Engineering Cryptographic Software

Elliptic-Curve Arithmetic

Peter Schwabe

January 2026

Where were we?

- For scalar multiplication, we assumed a group G
  - of finite order ℓ,
  - that is commutative (Abelian),
  - that is cyclic with generator P, and
  - in which the **discrete-logarithm problem is hard**.

Where were we?

- For scalar multiplication, we assumed a group G
  - of finite order ℓ,
  - that is commutative (Abelian),
  - that is cyclic with generator P, and
  - in which the **discrete-logarithm problem is hard**.
- Today: make this group concrete

Fields

Definition

A set S together with two operations $(+, \cdot)$ is called a *field* $K = (S, +, \cdot)$ if

- $(S, +)$ is an Abelian group
- $(S \setminus \{0\}, \cdot)$ is an Abelian group, where 0 is the neutral element of $(S, +)$
- For all $a, b, c \in S$ it holds that $a \cdot (b + c) = a \cdot b + a \cdot c$ (distributivity)

Fields

Definition

A set S together with two operations $(+, \cdot)$ is called a *field* $K = (S, +, \cdot)$ if

- $(S, +)$ is an Abelian group
- $(S \setminus \{0\}, \cdot)$ is an Abelian group, where 0 is the neutral element of $(S, +)$
- For all $a, b, c \in S$ it holds that $a \cdot (b + c) = a \cdot b + a \cdot c$ **(distributivity)**

- Consider n-fold addition of 1, so, $n \cdot 1 = \underbrace{1 + 1 + 1 + \cdots + 1}_{n \text{ times}}$

Fields

Definition

A set S together with two operations $(+, \cdot)$ is called a *field* $K = (S, +, \cdot)$ if

- $(S, +)$ is an Abelian group
- $(S \setminus \{0\}, \cdot)$ is an Abelian group, where 0 is the neutral element of $(S, +)$
- For all $a, b, c \in S$ it holds that $a \cdot (b + c) = a \cdot b + a \cdot c$ **(distributivity)**

- Consider n-fold addition of 1, so, $n \cdot 1 = \underbrace{1 + 1 + 1 + \cdots + 1}_{n \text{ times}}$
  - If there is no n such that $n \cdot 1 = 0$, then the characterisic of K is $\text{char}(K) = 0$

Fields

Definition

A set S together with two operations $(+, \cdot)$ is called a *field* $K = (S, +, \cdot)$ if

- $(S, +)$ is an Abelian group
- $(S \setminus \{0\}, \cdot)$ is an Abelian group, where 0 is the neutral element of $(S, +)$
- For all $a, b, c \in S$ it holds that $a \cdot (b + c) = a \cdot b + a \cdot c$ (distributivity)

- Consider n-fold addition of 1, so, $n \cdot 1 = \underbrace{1 + 1 + 1 + \cdots + 1}_{n \text{ times}}$
  - If there is no n such that $n \cdot 1 = 0$, then the characterisic of K is $\text{char}(K) = 0$
  - Otherwise, $\text{char}(K) = p$ for the smallest p such that $p \cdot 1 = 0$

Fields

Definition

A set S together with two operations $(+, \cdot)$ is called a *field* $K = (S, +, \cdot)$ if

- $(S, +)$ is an Abelian group
- $(S \setminus \{0\}, \cdot)$ is an Abelian group, where 0 is the neutral element of $(S, +)$
- For all $a, b, c \in S$ it holds that $a \cdot (b + c) = a \cdot b + a \cdot c$ (distributivity)

- Consider n-fold addition of 1, so, $n \cdot 1 = \underbrace{1 + 1 + 1 + \cdots + 1}_{n \text{ times}}$
  - If there is no n such that $n \cdot 1 = 0$, then the characterisic of K is char$(K) = 0$
  - Otherwise, char$(K) = p$ for the smallest p such that $p \cdot 1 = 0$
  - If char$(K) = p \neq 0$, then p is prime

Examples

- The rationals $(\mathbb{Q}, +, \cdot)$ are a field

Examples

- The rationals $(\mathbb{Q}, +, \cdot)$ are a field
- The integers $(\mathbb{Z}, +, \cdot)$ are *not* a field
  - Remember, we don't have multiplicative inverses

