University of
Southern Denmark

Cryptographic Engineering

Elliptic-Curve Arithmetic

Ruben Niederhagen
(based on content by Peter Schwabe)

Department of Mathematics and Computer Science (IMADA)

Department of Mathematics
and Computer Science (IMADA)

- Let $\mathcal{G}$ be a cyclic, finite, abelian Group (written additively) and let P be a generator of $\mathcal{G}$.
- Alice chooses random $a \in \{0, \ldots, |\mathcal{G}| - 1\}$, computes aP, sends to Bob.
- Bob chooses random $b \in \{0, \ldots, |\mathcal{G}| - 1\}$, computes bP, sends to Alice.
- Alice computes joint key $a(bP)$.
- Bob computes joint key $b(aP)$.
- Discrete logarithm problem (DLP) in $\mathcal{G}$: given $kP \in \mathcal{G}$ and P, find k.
- Solving the DLP breaks security of Diffie-Hellman.

Groups with hard DLP:

- Traditional answer: $\mathbb{Z}_p^*$ with large prime-order subgroup.
- Modern answer: Elliptic curve over $\mathbb{F}_q$ with large prime-order subgroup.
- Sophisticated answer (not in this lecture): hyperelliptic curves of genus 2.

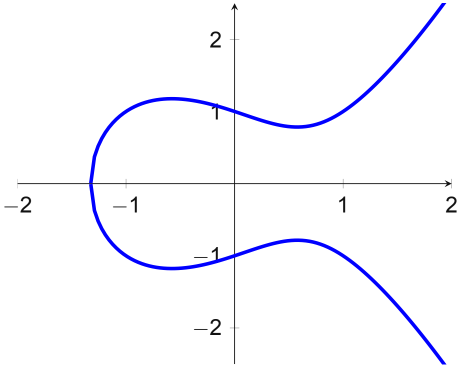
Typical view on elliptic curves:

Definition:
Let K be a field and let $a_1, a_2, a_3, a_4, a_6 \in K$.
Then the following equation defines an elliptic curve $\mathcal{E}$:

$$\mathcal{E} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

(if the discriminant Δ is not equal to zero).

This equation is called the Weierstrass form of an elliptic curve.



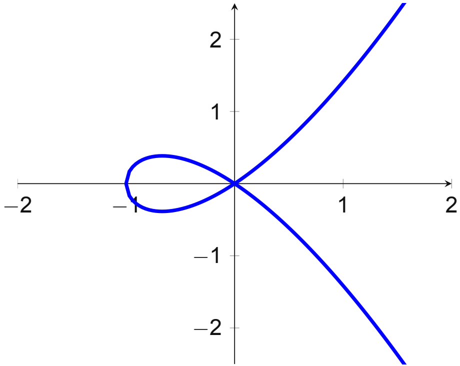
Typical view on elliptic curves:

Definition:
Let K be a field and let $a_1, a_2, a_3, a_4, a_6 \in K$.
Then the following equation defines an elliptic curve $\mathcal{E}$:

$$\mathcal{E} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

(if the discriminant Δ is not equal to zero).

This equation is called the Weierstrass form of an elliptic curve.



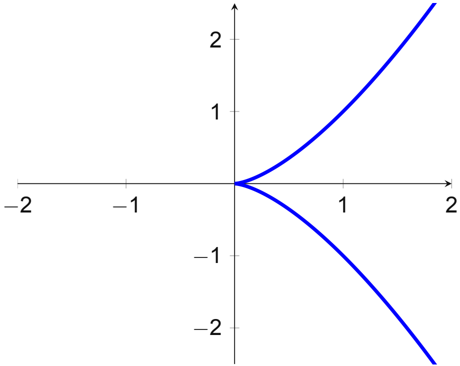
Typical view on elliptic curves:

Definition:
Let K be a field and let $a_1, a_2, a_3, a_4, a_6 \in K$.
Then the following equation defines an elliptic curve $\mathcal{E}$:

$$\mathcal{E} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

(if the discriminant Δ is not equal to zero).

This equation is called the Weierstrass form of an elliptic curve.



Typical view on elliptic curves:

Definition:

Let K be a field and let $a_1, a_2, a_3, a_4, a_6 \in K$. Then the following equation defines an elliptic curve $\mathcal{E}$:

$$\mathcal{E} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

(if the discriminant Δ is not equal to zero).

This equation is called the Weierstrass form of an elliptic curve.

