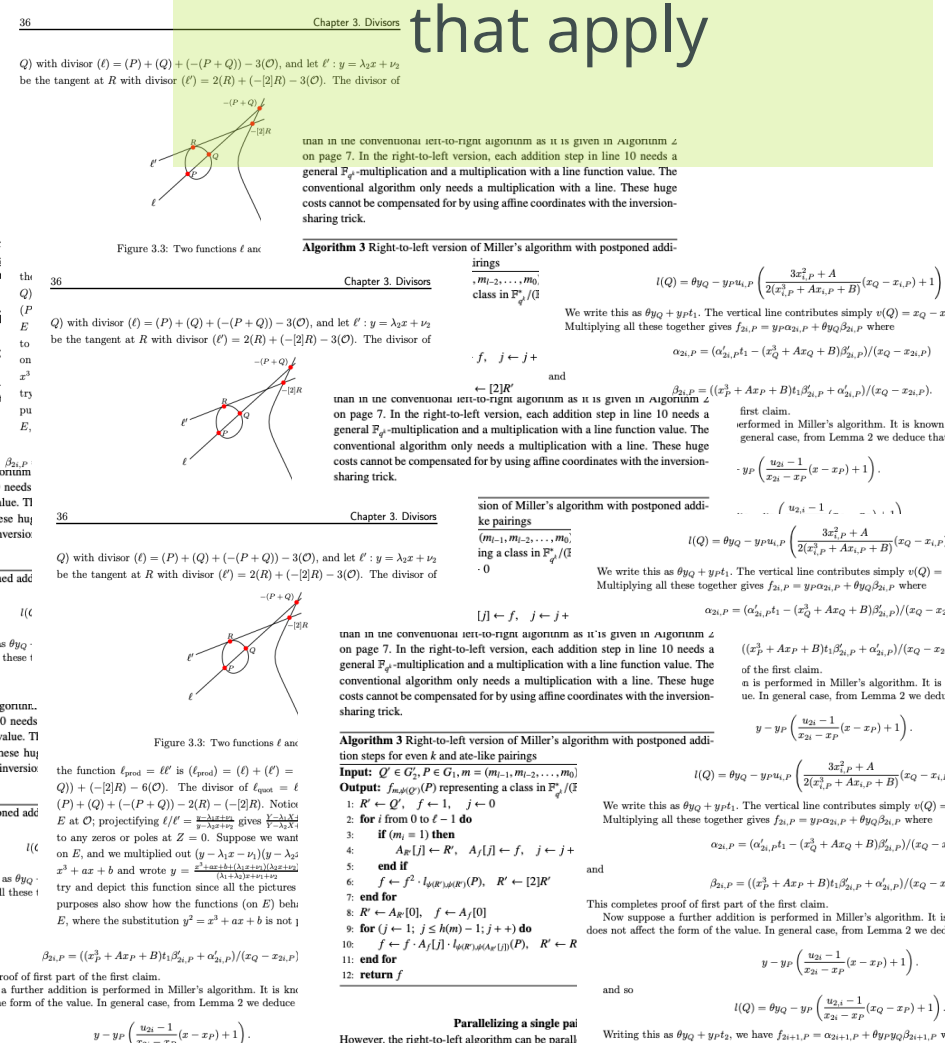
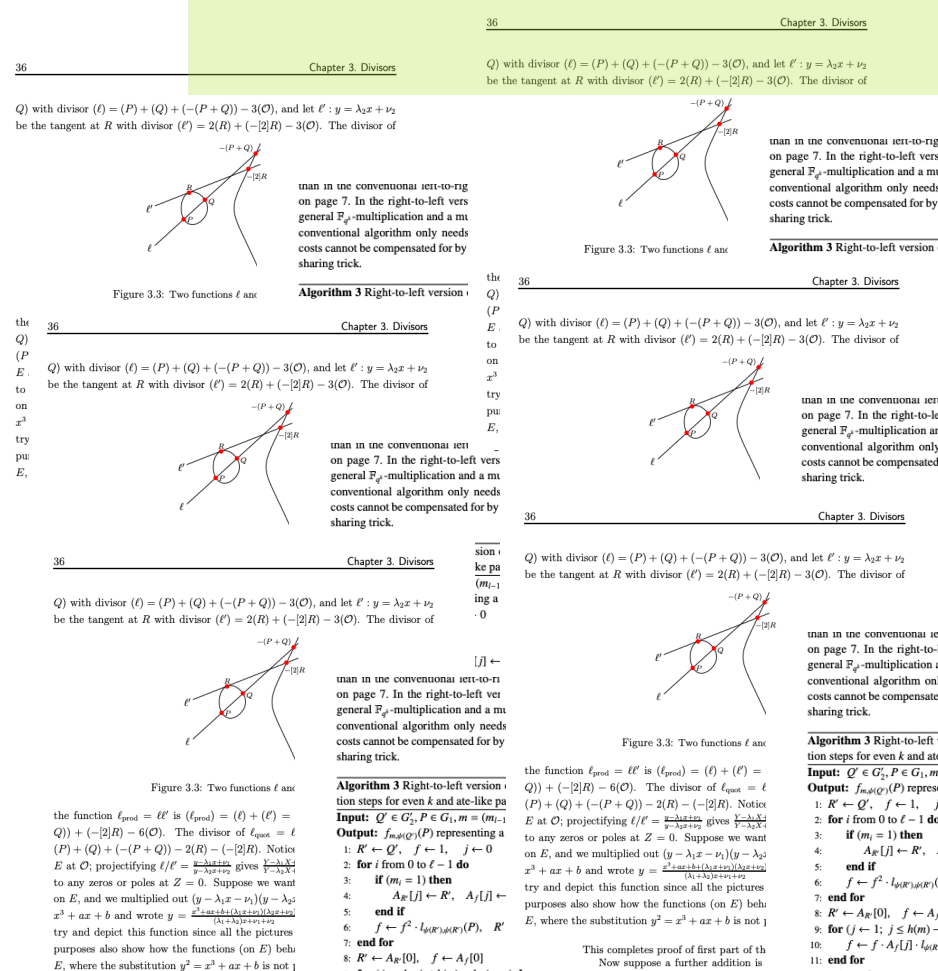
## 3