Examples

- The rationals $(\mathbb{Q}, +, \cdot)$ are a field
- The integers $(\mathbb{Z}, +, \cdot)$ are *not* a field
  - Remember, we don't have multiplicative inverses
- The reals $(\mathbb{R}, +, \cdot)$ are a field

Examples

- The rationals $(\mathbb{Q}, +, \cdot)$ are a field
- The integers $(\mathbb{Z}, +, \cdot)$ are *not* a field
  - Remember, we don't have multiplicative inverses
- The reals $(\mathbb{R}, +, \cdot)$ are a field
- $\{0, \ldots, q-1\}$ together with addition and multiplication modulo q is a field **if q is prime**
  - We typically denote this field $\mathbb{F}_q$
  - The characteristic of $\mathbb{F}_q$ is q

Examples

- The rationals $(\mathbb{Q}, +, \cdot)$ are a field
- The integers $(\mathbb{Z}, +, \cdot)$ are *not* a field
  - Remember, we don't have multiplicative inverses
- The reals $(\mathbb{R}, +, \cdot)$ are a field
- $\{0, \dots, q-1\}$ together with addition and multiplication modulo q is a field **if q is prime**
  - We typically denote this field $\mathbb{F}_q$
  - The characteristic of $\mathbb{F}_q$ is q
- The smallest field is $\{0, 1\}$ with addition and multiplication modulo 2
  - Addition is XOR
  - Multiplication is AND

Suitable/common groups

Groups with hard DLP

- Traditional answer (DH76 paper): $\mathbb{Z}_p^*$ with large prime-order subgroup

Suitable/common groups

Groups with hard DLP

- Traditional answer (DH76 paper): $\mathbb{Z}_p^*$ with large prime-order subgroup
  - Let $(G, \circ)$ be a group
  - Let H by a subset of G
  - Then $(H, \circ)$ is a *subgroup* of G if it is a group

Suitable/common groups

Groups with hard DLP

- Traditional answer (DH76 paper): $\mathbb{Z}_p^*$ with large prime-order subgroup
  - Let $(G, \circ)$ be a group
  - Let H by a subset of G
  - Then $(H, \circ)$ is a *subgroup* of G if it is a group
- Modern answer: Elliptic curve over $\mathbb{F}_q$ with large prime-order subgroup

Suitable/common groups

Groups with hard DLP

- Traditional answer (DH76 paper): $\mathbb{Z}_p^*$ with large prime-order subgroup
  - Let $(G, \circ)$ be a group
  - Let H by a subset of G
  - Then $(H, \circ)$ is a *subgroup* of G if it is a group
- Modern answer: Elliptic curve over $\mathbb{F}_q$ with large prime-order subgroup
- Sophisticated answer (not in this lecture): hyperelliptic curves of genus 2

Suitable/common groups

Groups with hard DLP

- Traditional answer (DH76 paper): $\mathbb{Z}_p^*$ with large prime-order subgroup
  - Let $(G, \circ)$ be a group
  - Let H by a subset of G
  - Then $(H, \circ)$ is a *subgroup* of G if it is a group
- 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 with char$(K) \notin \{2, 3\}$ and let $a, b \in K$. Then the following equation defines an elliptic curve E:

$$E : y^2 = x^3 + ax + b,$$

if the discriminant $\Delta = -64a^3 - 432b^2$ of E is not equal to zero. This equation is called the *short Weierstrass form* of an elliptic curve.

Rational points

Setup for cryptography

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

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

Rational points

Setup for cryptography

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

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

- The element $\mathcal{O}$ is the “point at infinity”

Setup for cryptography

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

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

- The element $\mathcal{O}$ is the “point at infinity”
- This set forms a group (together with addition law)

