Elliptic curves in Nemo

Jean Kieffer

École normale supérieure de Paris & INRIA

07/01/17

Key exchange from hard homogeneous spaces

Let G be an abelian group acting on a set X with some given point x_0. If the action is

Key exchange from hard homogeneous spaces

Let G be an abelian group acting on a set X with some given point x_0. If the action is

- easy to compute (polynomial time),

Key exchange from hard homogeneous spaces

Let G be an abelian group acting on a set X with some given point x_0. If the action is

- easy to compute (polynomial time),
- hard to invert (exponential time?),

Key exchange from hard homogeneous spaces

Let G be an abelian group acting on a set X with some given point x_0. If the action is

- easy to compute (polynomial time),
- hard to invert (exponential time?),

then there is an analogue of the Diffie–Hellman key exchange [2].

$$\begin{tikzcd}[column sep=small]
 & x_0 \arrow[dl, dash, "a"'] \arrow[dr, dash, "b"] & & \\
a \star x_0 \arrow[dr, dash, "b"'] & & b \star x_0 \arrow[dl, dash, "a"] & \text{public} \\
 & (ab) \star x_0 & & \text{shared secret}
\end{tikzcd}$$

The Couveignes–Rostovtsev–Stolbunov scheme

Question

Where can we find such an action?

The Couveignes–Rostovtsev–Stolbunov scheme

Question

Where can we find such an action?

Answer [2], [3]

Use the action of a class group on a set of isogenous elliptic curves.

The Couveignes–Rostovtsev–Stolbunov scheme

Question

Where can we find such an action?

Answer [2], [3]

Use the action of a class group on a set of isogenous elliptic curves.

Goals

- Explain what this means
- Describe the computations needed
- Discuss our EllipticCurves module for Nemo.

1. Context
2. An example in isogeny-based cryptography
   - Basics
   - Computations
3. The EllipticCurves module for Nemo
   - Contents
   - Further development
   - Some benchmarks
4. Conclusion

Elliptic curves over k

- *Elliptic curves* over a field k are algebraic curves that have an abelian group structure, e.g.

$$\begin{aligned} E_1 &: y^2 + a_1xy = x^3 + a_2x^2 + a_4x + a_6 \\ E_2 &: y^2 = x^3 + ax + b \\ E_3 &: By^2 = x^3 + Ax^2 + x. \end{aligned}$$

Weierstrass, Short Weierstrass and Montgomery forms, respectively.

Elliptic curves over k

- *Elliptic curves* over a field k are algebraic curves that have an abelian group structure, e.g.

$$\begin{aligned} E_1 &: y^2 + a_1xy = x^3 + a_2x^2 + a_4x + a_6 \\ E_2 &: y^2 = x^3 + ax + b \\ E_3 &: By^2 = x^3 + Ax^2 + x. \end{aligned}$$

  Weierstrass, Short Weierstrass and Montgomery forms, respectively.

- *Isogenies* are nonzero morphisms. Our isogenies will be defined over k. As rational maps, they have *degrees*.

Isogenies are subgroups

From now on, $k = \mathbb{F}_p$ is a prime finite field.

Isogenies are subgroups

From now on, $k = \mathbb{F}_p$ is a prime finite field.
Let E/k be an elliptic curve, and $\ell \neq p$ be an odd prime.
Giving the following is equivalent :

Isogenies are subgroups

From now on, $k = \mathbb{F}_p$ is a prime finite field.
Let E/k be an elliptic curve, and $\ell \neq p$ be an odd prime.
Giving the following is equivalent :

- An isogeny $E \to E'$ of degree ℓ

Isogenies are subgroups

From now on, $k = \mathbb{F}_p$ is a prime finite field.
Let E/k be an elliptic curve, and $\ell \neq p$ be an odd prime.
Giving the following is equivalent :

- An isogeny $E \to E'$ of degree ℓ
- Its kernel, which is a cyclic subgroup of E of order ℓ

Isogenies are subgroups

From now on, $k = \mathbb{F}_p$ is a prime finite field.
Let E/k be an elliptic curve, and $\ell \neq p$ be an odd prime.
Giving the following is equivalent :

- An isogeny $E \to E'$ of degree ℓ
- Its kernel, which is a cyclic subgroup of E of order ℓ
- A polynomial of degree $\frac{\ell-1}{2}$ in x defining the kernel.

Isogenies are subgroups

From now on, $k = \mathbb{F}_p$ is a prime finite field.
Let E/k be an elliptic curve, and $\ell \neq p$ be an odd prime.
Giving the following is equivalent :

- An isogeny $E \to E'$ of degree ℓ
- Its kernel, which is a cyclic subgroup of E of order ℓ
- A polynomial of degree $\frac{\ell-1}{2}$ in x defining the kernel.

If we know this *kernel polynomial*, we can easily find E' using Vélu's formulas.