take some literature

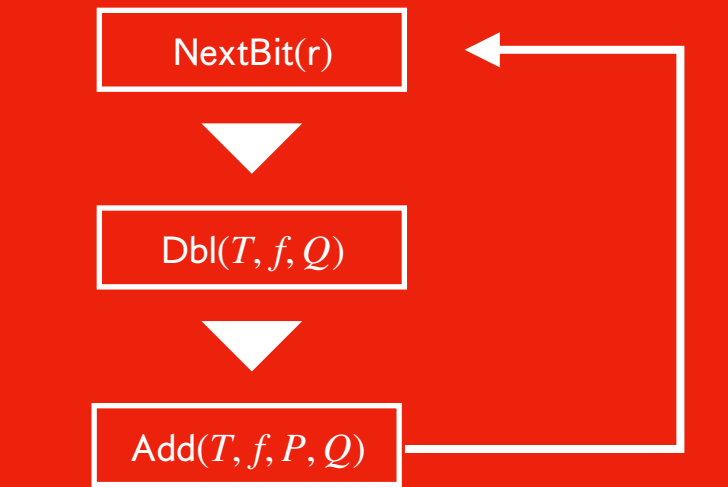
# implement all tricks that apply

# benchmark speed and finiteness

fast pairings



2



**Speeding-up  
general pairings**

!

### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides

✓

### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!

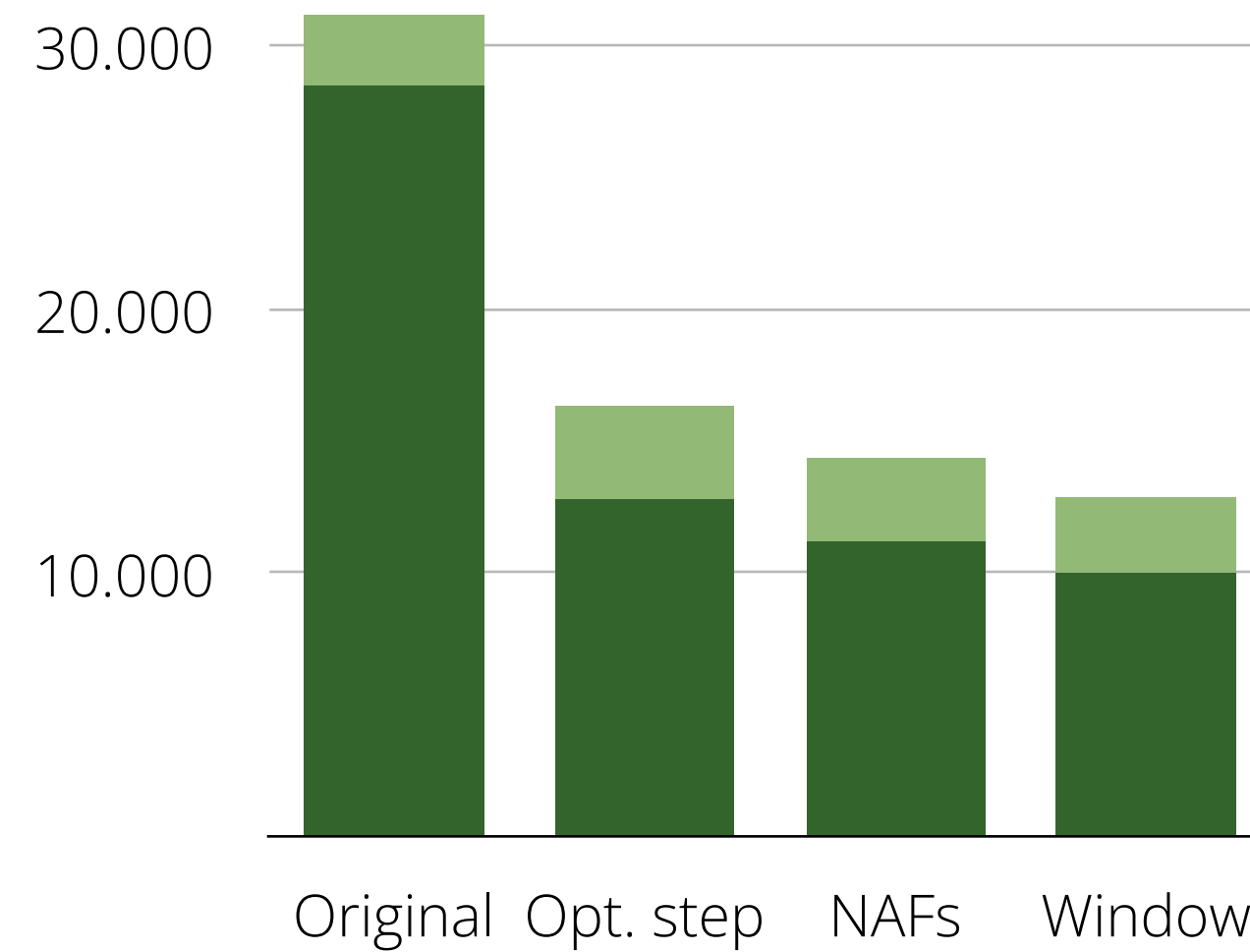
✓

### general approach

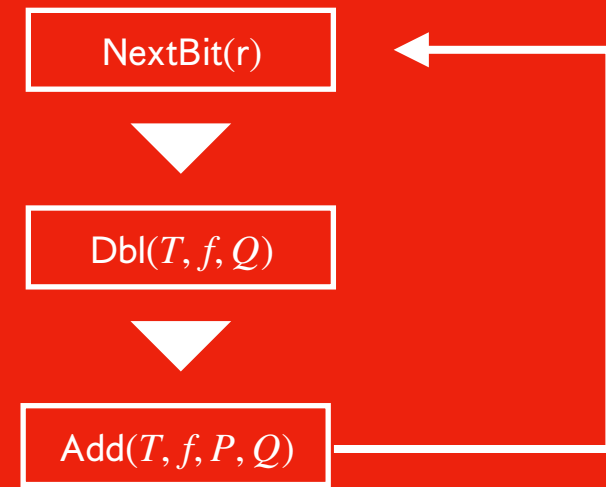
Instead I describe the general approach,  
and leave all details out

3

fast pairings



2



**Speeding-up  
general pairings**

!