Rational points

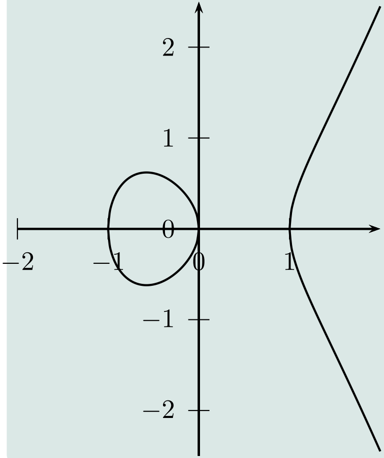
Setup for cryptography

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

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

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

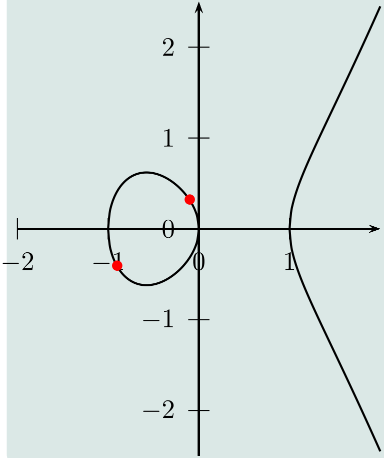
Graph of E over $\mathbb{R}$



Addition of points

- Add points
  $P = (-0,9; -0,4135)$ and
  $Q = (-0,1; 0,3146)$

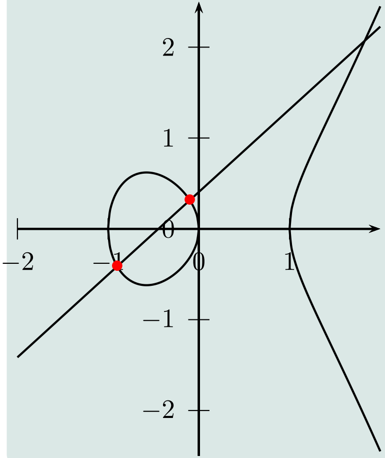
Graph of E over $\mathbb{R}$



Addition of points

- Add points $P = (-0,9; -0,4135)$ and $Q = (-0,1; 0,3146)$
- Compute line through the two points

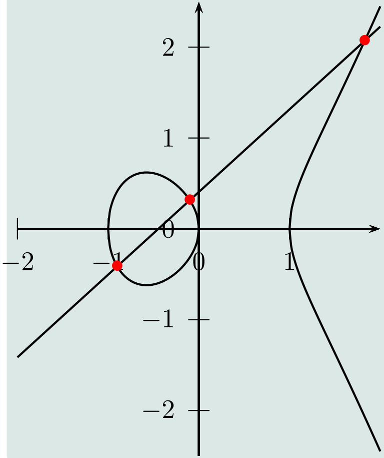
Graph of E over $\mathbb{R}$



Addition of points

- Add points $P = (-0,9; -0,4135)$ and $Q = (-0,1; 0,3146)$
- Compute line through the two points
- Determine third intersection $T = (x_T, y_T)$ with the elliptic curve

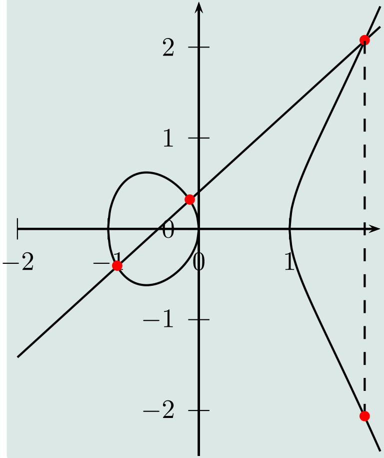
Graph of E over $\mathbb{R}$



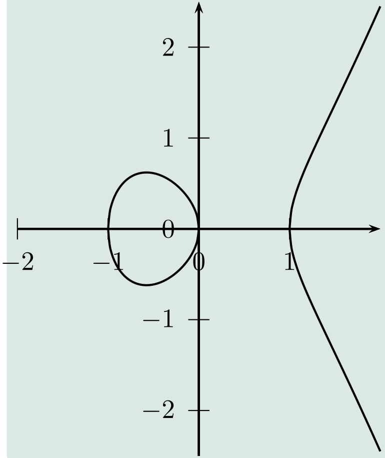
Addition of points

- Add points $P = (-0,9; -0,4135)$ and $Q = (-0,1; 0,3146)$
- Compute line through the two points
- Determine third intersection $T = (x_T, y_T)$ with the elliptic curve
- Result of the addition: $P + Q = (x_T, -y_T)$