Characteristic $\neq 2, 3$:

If $\text{char}(K) \neq 2, 3$ (e.g., $K = \mathbb{F}_p, p > 3$) we can use a simplified equation:

$$\mathcal{E} : y^2 = x^3 + ax + b$$

Characteristic 2:

If $\text{char}(K) = 2$ (e.g., $K = \mathbb{F}_{2^n}$) we can (usually) use a simplified equation:

$$\mathcal{E} : y^2 + xy = x^3 + ax^2 + b$$

Setup for cryptography:

- Choose $K = \mathbb{F}_q$.
- Consider the set of $\mathbb{F}_q$-rational points:

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6\} \cup \{\mathcal{O}\}$$

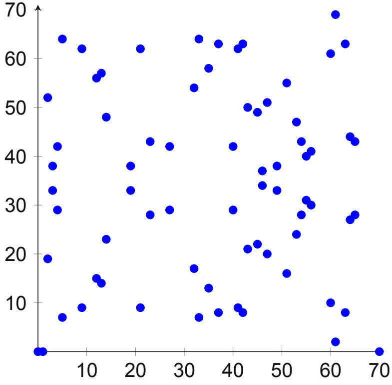
- The element $\mathcal{O}$ is the “point at infinity”.
- This set forms a group (together with addition law).
- Order of this group: $|E(\mathbb{F}_q)| \approx |\mathbb{F}_q|$

Group Law

Department of Mathematics
and Computer Science (IMADA)

Example curve: $y^2 = x^3 - x$ over $\mathbb{F}_{71}$

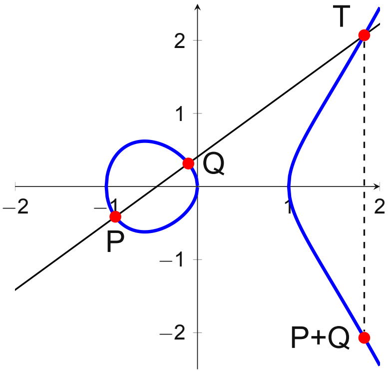
Graph of E over $\mathbb{F}_{71}$:



Group Law

Example curve: $y^2 = x^3 - x$ over $\mathbb{R}$

Graph of E over $\mathcal{R}$:



Addition of Points:

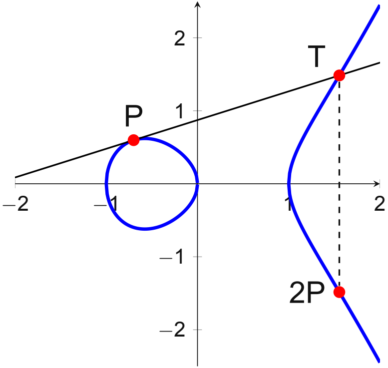
Add points P and Q:

- Compute line through the two points.
- Determine third intersection $T = (x_T, y_T)$ with the elliptic curve.
- Result of the addition: $P + Q = (x_T, -y_T)$.

Department of Mathematics
and Computer Science (IMADA)

Example curve: $y^2 = x^3 - x$ over $\mathbb{R}$

Graph of E over $\mathcal{R}$:



Doubling of Points:

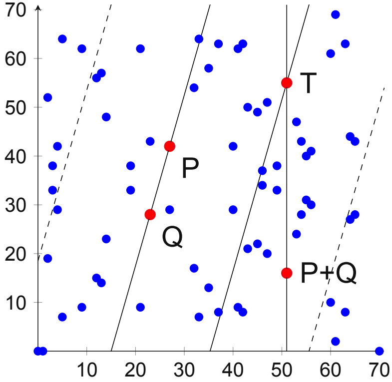
Double the point P:

- Compute Compute the tangent on P.
- Determine second intersection $T = (x_T, y_T)$ with the elliptic curve.
- Result of the doubling: $2P = (x_T, -y_T)$.

Department of Mathematics
and Computer Science (IMADA)

Example curve: $y^2 = x^3 - x$ over $\mathbb{F}_{71}$

Graph of E over $\mathbb{F}_{71}$:



Addition of Points:

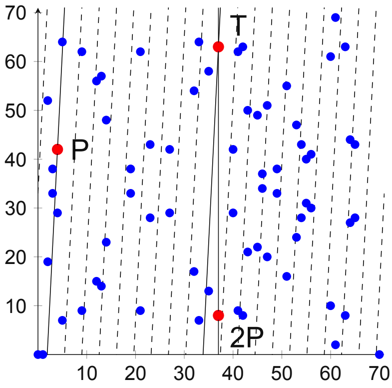
Add points P and Q:

- Compute line through the two points.
- Determine third intersection $T = (x_T, y_T)$ with the elliptic curve.
- Result of the addition: $P + Q = (x_T, -y_T)$.