### general notice

Computing pairings fast is quite technical.  
Better suited for papers than slides

✓

### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!

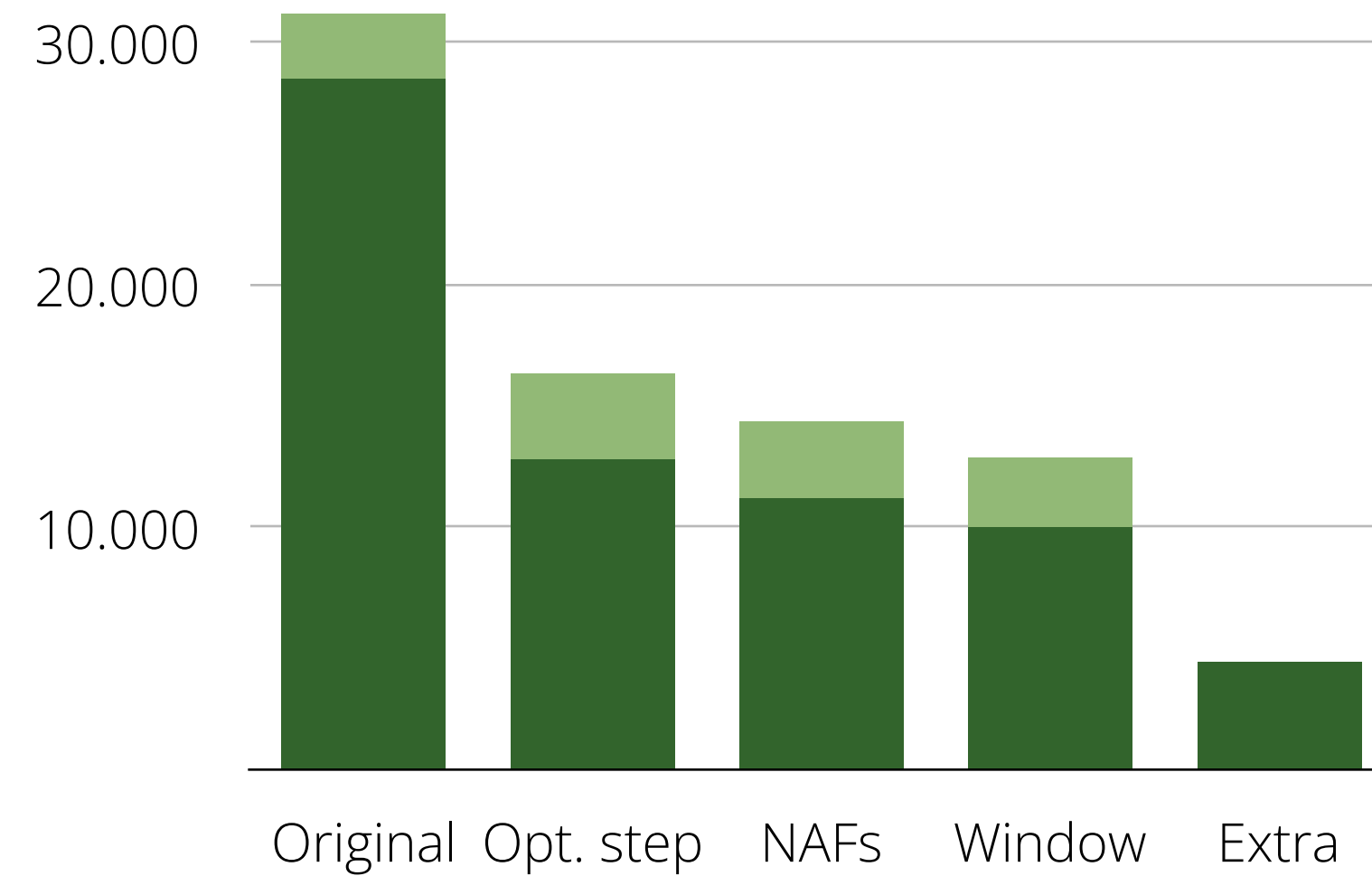
✓

### general approach

Instead I describe the general approach,  
and leave all details out

3

fast pairings

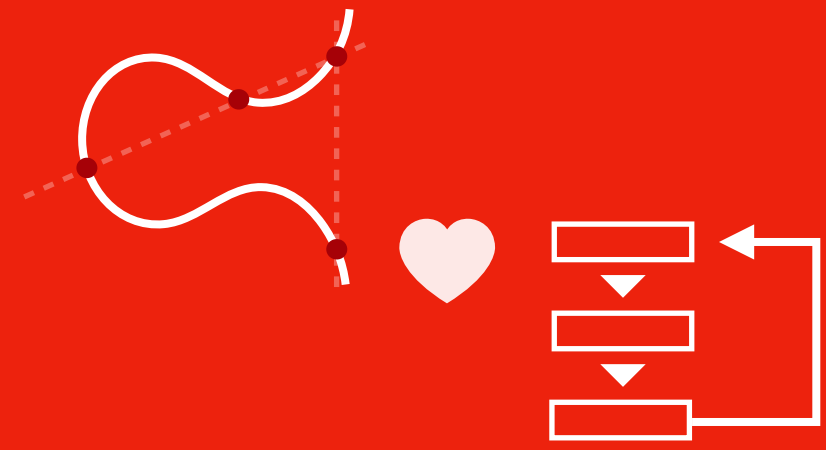


### extra pairings

if you have already computed  
 $e(P, Q_1)$ ,

it is very efficient to compute  
 $e(P, Q_2)$

# Fast pairings ♥ Isogeny crypto



**Applying pairings  
in isogeny crypto**



#### fast pairings

Optimized pairing  
computation for the specific  
scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$