Graph of E over $\mathbb{R}$



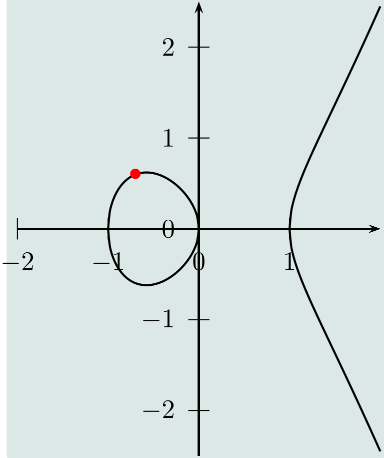
Graph of E over $\mathbb{R}$



Point doubling

- Double the point $P = (-0.7, 0.5975)$

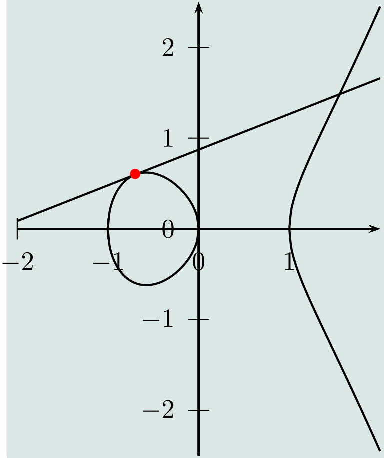
Graph of E over $\mathbb{R}$



Point doubling

- Double the point $P = (-0.7, 0.5975)$
- Compute the tangent on P

Graph of E over $\mathbb{R}$



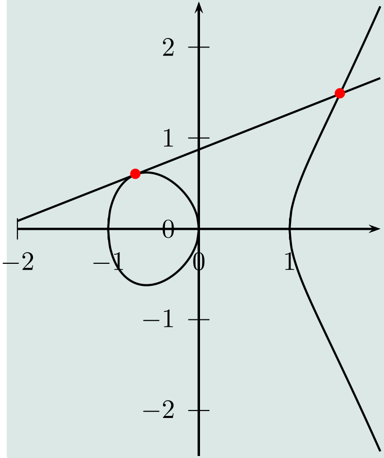
The group law

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

Point doubling

- Double the point $P = (-0.7, 0.5975)$
- Compute the tangent on P
- Determine second intersection $T = (x_T, y_T)$ with the elliptic curve

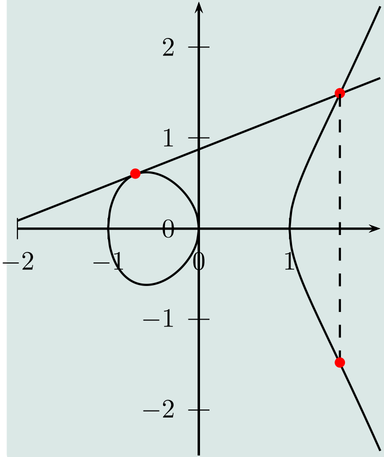
Graph of E over $\mathbb{R}$



Point doubling

- Double the point $P = (-0.7, 0.5975)$
- Compute the tangent on P
- Determine second intersection $T = (x_T, y_T)$ with the elliptic curve
- Result of the addition: $P + Q = (x_T, -y_T)$

Graph of E over $\mathbb{R}$



Group law in formulas

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

Group law in formulas

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

Point addition

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

Group law in formulas

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

Point addition

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

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

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

Group law in formulas

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

Point addition

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

- $P = (x_P, y_P), 2P = (x_R, y_R)$ 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 $(x, -y)$

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 $P + (-P)$, also don't work for $\mathcal{O}$
- Implementations need to distinguish these cases!

Finding a suitable curve

Security requirements for ECC

- $\ell = |E(\mathbb{F}_q)|$ must have large prime-order subgroup (Pohlig-Hellman)
- For n bits of security we need $2n$-bit prime-order subgroup (Pollard's ρ)

Finding a suitable curve

Security requirements for ECC

- $\ell = |E(\mathbb{F}_q)|$ must have large prime-order subgroup (Pohlig-Hellman)
- For n bits of security we need $2n$-bit prime-order subgroup (Pollard's ρ)
- Impossible to transfer DLP to less secure groups:
  - ℓ must not be equal to q
  - We need $\ell \nmid p^k - 1$ for small k

Finding a suitable curve

Security requirements for ECC

- $\ell = |E(\mathbb{F}_q)|$ must have large prime-order subgroup (Pohlig-Hellman)
- For n bits of security we need $2n$-bit prime-order subgroup (Pollard's ρ)
- Impossible to transfer DLP to less secure groups:
  - ℓ must not be equal to q
  - We need $\ell \nmid p^k - 1$ for small 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

