

# CSE 207B: Applied Cryptography

**Nadia Heninger**

UCSD

Fall 2025 Lecture 13

# Announcements

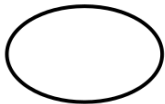
1. HW 6 is due next lecture.
2. HW 7 will be online shortly.

**Last time:**

- Digital signatures

**This time:**

- Elliptic curve cryptography



Ellipse



Elliptical

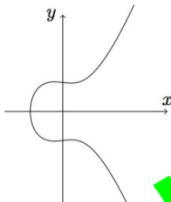


Eclipse



...

Ellipsis



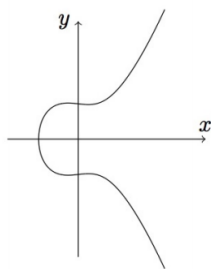
Elliptic  
Curve



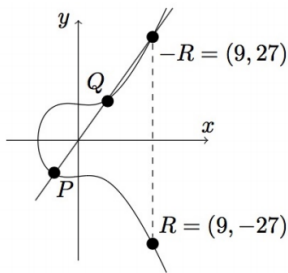
## Elliptic curves: Motivations

- Factoring and finite field discrete log both have subexponential-time algorithms.
- Current records: 829 bits for factoring, 795 bits for discrete log.
- Current recommendations: use 2048-bit public keys for RSA, Diffie-Hellman, or DSA.
- These are large, so the operations are slow.
- The best classical cryptanalysis we have for elliptic curves is much weaker, exponential time, so key sizes can be much smaller for the same security level.

# Points on an elliptic curve



(a) The curve

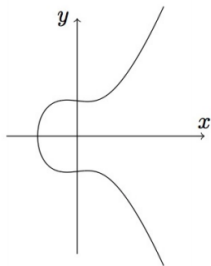


(b) Adding  $P = (-1, -3)$  and  $Q = (1, 3)$

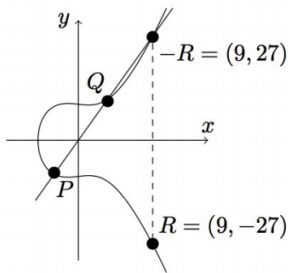
**Figure 15.1:** The curve  $y^2 = x^3 - x + 9$  over the reals (not drawn to scale)

- Every intro to elliptic curves includes a figure like this.
- The curve is defined by an equation  $y^2 = x^3 + ax + b$ .
- This picture plots this curve over  $\mathbb{R}^2$ .
- Classically, Diophantus and Poincaré were interested in rational points  $(x, y) \in \mathbb{Q}^2$  on this curve.

# Points on an elliptic curve



(a) The curve



(b) Adding  $P = (-1, -3)$  and  $Q = (1, 3)$

**Figure 15.1:** The curve  $y^2 = x^3 - x + 9$  over the reals (not drawn to scale)

- Poincare's method for finding rational points:
  - Take two rational points, define a line (like  $y = 3x$  above), and substitute to get a univariate cubic equation.
  - Since it has two rational roots already, the third root is also rational.
  - Get two new points for free:  $R$  and  $-R$ .
- Can define this as a group law:  $P + Q = -R$ .
- (The operation “+” means apply the above procedure.)

# Elliptic curves over finite fields

- For cryptography, we define curves over  $\mathbb{F}_p$ . Still have curve equation  $y^2 = x^3 + ax + b$ , with  $a, b \in \mathbb{F}_p$ ,  $4a^3 + 27b^2 \neq 0$ .
- This is called Weierstrass form. Every elliptic curve can be written in this form.
- Can write down group law in terms of  $(x, y)$  coordinates but it's long and has multiple cases.
- Identity element: Special point  $\mathcal{O}$  is "point at infinity".
- Group order: Hasse:  $|E(\mathbb{F}_{p^e})| = p^e + 1 - t$  for some  $-2\sqrt{p^e} \leq t \leq 2\sqrt{p^e}$ .

Some curves admit nicer representations with computational benefits:

- Montgomery form speeds up addition.
- Edwards curves have a simpler addition law that also allows faster operations.
- A curve  $E/\mathbb{F}_p$  in Edwards form can be written  $x^2 + y^2 = 1 + dx^2y^2$ ,  $d \in \mathbb{F}_p$ ,  $d \notin \{0, 1\}$ .
- Edwards point addition:  $P_1 = (x_1, y_1)$ ,  $P_2 = (x_2, y_2)$

$$P_1 + P_2 = \left( \frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

# Elliptic curve scalar multiplication and discrete log

We have a group law  $P + Q = R$ , usually written as “addition”.

## Scalar multiplication of points

- If we iterate the group law  $a$  times on a point  $P$ , we get a point  $aP$ .
- Input:  $a \in \mathbb{Z}$ ,  $P \in E(\mathbb{F}_p)$ , Output:  $aP \in E(\mathbb{F}_p)$
- This can be implemented efficiently with double-and-add, analogous to square-and-multiply that we saw before.

# Elliptic curve scalar multiplication and discrete log

We have a group law  $P + Q = R$ , usually written as “addition”.