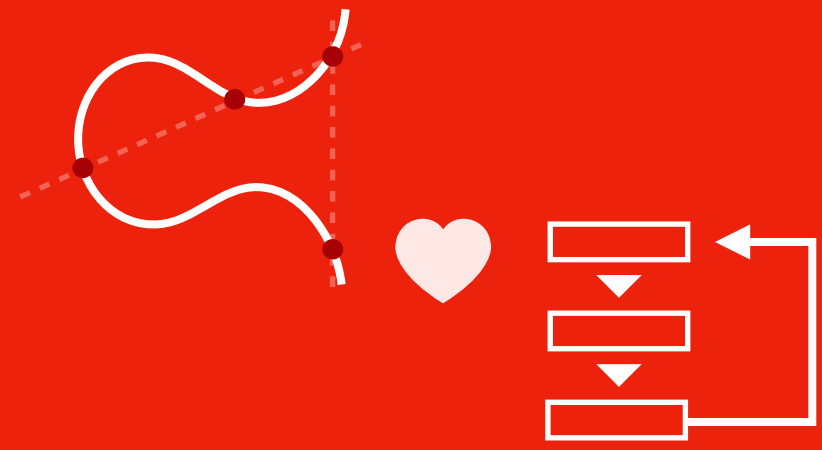
&



#### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ ,  
don't use curve arithmetic  
but pairing  $e(P, Q)$  to get  
overlap in orders!

**Faster isogeny subroutines**



## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$

&



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points





## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$

&



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .



## Applying pairings in isogeny crypto

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .

speedup: -75%



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$

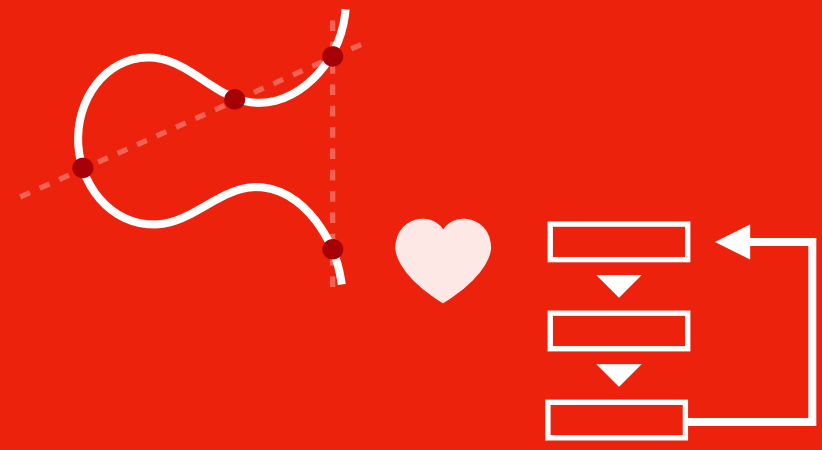
&



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines



## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$

&



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .

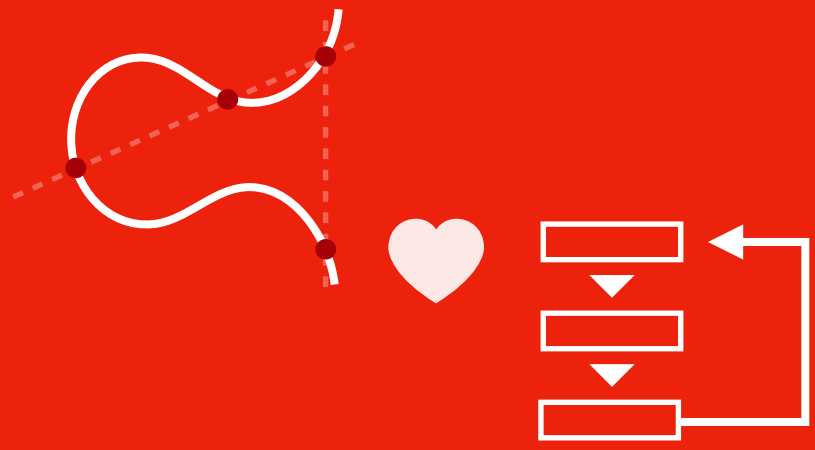
speedup: -75%



### compute full torsion $P$

In some CSIDH variants, we get  $E$

**Q:** find  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$  of order  $p + 1$ , e.g. full torsion points



## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .

speedup: -75%

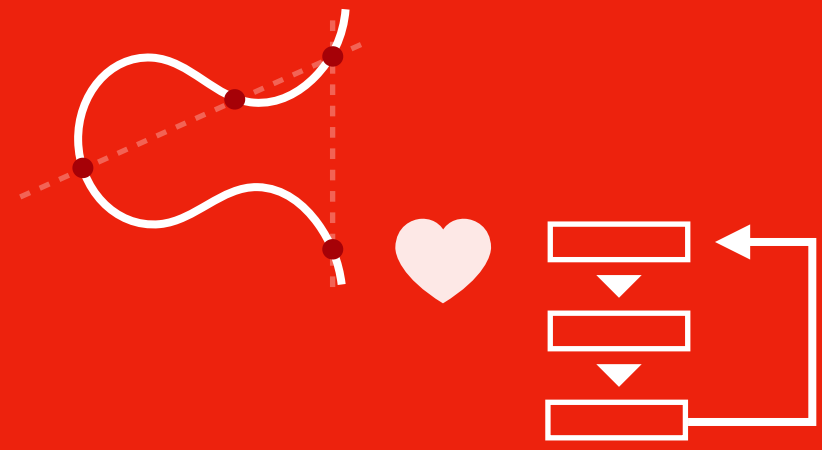


### compute full torsion $P$

In some CSIDH variants, we get  $E$

**Q:** find  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$  of order  $p + 1$ , e.g. full torsion points

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Compute order  $\zeta$  and apply Gauss' algorithm.



## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .

speedup: -75%



### compute full torsion $P$

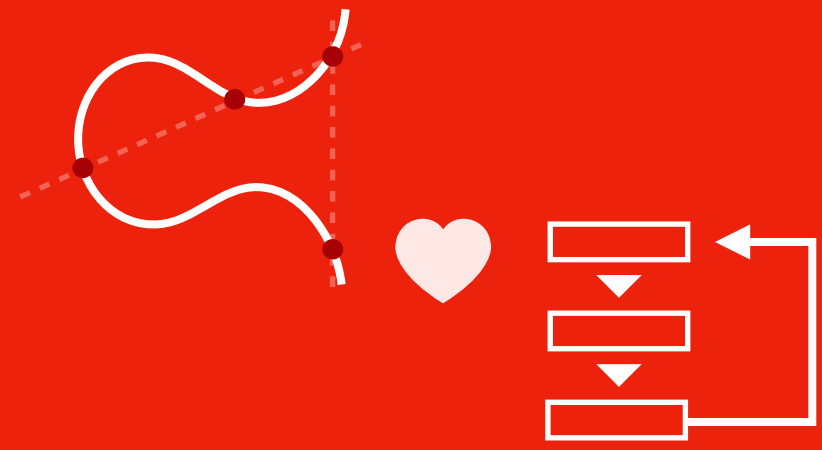
In some CSIDH variants, we get  $E$

**Q:** find  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$  of order  $p + 1$ , e.g. full torsion points

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Compute order  $\zeta$  and apply Gauss' algorithm.

speedup: case dependent, up to -75%





## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .



speedup: -75%

### compute full torsion $P$

In some CSIDH variants, we get  $E$

**Q:** find  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$  of order  $p + 1$ , e.g. full torsion points

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Compute order  $\zeta$  and apply Gauss' algorithm.

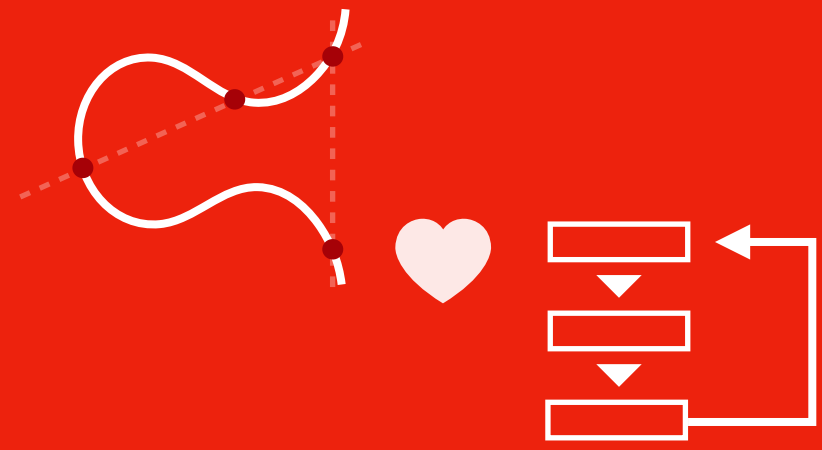


speedup: case dependent, up to -75%

### verify supersingularity

In some CSIDH variants, we get  $E$

**Q:** is  $E$  even supersingular? verify that it is!



## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .



speedup: -75%

### compute full torsion $P$

In some CSIDH variants, we get  $E$

**Q:** find  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$  of order  $p + 1$ , e.g. full torsion points

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Compute order  $\zeta$  and apply Gauss' algorithm.



speedup: case dependent, up to -75%

### verify supersingularity

In some CSIDH variants, we get  $E$

**Q:** is  $E$  even supersingular? verify that it is!

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Verify order  $\zeta \geq 4\sqrt{p}$ .





## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .



speedup: -75%

### compute full torsion $P$

In some CSIDH variants, we get  $E$

**Q:** find  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$  of order  $p + 1$ , e.g. full torsion points

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Compute order  $\zeta$  and apply Gauss' algorithm.



speedup: case dependent, up to -75%

### verify supersingularity

In some CSIDH variants, we get  $E$

**Q:** is  $E$  even supersingular? verify that it is!

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Verify order  $\zeta \geq 4\sqrt{p}$ .



speedup: -27% compared to CSIDH's





## Applying pairings in isogeny crypto



### fast pairings

Optimized pairing computation for the specific scenario  $P \in E(\mathbb{F}_p)$ ,  $Q \in E'(\mathbb{F}_p)$



### core idea

For  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ , don't use curve arithmetic but pairing  $e(P, Q)$  to get overlap in orders!

## Faster isogeny subroutines

### verify full torsion $P$

In some CSIDH variants, we are given  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$ .