Standardized curves

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

Standardized 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

Standardized 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

Standardized 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

Standardized 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)

Standardized 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)
- SM2 (China), one prime-field curve (256 bits)

Putting it together

- Choose security level (e.g., 128 bits)

Putting it together

- Choose security level (e.g., 128 bits)
- Pick standard curve, e.g., NIST-P256

Putting it together

- Choose security level (e.g., 128 bits)
- Pick standard curve, e.g., NIST-P256
- Implement field arithmetic (more tomorrow)

Putting it together

- Choose security level (e.g., 128 bits)
- Pick standard curve, e.g., NIST-P256
- Implement field arithmetic (more tomorrow)
- Implement ECC addition and doubling

Putting it together

- Choose security level (e.g., 128 bits)
- Pick standard curve, e.g., NIST-P256
- Implement field arithmetic (more tomorrow)
- Implement ECC addition and doubling
- Implement scalar multiplication

Putting it together

- Choose security level (e.g., 128 bits)
- Pick standard curve, e.g., NIST-P256
- Implement field arithmetic (more tomorrow)
- Implement ECC addition and doubling
- Implement scalar multiplication
- Maybe implement fixed-basepoint scalar multiplication

Putting it together

- Choose security level (e.g., 128 bits)
- Pick standard curve, e.g., NIST-P256
- Implement field arithmetic (more tomorrow)
- Implement ECC addition and doubling
- Implement scalar multiplication
- Maybe implement fixed-basepoint scalar multiplication
- You're done with ECDH software

Putting it together

- Choose security level (e.g., 128 bits)
- Pick standard curve, e.g., NIST-P256
- Implement field arithmetic (more tomorrow)
- Implement ECC addition and doubling
- Implement scalar multiplication
- Maybe implement fixed-basepoint scalar multiplication
- You're done with BAD (!) ECDH software

Problem I: inversions

Inversions

- 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

Problem I: inversions

Inversions

- 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

Problem I: inversions

Inversions

- 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

Problem I: inversions

Inversions

- 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 $y_P = Y_P/Z_P^3$

Problem I: inversions

Inversions

- 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 $y_P = Y_P/Z_P^3$
- Important: Never *send* projective representation, always convert to affine!

Problem II: group-law special cases

- Addition of $P + Q$ needs to distinguish different cases:
  - 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

Problem II: group-law special cases

- Addition of $P + Q$ needs to distinguish different cases:
  - 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
- Similar for doubling P:
  - If $P = \mathcal{O}$ return P
  - Else if $y_P = 0$ return $\mathcal{O}$
  - Else use doubling formulas

Problem II: group-law special cases

- Addition of $P + Q$ needs to distinguish different cases:
  - 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
- Similar for 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 horrible

Problem II: group-law special cases

- Addition of $P + Q$ needs to distinguish different cases:
  - 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
- Similar for 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 horrible
- Good news: Can avoid the checks when computing $k \cdot P$ and $k < |E(\mathbb{F}_q)|$

Problem II: group-law special cases

- Addition of $P+Q$ needs to distinguish different cases:
  - 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
- Similar for 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 horrible
- Good news: Can avoid the checks when computing $k \cdot P$ and $k < |E(\mathbb{F}_q)|$
- Bad news: Side-channel countermeasures use $k > |E(\mathbb{F}_q)|$

Problem II: group-law special cases

- Addition of $P + Q$ needs to distinguish different cases:
  - 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
- Similar for 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 horrible
- 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

Problem II: group-law special cases

- Addition of $P + Q$ needs to distinguish different cases:
  - 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
- Similar for 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 horrible
- 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
- Baseline: *simple* implementations are likely to be wrong or insecure

Solution I: Montgomery curves

- Consider elliptic curves of the form $By^2 = x^3 + Ax^2 + x$.
- Montgomery in 1987 showed how to perform x-coordinate-based arithmetic:
  - Given the x-coordinate x_P of P, and
  - given the x-coordinate x_Q of Q, and
  - given the x-coordinate x_{P-Q} of $P - Q$

Solution I: Montgomery curves

- Consider elliptic curves of the form $By^2 = x^3 + Ax^2 + x$.
- Montgomery in 1987 showed how to perform x-coordinate-based arithmetic:
  - Given the x-coordinate x_P of P, and
  - given the x-coordinate x_Q of Q, and
  - given the x-coordinate x_{P-Q} of $P - Q$
  - compute the x-coordinate x_R of $R = P + Q$