Group Law

Example curve: $y^2 = x^3 - x$ over $\mathbb{F}_{71}$

Graph of E over $\mathbb{F}_{71}$:



Doubling of Points:

Double the point P:

- Compute Compute the tangent on P.
- Determine second intersection $T = (x_T, y_T)$ with the elliptic curve.
- Result of the doubling: $2P = (x_T, -y_T)$.

Group law in formulas:

Curve equation: $y^2 = x^3 + ax + b$

Point addition:

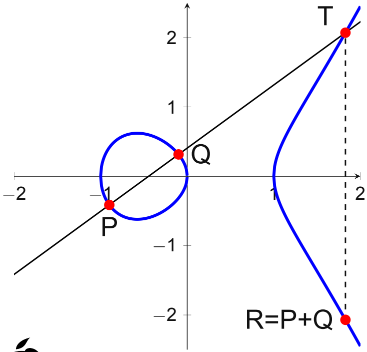
$P = (x_P y_P), Q = (x_Q, y_Q)$

Group law in formulas:

Curve equation: $y^2 = x^3 + ax + b$

Point addition:

$P = (x_P y_P), Q = (x_Q, y_Q)$



$$y = \lambda x + b$$

$$\boxed{\lambda = \frac{y_Q - y_P}{x_Q - x_P}}$$

$$y_P = \lambda x_P + b$$

$$b = y_P - \lambda x_P$$

$$y = \lambda x + y_P - \lambda x_P$$

$$\boxed{y = \lambda(x - x_P) + y_P}$$

$$\left(\lambda(x - x_P) + y_P\right)^2 = x^3 + ax + b$$

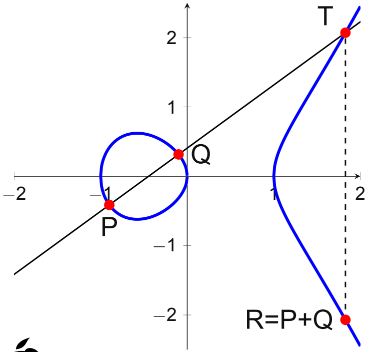
$$x^3 - \lambda^2 x^2 + (a + 2\lambda^2 x_P - 2\lambda y_P)x + b - (\lambda x_P - y_P)^2 = 0$$

Group law in formulas:

Curve equation: $y^2 = x^3 + ax + b$

Point addition:

$P = (x_P y_P), Q = (x_Q, y_Q)$



Vieta's formula:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0$$

For roots $r_1, r_2, \ldots, r_n$ we have:

$$r_1 + r_2 + \cdots + r_n = -\frac{a_{n-1}}{a_n}$$

$$x_P + x_Q + x_T = \lambda^2$$

$$x_T = x_R = \lambda^2 - x_P - x_Q$$

Line equation:

$$y_T = \lambda(x_T - x_P) + y_P$$

$$y_R = -y_T = \lambda(x_P - x_R) - y_P$$

Group law in formulas:

Curve equation: $y^2 = x^3 + ax + b$

Point addition:

$P = (x_P y_P), Q = (x_Q, y_Q)$
$\rightarrow P + Q = R = (x_R, y_R)$ with

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q$$

$$y_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_R) - y_P$$

Group law in formulas:

Curve equation: $y^2 = x^3 + ax + b$

Point addition:

$P = (x_P y_P), Q = (x_Q, y_Q)$
$\rightarrow P + Q = R = (x_R, y_R)$ with

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q$$

$$y_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_R) - y_P$$

Point doubling:

$P = (x_P y_P)$
$\rightarrow 2P = (x_R, y_R)$ with

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P$$

$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P$$

More Weierstrass curve group law:

- Neutral element is $\mathcal{O}$.
- Inverse of a point (x, y) is $(x, -y)$.
- Note: Formulas don't work for $P + (-P)$ and also don't work for $\mathcal{O}$.
- Need to distinguish these cases!
- "Uniform" addition law in Hışıl's Ph.D. thesis[1], Section 5.5.2:
  - Move special cases to other points.
  - Not safe to use on arbitrary input points!
- Formulas for curves over $\mathbb{F}_{2^k}$ look slightly different, but same special cases.