**Q:** verify that both  $P$  and  $Q$  have order  $p + 1$ , e.g. full torsion points

**A:** compute  $\zeta = e(P, Q)$  and check that order  $\zeta$  is  $p + 1$ .



speedup: -75%

### compute full torsion $P$

In some CSIDH variants, we get  $E$

**Q:** find  $P \in E(\mathbb{F}_p)$  and  $Q \in E'(\mathbb{F}_p)$  of order  $p + 1$ , e.g. full torsion points

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Compute order  $\zeta$  and apply Gauss' algorithm.



speedup: case dependent, up to -75%

### verify supersingularity

In some CSIDH variants, we get  $E$

**Q:** is  $E$  even supersingular? verify that it is!

**A:** take random,  $P, Q$ , then find  $\zeta = e(P, Q)$ . Verify order  $\zeta \geq 4\sqrt{p}$ .



speedup: +2% compared to Doliskani

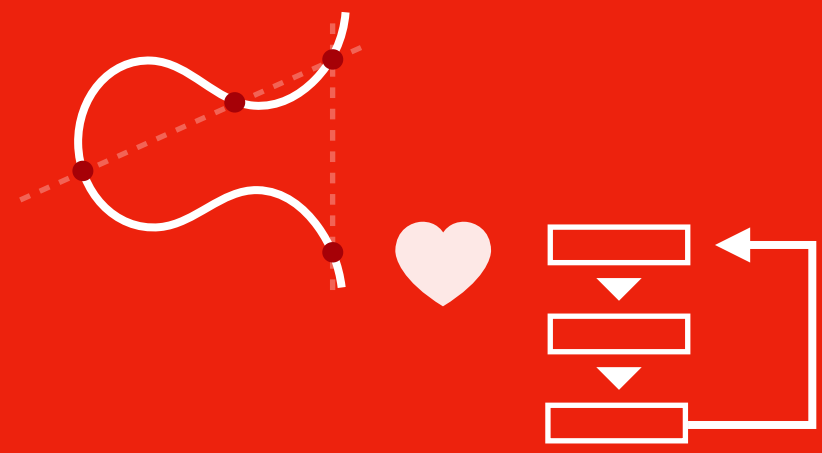


## Applying pairings in isogeny crypto

### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free



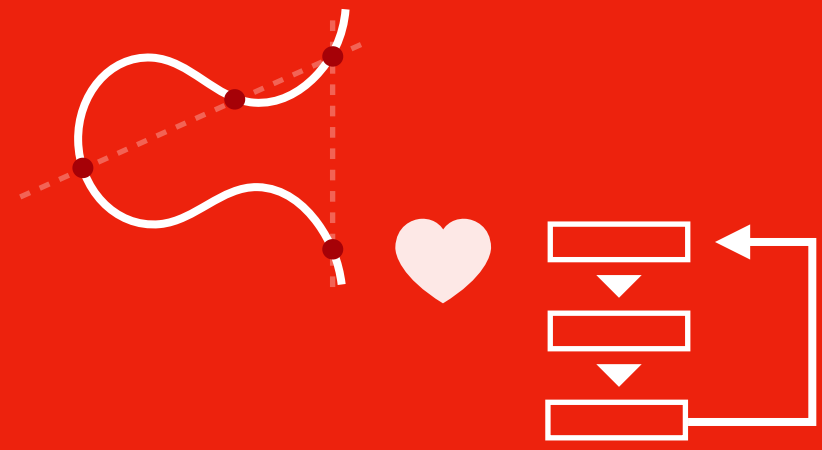
## Applying pairings in isogeny crypto

### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?



## Applying pairings in isogeny crypto

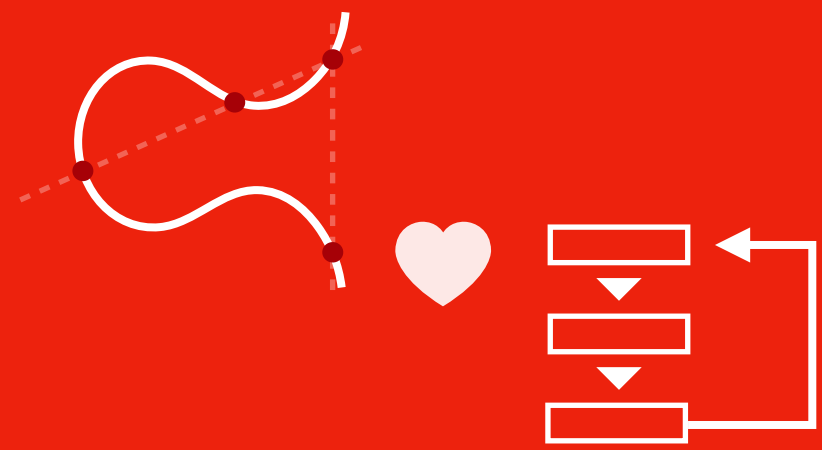
### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?

- ✓ classical security well understood



## Applying pairings in isogeny crypto

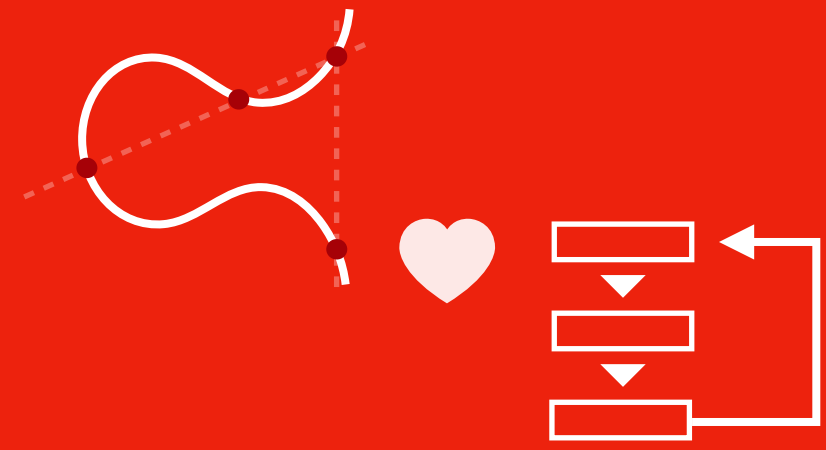
### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?