Action of the class group

Proposition

Let $E/\mathbb{F}_p$ be an ordinary elliptic curve.

- The ring $\mathrm{End}(E)$ is isomorphic to a quadratic order $\mathcal{O}$.
- For each prime number ℓ, there are either 2 (split case), 1 (ramified case) or 0 (inert case) ideals in $\mathcal{O}$ of norm ℓ.
  From now on, ℓ will always be prime, odd and split.
- Ideal of norm ℓ = tuple (ℓ, v), $v \in \mathbb{Z}/\ell\mathbb{Z}$.
- There is an action on the set of elliptic curves with CM by $\mathcal{O}$. Ideals of norm ℓ act as ℓ-isogenies.
- This action is simply transitive.

Action of the class group

Proposition

Let $E/\mathbb{F}_p$ be an ordinary elliptic curve.

- The ring $\mathrm{End}(E)$ is isomorphic to a quadratic order $\mathcal{O}$.
- For each prime number ℓ, there are either 2 (split case), 1 (ramified case) or 0 (inert case) ideals in $\mathcal{O}$ of norm ℓ. From now on, ℓ will always be prime, odd and split.
- Ideal of norm ℓ = tuple (ℓ, v), $v \in \mathbb{Z}/\ell\mathbb{Z}$.
- There is an action on the set of elliptic curves with CM by $\mathcal{O}$. Ideals of norm ℓ act as ℓ-isogenies.
- This action is simply transitive.

Question

How can we compute this action ?

Main algorithm

Problem

Given $E/\mathbb{F}_p$ and a prime $\ell \neq p$, how can we compute the curves linked to E by an ℓ-isogeny?

Main algorithm

Problem

Given $E/\mathbb{F}_p$ and a prime $\ell \neq p$, how can we compute the curves linked to E by an ℓ-isogeny?

Most general idea

Let $\Phi_\ell(X, Y)$ be the ℓ^{th} classical modular polynomial. The two roots j_1, j_2 of

$$\Phi_\ell(j(E), Y)$$

are the j-invariants of the neighbors of E. To choose the one corresponding to an ideal (ℓ, v):

- compute the kernel $K(x)$ of the isogeny $E \to j_1$
- check if the Frobenius acts on it as scalar mult. by v:

$$(x^p, y^p) \stackrel{?}{=} [v] \cdot (x, y) \mod K(x) \text{ and curve equation.}$$

Bostan–Morain–Salvy–Schost [1]

Question

How can we compute the kernel $K(x)$ of $\phi : E \to j_1$?

Bostan–Morain–Salvy–Schost [1]

Question

How can we compute the kernel $K(x)$ of $\phi \ : \ E \to j_1$?

Idea

If ϕ is *normalized*, the rational fraction defining it satisfies a simple differential equation.

Bostan–Morain–Salvy–Schost [1]

Question

How can we compute the kernel $K(x)$ of $\phi : E \to j_1$?

Idea

If ϕ is *normalized*, the rational fraction defining it satisfies a simple differential equation.

Algorithm

- Normalize ϕ (involves evaluating modular polynomials)
- Solve this ODE in power series up to a certain precision with a Newton iteration
- Recover $K(x)$ using the Berlekamp–Massey rational reconstruction algorithm.

Another solution

Problem

Given $E/\mathbb{F}_p$ and a prime $\ell \neq p$, how can we compute the curves linked to E by an ℓ-isogeny?

Finding roots of modular polynomials is costly : $\Phi_\ell(X, Y)$ has degree $\ell + 1$ in both variables.

Another solution

Problem

Given $E/\mathbb{F}_p$ and a prime $\ell \neq p$, how can we compute the curves linked to E by an ℓ-isogeny?

Finding roots of modular polynomials is costly : $\Phi_\ell(X, Y)$ has degree $\ell + 1$ in both variables.

More specific idea

Suppose that for some d, K is the only subgroup of order ℓ in E whose points are defined over $\mathbb{F}_{p^d}$.

- Look for ℓ-torsion points over this field to find K, using scalar multiplications
- Compute the curve E/K using Vélu's formulas.

The isogeny $E \to E/K$ has degree ℓ.

Finding adequate curves

This second method is only efficient with small-degree extensions.
Not every curve satisfies the conditions before for small d: we have to look for adequate curves.
In practice, we have to use both algorithms, general and specific.