Solution I: Montgomery curves

- Consider elliptic curves of the form $By^2 = x^3 + Ax^2 + x$.
- Montgomery in 1987 showed how to perform x-coordinate-based arithmetic:
  - Given the x-coordinate x_P of P, and
  - given the x-coordinate x_Q of Q, and
  - given the x-coordinate x_{P-Q} of $P - Q$
  - compute the x-coordinate x_R of $R = P + Q$
- This is called *differential addition*

Solution I: Montgomery curves

- Consider elliptic curves of the form $By^2 = x^3 + Ax^2 + x$.
- Montgomery in 1987 showed how to perform x-coordinate-based arithmetic:
  - Given the x-coordinate x_P of P, and
  - given the x-coordinate x_Q of Q, and
  - given the x-coordinate x_{P-Q} of $P - Q$
  - compute the x-coordinate x_R of $R = P + Q$
- This is called *differential addition*
- Less efficient differential-addition formulas for other curve shapes

Solution I: Montgomery curves

- Consider elliptic curves of the form $By^2 = x^3 + Ax^2 + x$.
- Montgomery in 1987 showed how to perform x-coordinate-based arithmetic:
  - Given the x-coordinate x_P of P, and
  - given the x-coordinate x_Q of Q, and
  - given the x-coordinate x_{P-Q} of $P - Q$
  - compute the x-coordinate x_R of $R = P + Q$
- This is called *differential addition*
- Less efficient differential-addition formulas for other curve shapes
- Use to efficiently compute the x-coordinate of kP given only the x-coordinate of P
- For this, let's use projective representation $(X : Z)$ with $x = (X/Z)$

One Montgomery “ladder step”

const $a24 = (A+2)/4$ (A from the curve equation)
function LADDERSTEP($x_{Q-P}, X_P, Z_P, X_Q, Z_Q$)
$t_1 \leftarrow X_P + Z_P$
$t_6 \leftarrow t_1^2$
$t_2 \leftarrow X_P - Z_P$
$t_7 \leftarrow t_2^2$
$t_5 \leftarrow t_6 - t_7$
$t_3 \leftarrow X_Q + Z_Q$
$t_4 \leftarrow X_Q - Z_Q$
$t_8 \leftarrow t_4 \cdot t_1$
$t_9 \leftarrow t_3 \cdot t_2$
$X_{P+Q} \leftarrow (t_8 + t_9)^2$
$Z_{P+Q} \leftarrow x_{Q-P} \cdot (t_8 - t_9)^2$
$X_{2P} \leftarrow t_6 \cdot t_7$
$Z_{2P} \leftarrow t_5 \cdot (t_7 + a24 \cdot t_5)$
return ($X_{2P}, Z_{2P}, X_{P+Q}, Z_{P+Q}$)
end function

The Montgomery ladder

Require: A scalar $0 \leq k \in \mathbb{Z}$ and the x-coordinate x_P of some point P
Ensure: (X_{kP}, Z_{kP}) fulfilling $x_{kP} = X_{kP}/Z_{kP}$

$x_1 = x_P$; $X_2 = 1$; $Z_2 = 0$; $X_3 = x_P$; $Z_3 = 1$
for $i \leftarrow n-1$ downto 0 **do**
  if bit i of k is 1 **then**
    $(X_3, Z_3, X_2, Z_2) \leftarrow$ LADDERSTEP$(x_1, X_3, Z_3, X_2, Z_2)$
  else
    $(X_2, Z_2, X_3, Z_3) \leftarrow$ LADDERSTEP$(x_1, X_2, Z_2, X_3, Z_3)$
  end if
end for
return X_2/Z_2

The Montgomery ladder (ctd.)