- ✓ classical security well understood
- ? quantum security well understood



## Applying pairings in isogeny crypto

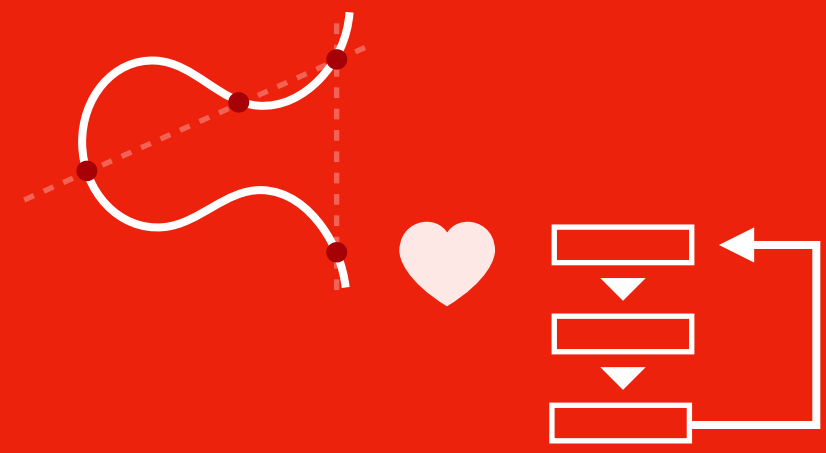
### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?

- ✓ classical security well understood
- ? quantum security well understood
- ? quite slow constant-time



## Applying pairings in isogeny crypto

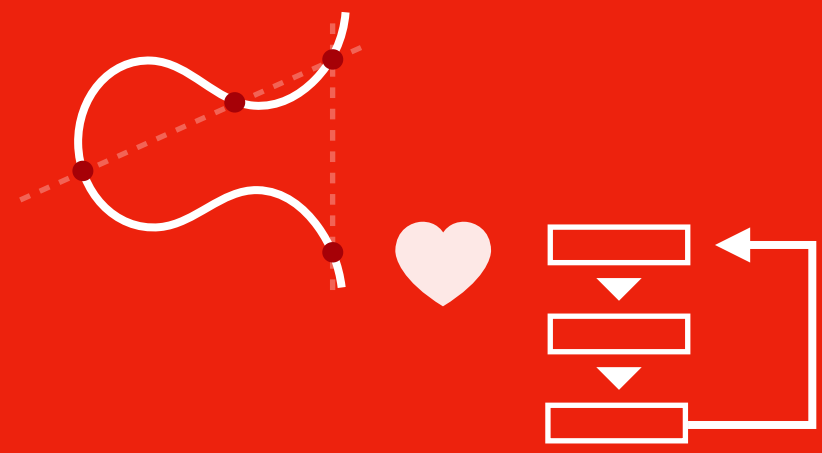
### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?

- ✓ classical security well understood
- ? quantum security well understood
- ? quite slow constant-time
- ✗ *very slow* deterministic, dummy-free



## Applying pairings in isogeny crypto

### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

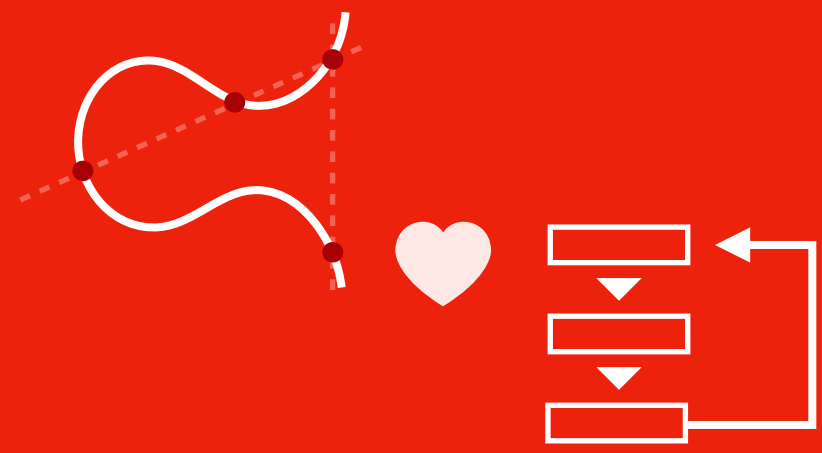
#### CSIDH's maturity?

- ✓ classical security well understood
- ? quantum security well understood
- ? quite slow constant-time
- ✗ *very slow* deterministic, dummy-free

how do we achieve fast high-security CSIDH?  
*constant-time, deterministic, dummy-free*







## Applying pairings in isogeny crypto

### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?

- ✓ classical security well understood
- ? quantum security well understood
- ? quite slow constant-time
- ✗ *very slow* deterministic, dummy-free

how do we achieve fast high-security CSIDH?  
*constant-time, deterministic, dummy-free*



**previously**

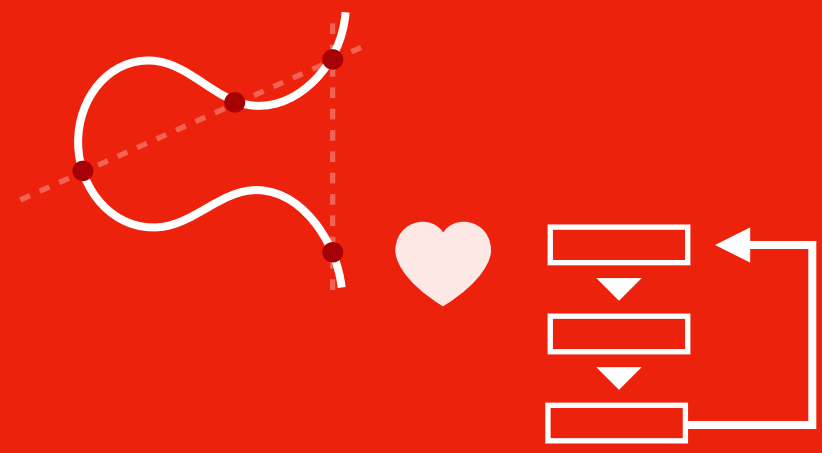


**with pairings**



**to do**

- add **seed** for torsion points in key
- **slow** verification of torsion points
- **slow** group action due to dummy-free



## Applying pairings in isogeny crypto

### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?