What we would like Nemo to do

In the general method:

- Define elliptic curves over finite fields and general rings
- Define isogenies, sacalar multiplication and isomorphisms
- Find roots of polynomials over finite fields
- Solve ODEs in power series with Newton iterations
- Berlekamp–Massey

What we would like Nemo to do

In the general method:

- Define elliptic curves over finite fields and general rings
- Define isogenies, sacalar multiplication and isomorphisms
- Find roots of polynomials over finite fields
- Solve ODEs in power series with Newton iterations
- Berlekamp–Massey

In the specific method:

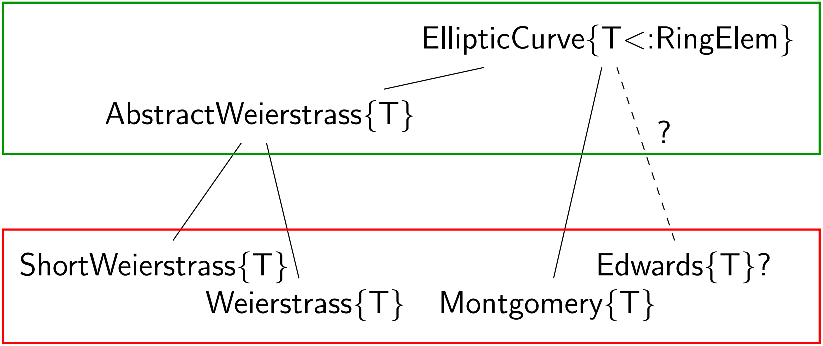
- Define points on elliptic curves
- Arithmetic operations on elliptic curves
- Extensions of finite fields.

Types for curves

We want both Weierstrass models (all curves have one) and Montgomery models (efficient arithmetic).

Types for curves

We want both Weierstrass models (all curves have one) and Montgomery models (efficient arithmetic).



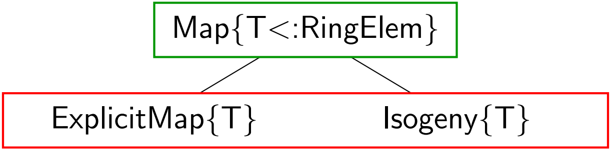
E.g. `j-invariant` is defined for the EllipticCurve type, while `a-invariants` is only defined for AbstractWeierstrass.

Types for maps

Maps can be isomorphisms between different models (evaluate on points), isogenies (compute image and kernels), scalar multiplications (both?).

Types for maps

Maps can be isomorphisms between different models (evaluate on points), isogenies (compute image and kernels), scalar multiplications (both?).




```
immutable ExplicitMap{T} <:  Map{T}
domain::EllipticCurve{T}
image::EllipticCurve{T}
map::Function
end
```

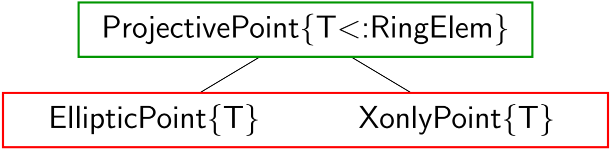
```
immutable Isogeny{T} <:  Map{T}
domain::EllipticCurve{T}
degree::Integer
kernel::PolyElem{T}
image::EllipticCurve{T}
end
```

Types for points

- Should points be attached with a curve?
- Arithmetic on Montgomery curves is much more efficient using only x-coordinates.

Types for points

- Should points be attached with a curve?
- Arithmetic on Montgomery curves is much more efficient using only x-coordinates.




```
type EllipticPoint{T} <:
ProjectivePoint{T}
X::T
Y::T
Z::T
curve::EllipticCurve{T}
end
```

```
type XonlyPoint{T} <:
ProjectivePoint{T}
X::T
Z::T
curve::MontgomeryCurve{T}
end
```

What EllipticCurves can do

- Define curves, points and maps, check for equality and validity
- Basic functions such as invariants
- Compute isomorphisms between different models
- Generic arithmetic on Weierstrass/Montgomery curves
- Efficient x-only arithmetic on Montgomery curves
- Division polynomials for ShortWeierstrass
- Isogeny computations : Vélu's formulas and the BMSS algorithm for short Weierstrass and Montgomery curves
- Modular polynomials (for small ℓ's)
- Over finite fields : random points, torsion points, computation of Frobenius eigenvalues.

Useful things that should be done elsewhere

In finite fields :

- Multiplicative orders
- Random elements
- Square roots
- Roots of polynomials and irreducible polynomials
- Field extensions over *prime* fields

Others:

- Derivatives of multivariate polynomials

Further possible development

- Call (system) PARI to compute the cardinality of curves over finite fields
- Compute modular polynomials/equations on the fly?
- Zeta functions?
- Have p-adic numbers to compute isogenies in small characteristic?
- Go down the arithmetic route for elliptic curves over number fields or local fields?

Three ways to compute roots over $\mathbb{F}_p$

At present, there is no direct way to do this in Nemo.