Require: A scalar $0 \leq k \in \mathbb{Z}$ and the x-coordinate x_P of some point P
Ensure: (X_{kP}, Z_{kP}) fulfilling $x_{kP} = X_{kP}/Z_{kP}$

$X_1 = x_P$; $X_2 = 1$; $Z_2 = 0$; $X_3 = x_P$; $Z_3 = 1$
for $i \leftarrow n-1$ downto 0 **do**
  $b \leftarrow$ bit i of s
  $c \leftarrow b \oplus p$
  $p \leftarrow b$
  $(X_2, X_3) \leftarrow$ CSWAP(X_2, X_3, c)
  $(Z_2, Z_3) \leftarrow$ CSWAP(Z_2, Z_3, c)
  $(X_2, Z_2, X_3, Z_3) \leftarrow$ LADDERSTEP$(x_1, X_2, Z_2, X_3, Z_3)$
end for
return X_2/Z_2

(Dis-)advantages of the Montgomery ladder

Advantages:

- Works on all inputs, no special cases

(Dis-)advantages of the Montgomery ladder

Advantages:

- Works on all inputs, no special cases
- Very regular structure, easy to protect against timing attacks
  - Replace the if statement by conditional swap
  - Be careful with constant-time swaps
- Point compression/decompression for free

(Dis-)advantages of the Montgomery ladder

Advantages:

- Works on all inputs, no special cases
- Very regular structure, easy to protect against timing attacks
  - Replace the if statement by conditional swap
  - Be careful with constant-time swaps
- 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

(Dis-)advantages of the Montgomery ladder

Advantages:

- Works on all inputs, no special cases
- Very regular structure, easy to protect against timing attacks
  - Replace the if statement by conditional swap
  - Be careful with constant-time swaps
- 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

(Dis-)advantages of the Montgomery ladder

Advantages:

- Works on all inputs, no special cases
- Very regular structure, easy to protect against timing attacks
  - Replace the if statement by conditional swap
  - Be careful with constant-time swaps
- 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: (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"

Solution II: (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

Solution II: (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 $P + P$

Solution II: (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 (for cryptography) transformation from Weierstrass to (twisted) Edwards only for some curves

Solution II: (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 (for cryptography) transformation from Weierstrass to (twisted) Edwards only for some curves
- Always efficient: transformation between Montgomery curves and twisted Edwards curves

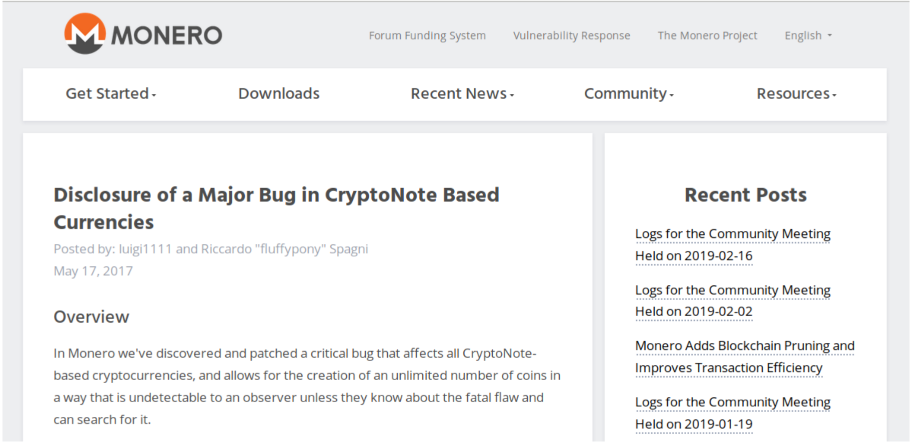
Solution II: (twisted) Edwards curves

- 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 (for cryptography) transformation from Weierstrass to (twisted) Edwards only for some curves
- Always efficient: transformation between Montgomery curves and twisted Edwards curves
- Again: simple implementations are likely to be correct and secure

Solution II: (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 (for cryptography) transformation from Weierstrass to (twisted) Edwards only for some curves
- Always efficient: transformation between Montgomery curves 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

Forum Funding System | Vulnerability Response | The Monero Project | English

Get Started | Downloads | Recent News | Community | Resources

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.

Recent Posts

- Logs for the Community Meeting Held on 2019-02-16
- Logs for the Community Meeting Held on 2019-02-02
- Monero Adds Blockchain Pruning and Improves Transaction Efficiency
- Logs for the Community Meeting Held on 2019-01-19

So, what's the deal with the cofactor?