[1] http://eprints.qut.edu.au/33233/

Security requirements for ECC:

- $\ell = |E(\mathbb{F}_q)|$ must have large prime-order subgroup.
- For n bits of security we need $2n$-bit prime-order subgroup.
- It must be impossible to transfer DLP to less secure groups:
  - ℓ must not be equal to q.
  - We need $\ell \mid q^k - 1$ for *large* k.

Finding a curve:

- Fix finite field $\mathbb{F}_q$ of suitable size.
- Fix curve parameter a (quite common: $a = -3$).
- Pick curve parameter b until E fulfills desired properties.
- This requires efficient "point counting".
- This requires efficient factorization or primality proving.

Standard Curves

“The nice thing about standards is that you have so many to choose from.”
– Andrew S. Tanenbaum

- Various standardized curves, most well-known: NIST curves:
  - Big-prime field curves with 192, 224, 256, 384, and 521 bits.
  - Binary curves with 163, 233, 283, 409, and 571 bits.
  - Binary Koblitz curves with 163, 233, 283, 409, and 571 bits.
- SECG curves (Certicom), prime-field and binary curves.
- Brainpool curves (BSI), only prime-field curves.
- FRP256v1 (ANSSI), one prime-field curve (256 bits).
- Curve25519 (Bernstein; RFC 7748, FIPS 186-5 draft 2019), prime-field curve.

Department of Mathematics
and Computer Science (IMADA)

Curves over big-prime fields:

- Many fields of a given size
  $\Rightarrow$ many curves.
- Efficient in software (can use hardware multipliers).
- Less efficient in hardware.

Curves over binary fields:

- Important for security:
  Exponent k of $\mathbb{F}_{p^k}$ has to be prime.
- Not many fields
  (not that many curves).
- More efficient in hardware.
- Efficient in software only on some microarchitectures.
- Hard to implement securely in software on some other microarchitectures.

Putting it all together:

- Choose security level (e.g., 128 bits).
- Decide whether you want binary or big-prime field arithmetic; let's say big prime.
- Pick corresponding standard curve, e.g., NIST-P256.
- Implement field arithmetic.
- Implement ECC addition and doubling.
- Implement scalar multiplication ("double and add" – next lecture).
- You're done with BAD (!) ECDH software.

Problem I: Inversion

- Adding $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ needs an inversion in $\mathbb{F}_q$.
- Inversions are expensive.
- Constant-time inversions are even more expensive.

Solution — projective coordinates:

- Store fractions of elements of $\mathbb{F}_q$, invert only once at the end.
- Represent points in projective coordinates:
  $P = (X_P : Y_P : Z_P)$ with $x_P = X_P/Z_P$ and $y_P = Y_P/Z_P$.
- The point $(1 : 1 : 0)$ is the point at infinity.
- Also possible — weighted projective coordinates:
  - Jacobian coordinates: $P = (X_P : Y_P : Z_P)$ with $x_P = X_P/Z_P^2$ and $y_P = Y_P/Z_P^3$.
  - López-Dahab coordinates: $P = (X_P : Y_P : Z_P)$ with $x_P = X_P/Z_P$ and $y_P = Y_P/Z_P^2$.
    (for binary curves)
- Important: *Never* send projective representation, *always* convert to affine!

Problem II: Group-Law Special Cases

Addition of $P + Q$:

- If $P = \mathcal{O}$ return Q.
- Else if $Q = \mathcal{O}$ return P.
- Else if $P = Q$ call doubling routine.
- Else if $P = -Q$ return $\mathcal{O}$.
- Else use addition formulas.

Doubling P:

- If $P = \mathcal{O}$ return P.
- Else if $y_P = 0$ return $\mathcal{O}$.
- Else use doubling formulas.

- Constant-time implementations of this are hard.
- Good news: Can avoid the checks when computing $k \cdot P$ and $k < |E(\mathbb{F}_q)|$.
- Bad news: Side-channel countermeasures use $k > |E(\mathbb{F}_q)|$.
- More bad news: Doesn't work for multi-scalar multiplication (next lecture).
- Baseline: *Simple* implementations are likely to be wrong or insecure!