Sol. 1 (Nemo)

```
function roots(P)
A = parent(P)
X = gen(A)
R = ResidueRing(A, P)
Frob = R(X)^BigInt(p)
Frob = data(Frob)
g = gcd(Frob - X, P)
fact = factor(g)
...  # recover roots
end
```

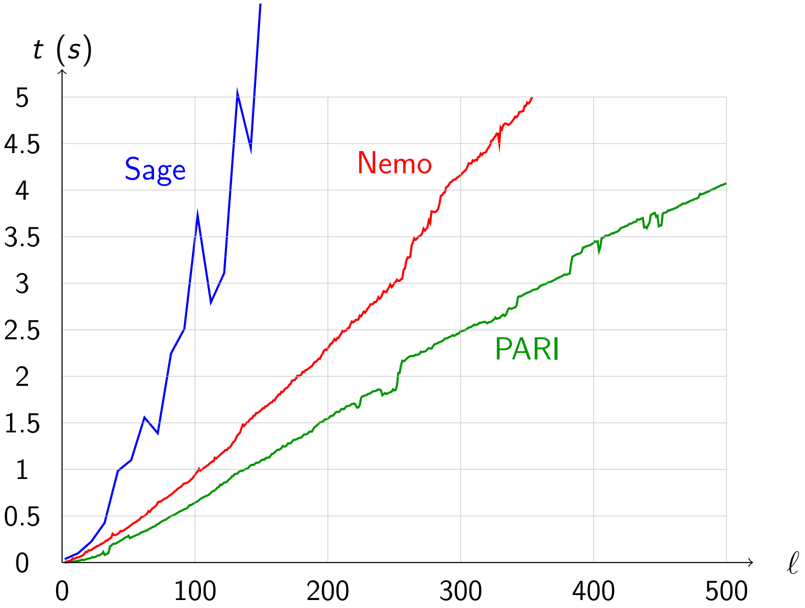
Sol. 2 (Sage/PARI)

```
def roots(P):
Q = pari(P)
rts = Q.polrootsmod(p)
return rts.sage()
```

Sol. 3 (Sage)

```
def roots(P):
A = P.parent()
X = A.gen()
R = A.quotient(P)
Frob = R(X)**p
Frob = Frob.lift()
g = gcd(Frob, P)
return g.roots()
```

Timing results



Three ways to compute scalar multiplications

Sol. 1 (Nemo)

```
E = Weierstrass(...)
Fext, _ = FiniteField(p^d, alpha)
Eext = base_extend(E, Fext)
P = random(Eext)
times(p^d, P)
```

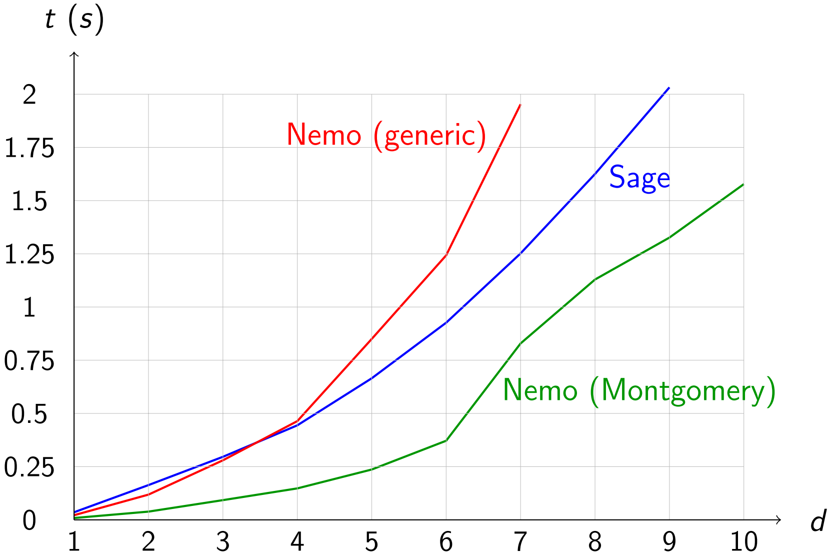
Sol. 2 (Nemo)

```
E = Montgomery(...)
Fext, _ = FiniteField(p, d, alpha)
Eext = base_extend(E, Fext)
P = randomXonly(Eext)
times(p^d, P)
```

Sol. 3 (Sage)

```
E = EllipticCurve(...)
Fext = FiniteField(p**d, "alpha")
Eext = E.base_extend(Fext)
P = Eext.random_element()
C = p**d
C * P
```

Timing results



Questions

- Can we do better to compute roots of polynomials over finite fields?
- Can we do better for non-prime finite fields?

Take home messages

- ...

Thank you!

References

A. Bostan, F. Morain, B. Salvy, and É. Schost.
Fast algorithms for computing isogenies between elliptic curves.
Mathematics of Computation, 77(263):1755–1778, 2008.

J.-M. Couveignes.
Hard homogeneous spaces.
preprint, 2006.

A. Rostovtsev and A. Stolbunov.
Public-key cryptosystem based on isogenies, 2006.