- Protocols need to be careful to avoid subgroup attacks
- Monero screwed this up, which allowed double-spending
- Elegant solution: "Decaf" and "Ristretto" encoding by Hamburg, see:
  - `https://eprint.iacr.org/2015/673.pdf`
  - `https://ristretto.group`
  - `https://github.com/otrv4/libgoldilocks`
- This is also used in the code of `assignment2-ecdh25519`

Solution III: Complete group law on Weierstrass curves

- Bosma, Lenstra, 1995: complete group law for Weierstrass curves
- Problem: Extremely inefficient

Solution III: Complete group law on Weierstrass curves

- Bosma, Lenstra, 1995: complete group law for Weierstrass curves
- Problem: Extremely inefficient
- Renes, Costello, Batina, 2016: Much faster complete group law for Weierstrass curves
- Less efficient than (twisted) Edwards
- Overhead quite architecture-dependent (Schwabe, Sprenkels, 2019)
- Covers all curves

Problem III: Wrong-curve attacks

ECDH attack scenario

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

Problem III: Wrong-curve attacks

ECDH attack scenario

- Alice sends point on different (insecure) curve with small subgroup
- Bob computes “shared key” in that small subgroup
- Alice obtains “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!)

Problem III: Wrong-curve attacks

ECDH attack scenario

- Alice sends point on different (insecure) curve with small subgroup
- Bob computes "shared key" in that small subgroup
- Alice obtains "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

Problem III: Wrong-curve attacks

ECDH attack scenario

- Alice sends point on different (insecure) curve with small subgroup
- Bob computes "shared key" in that small subgroup
- Alice obtains "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.*

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.*

- It is pretty clear that NSA put a backdoor in Dual_EC_DRBG
- Constants of NIST curves have been obtained by hashing random values
- No-backdoor claim: We know the preimages

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.

- It is pretty clear that NSA put a backdoor in Dual_EC_DRBG
- Constants of NIST curves 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

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.

- It is pretty clear that NSA put a backdoor in Dual_EC_DRBG
- Constants of NIST curves 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

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.*

- It is pretty clear that NSA put a backdoor in Dual_EC_DRBG
- Constants of NIST curves 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

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.

- It is pretty clear that NSA put a backdoor in Dual_EC_DRBG
- Constants of NIST curves 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
- Question for ECC: who do you trust to pick the curve?

Point representation and arithmetic

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

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

Summary

- If you have to use Weierstraß (e.g., NIST) curves:
  - Use complete formulas by Renes-Costello-Batina
  - (Alternative: make sure that you don't trigger special cases)

Summary

- If you have to use Weierstraß (e.g., NIST) curves:
  - Use complete formulas by Renes-Costello-Batina
  - (Alternative: make sure that you don’t trigger special cases)
- If you can use Montgomery or twisted Edwards curves:
  - For ECDH, typically use Mongomery curve and ladder
  - For signatures, typically use twisted Edwards curve

Summary

- If you have to use Weierstraß (e.g., NIST) curves:
  - Use complete formulas by Renes-Costello-Batina
  - (Alternative: make sure that you don't trigger special cases)
- If you can use Montgomery or twisted Edwards curves:
  - For ECDH, typically use Mongomery curve and ladder
  - For signatures, typically use twisted Edwards curve
  - **assignment2-ecdh25519** does ECDH on twisted Edwards curve!

Summary

- If you have to use Weierstraß (e.g., NIST) curves:
  - Use complete formulas by Renes-Costello-Batina
  - (Alternative: make sure that you don't trigger special cases)
- If you can use Montgomery or twisted Edwards curves:
  - For ECDH, typically use Mongomery curve and ladder
  - For signatures, typically use twisted Edwards curve
  - assignment2-ecdh25519 does ECDH on twisted Edwards curve!
- If you can choose encoding for twisted Edwards points, use Decaf/Ristretto

Summary

- If you have to use Weierstraß (e.g., NIST) curves:
  - Use complete formulas by Renes-Costello-Batina
  - (Alternative: make sure that you don't trigger special cases)
- If you can use Montgomery or twisted Edwards curves:
  - For ECDH, typically use Mongomery curve and ladder
  - For signatures, typically use twisted Edwards curve
  - **assignment2-ecdh25519** does ECDH on twisted Edwards curve!
- If you can choose encoding for twisted Edwards points, use Decaf/Ristretto
- Most common Montgomery / twisted Edwards curve: Curve25519
  - Defined over finite field $\mathbb{F}_{2^{255}-19}$
  - Used in Montgomery form in X25519 ECDH
  - Used in twisted Edwards form in Ed25519 signatures