- ✓ classical security well understood
- ? quantum security well understood
- ? quite slow constant-time
- ✗ *very slow* deterministic, dummy-free

how do we achieve fast high-security CSIDH?  
*constant-time, deterministic, dummy-free*



**previously**

- add **seed** for torsion points in key
- **slow** verification of torsion points
- **slow** group action due to dummy-free

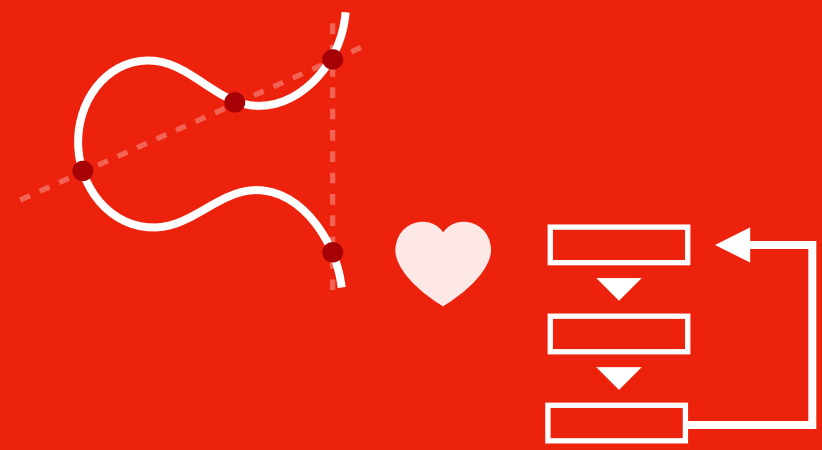


**with pairings**

- **fast** verification of torsion points
- removes probability from CTIDH
- improved group action and ss verify!



**to do**



## Applying pairings in isogeny crypto

### why pairings at all?

#### scheme maturity

- classical security well understood
- quantum security well understood
- fast, constant-time implementation
- deterministic and dummy-free

#### CSIDH's maturity?

- ✓ classical security well understood
- ? quantum security well understood
- ? quite slow constant-time
- ✗ *very slow* deterministic, dummy-free

how do we achieve fast high-security CSIDH?  
*constant-time, deterministic, dummy-free*



**previously**

- add **seed** for torsion points in key
- **slow** verification of torsion points
- **slow** group action due to dummy-free



**with pairings**

- **fast** verification of torsion points
- removes probability from CTIDH
- improved group action and ss verify!



**to do**

- analyse **optimal** use of torsion
- can we use **faster** torsion finding?
- can improve group action!

**Thank you!**

**Any questions\*?**

\*If not, I have a question for you...



**Constant-time  
Gauss' algorithm?**

## *Finite field world*

**Q:** Given  $\mathbb{F}_q$  find generator  $\zeta$  for  $\mathbb{F}_q^*$

## *Curve world*

Given curve  $E$  over  $\mathbb{F}_p$ ,  
find full torsion point  $P$



Constant-time  
Gauss' algorithm?

## Finite field world

**Q:** Given  $\mathbb{F}_q$  find generator  $\zeta$  for  $\mathbb{F}_q^*$

**A:**

### GAUSS' ALGORITHM

1. Take random  $\zeta \in \mathbb{F}_q$ , compute  $t = \text{Order}(\zeta)$
2. If  $t = q - 1$ , **stop**,
3. **else** take random  $\beta \in \mathbb{F}_q^*$  and compute  $s = \text{Order}(\beta)$ 
  - a. if  $s = q - 1$ , **stop**
  - b. **else** find coprime  $d \mid t$  and  $e \mid s$  with  $d \cdot e = \text{lcm}(t, s)$
  - c. set  $\zeta \leftarrow \zeta^{t/d} \cdot \beta^{s/e}$  and  $t \leftarrow d \cdot e$  and **repeat** from 2.

## Curve world

Given curve  $E$  over  $\mathbb{F}_p$ ,  
find full torsion point  $P$



Take  $P$  and  $Q$ ,  
Compute their torsion.  
If  $P$  not full torsion,  
take right multiple  $Q$   
set  $P \leftarrow P + Q$  to fill  
missing torsion in  $P$   
repeat until full torsion



Constant-time  
Gauss' algorithm?

## Finite field world

**Q:** Given  $\mathbb{F}_q$  find generator  $\zeta$  for  $\mathbb{F}_q^*$

**A:**

### GAUSS' ALGORITHM

1. Take random  $\zeta \in \mathbb{F}_q$ , compute  $t = \text{Order}(\zeta)$
2. If  $t = q - 1$ , **stop**,
3. **else** take random  $\beta \in \mathbb{F}_q^*$  and compute  $s = \text{Order}(\beta)$ 
  - a. if  $s = q - 1$ , **stop**
  - b. **else** find coprime  $d \mid t$  and  $e \mid s$  with  $d \cdot e = \text{lcm}(t, s)$
  - c. set  $\zeta \leftarrow \zeta^{t/d} \cdot \beta^{s/e}$  and  $t \leftarrow d \cdot e$  and **repeat** from 2.

**Q:** Given  $\mathbb{F}_q$  find generator  $\zeta$  for  $\mathbb{F}_q^*$  *in constant-time*

## Curve world

Given curve  $E$  over  $\mathbb{F}_p$ ,  
find full torsion point  $P$



Take  $P$  and  $Q$ ,  
Compute their torsion.  
If  $P$  not full torsion,  
take right multiple  $Q$   
set  $P \leftarrow P + Q$  to fill  
missing torsion in  $P$   
repeat until full torsion



Given curve  $E$  over  $\mathbb{F}_p$ ,  
find full torsion point  $P$   
*in constant-time*