Solution II.A: Montgomery Curves

- Use Montgomery curve: $E_M : By^2 = x^3 + Ax^2 + x$.
- Use x-coordinate-only differential addition chain (“Montgomery ladder”, next lecture).
- Advantages:
  - Works on all inputs, no special cases.
  - Very regular structure, easy to protect against timing attacks.
  - Point compression/decompression for free.
  - Easy to implement, harder to screw up in hard-to-detect ways.
  - Simple implementations are likely to be correct and secure.
- Disadvantages:
  - Not all curves can be converted to Montgomery shape.
  - Always have a cofactor of at least 4.
  - Ladders on general Weierstrass curves are much less efficient.
  - We only get the x coordinate of the result, tricky for signatures.
  - Can reconstruct y, but that involves some additional cost.

Solution II.B: (Twisted) Edwards Curves

- Edwards, 2007: New form for elliptic curves (“Edwards curves”).
- Bernstein, Lange, 2007: Very fast addition and doubling on these curves.
- Bernstein, Birkner, Joye, Lange, Peters, 2008:
  Generalize the idea to “twisted Edwards curves”.
- Core advantage of (twisted) Edwards curves — complete group law:
  - No need to handle special cases.
  - No “point at infinity” to work with.
- Can speed up doubling, but addition formulas work for $P + P$.
- Efficient transformation from Weierstrass to (twisted) Edwards only for some curves.
- Always efficient: Transformation between Montgomery and twisted Edwards curves.
- Again: Simple implementations are likely to be correct and secure.
- Disadvantage: Always have a cofactor of at least 4.

So, what's the deal with the cofactor?


MONERO

Community Crowdfunding Vulnerability R

Get Started - Downloads Blog Comm

Disclosure of a Major Bug in CryptoNote Based Currencies

Posted by: luigi1111 and Riccardo "fluffypony" Spagni

May 17, 2017

Overview

In Monero we've discovered and patched a critical bug that affects all CryptoNote-based cryptocurrencies, and allows for the creation of an unlimited number of coins in a way that is undetectable to an observer unless they know about the fatal flaw and can search for it.

- Protocols need to be careful to avoid subgroup attacks.
- Monero screwed this up, which allowed double-spending.
- Elegant solution: "Ristretto" encoding based on Hamburg's "Decaf", see: `https://ristretto.group/`.

- Bosma, Lenstra, 1995: Complete group law for Weierstrass curves.
- Problem: Extremely inefficient.
- Renes, Costello, Batina, 2016: Fast complete group law for Weierstrass curves.
- Less efficient than (twisted) Edwards.
- Covers all curves.

Problem III: Wrong-Curve Attacks

Reminder: $y^2 = x^3 + ax + b$ and b does not appeat in addition formulae.

ECDH attack scenario:

- Alice sends point on different (insecure) curve with small subgroup.
- Bob computes "shared key" in that small subgroup.
- Alice learns "shared key" through brute force.
- Alice learns Bob's secret scalar modulo the order of the small subgroup.

Countermeasures:

- Check that input point is on the curve (functional tests will miss this!).
- Send compressed points $(x, \text{parity}(y))$.
  Decompression returns (x, y) on the curve or fails.
- Send only x (Montgomery ladder); but: x could still be on the "twist" of E.
  Make sure that the twist is also secure ("twist security").

Problem IV: Backdoors in standards?

"I no longer trust the [NIST Elliptic Curves] constants. I believe the NSA has manipulated them through their relationships with industry."

— Bruce Schneier, 2013.

- There are concerns that NSA might have put a backdoor in Dual_EC_DRBG.
- More details at `https://projectbullrun.org/dual-ec/`.
- More details in a later lecture.

Problem IV: Backdoors in standards?

“I no longer trust the [NIST Elliptic Curves] constants. I believe the NSA has manipulated them through their relationships with industry. ”

— Bruce Schneier, 2013.

- Constants of NIST curves (and other standards) have been obtained by hashing random values.
- No-backdoor claim: We know the preimages.
- Possible attack if you know a class of vulnerable curves: Generate random seeds until you have found a vulnerable (and seemingly secure) curve.
- Fact: There are no known insecurities of NIST curves.
- Fact: There is no proof that there are no intentional vulnerabilities in NIST curves.
- For more details, see BADA55 elliptic curves: `http://bada55.cr.yp.to/`.

Department of Mathematics
and Computer Science (IMADA)

Choosing a safe curve:
Overview of various elliptic curves and thorough security analysis by Bernstein and Lange:

`https://safecurves.cr.yp.to/`

(Doesn't list cofactor-1 curves, so best to combine with Ristretto.)

Point representation and arithmetic:
Collection of elliptic-curve shapes, point representations and group-operation formulas by Bernstein and Lange:

`https://www.hyperelliptic.org/EFD/`