## Scalar multiplication of points

- If we iterate the group law  $a$  times on a point  $P$ , we get a point  $aP$ .
- Input:  $a \in \mathbb{Z}$ ,  $P \in E(\mathbb{F}_p)$ , Output:  $aP \in E(\mathbb{F}_p)$
- This can be implemented efficiently with double-and-add, analogous to square-and-multiply that we saw before.

## Elliptic curve discrete log

- Input: base point  $P$ , target  $Q \in E(\mathbb{F}_p)$ ; output  $a$  s.t.  $aP = Q$ .
- For some families of curves, best algorithms known are generic group algorithms: Pollard rho, baby step giant step, take  $O(\sqrt{q})$  time for group order  $q$ .
- Broken in polynomial time by a quantum computer.

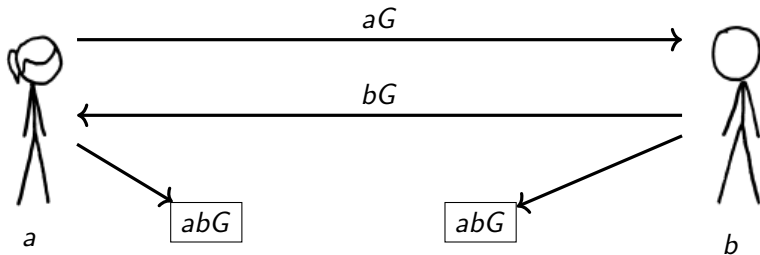
# Elliptic Curve Diffie-Hellman

## Public Parameters

$E$  an elliptic curve over  $\mathbb{F}_p$

$G$  group generator of order  $n$

## Key Exchange



# ECDSA (Digital Signature Algorithm)

## Public Key

$E$  an elliptic curve over  $\mathbb{F}_p$

$G$  group generator (base point)  
on  $E$  of order  $n$

$$Q = dG$$

## Private Key

$d$  private key

## Verify

$$u_1 = H(m)s^{-1} \bmod n$$

$$u_2 = rs^{-1} \bmod n$$

$$r \stackrel{?}{=} \text{x-coord of } u_1G + u_2Q$$

## Sign

Generate random  $k$ .

$r = \text{x-coordinate of } kG$ .

$$s = k^{-1}(H(m) + dr) \bmod n$$

Output  $(r, s)$

# Standardized Elliptic Curves

Curves that withstand all known attacks are more complex to generate than primes, so everyone uses a small number of curves.

## NIST P256

- Works over  $\mathbb{F}_p$  with  $p \approx 2^{256}$ .
- Has prime order  $q \approx 2^{256}$ .
- Best attack:  $\sqrt{q}$  time  $\approx 2^{128}$ .
- Curve parameters generated from deterministic algorithm by opaque seed of unknown generation.

# Standardized Elliptic Curves

Curves that withstand all known attacks are more complex to generate than primes, so everyone uses a small number of curves.

## NIST P256

- Works over  $\mathbb{F}_p$  with  $p \approx 2^{256}$ .
- Has prime order  $q \approx 2^{256}$ .
- Best attack:  $\sqrt{q}$  time  $\approx 2^{128}$ .
- Curve parameters generated from deterministic algorithm by opaque seed of unknown generation.

## Curve25519

- Edwards curve designed by Daniel J. Bernstein
- Designed to make implementations “secure by default”:
  - Simplified group law easier to protect against side-channel attacks
  - Minimal/no point validation required
  - “Twist-secure”: Also minimize validation necessary for implementations that only input x-coordinate.

# Elliptic curve factoring method

Elliptic curves are useful for factoring too!

Recall  $p - 1$  method to factor an integer  $N$ :

1. Generate composite  $M = p_1^{e_1} \dots p_k^{e_k}$
2. Check if  $\gcd(a^M - 1 \bmod N, N) \neq 1$ .

Fermat's little theorem:  $a^M \equiv 1 \pmod{p}$  if  $p - 1 | M$

Finds factors where  $p - 1 | M$ , so only works if  $p - 1$  has only small factors.

# Elliptic curve factoring method

Lenstra

1. Generate highly composite  $M = p_1^{e_1} \dots p_k^{e_k}$ .
2. Choose curve  $E$  and point  $P$  on  $E$ , working modulo  $N$ .
3.  $Q = MP$ .
4. If  $\gcd(x(Q), N)$  factors  $N$ , done, otherwise repeat for different curves.

Factors  $N$  if order of  $P$  in  $E(\mathbb{F}_p)$  divides  $M$ , for factor  $p$  of  $N$ .

# Elliptic curve factoring method

Lenstra

1. Generate highly composite  $M = p_1^{e_1} \dots p_k^{e_k}$ .
2. Choose curve  $E$  and point  $P$  on  $E$ , working modulo  $N$ .
3.  $Q = MP$ .
4. If  $\gcd(x(Q), N)$  factors  $N$ , done, otherwise repeat for different curves.

Factors  $N$  if order of  $P$  in  $E(\mathbb{F}_p)$  divides  $M$ , for factor  $p$  of  $N$ .

## Theorem

Hasse  $p + 1 - 2\sqrt{p} < |E(\mathbb{F}_p)| < p + 1 + 2\sqrt{p}$

In Pollard  $p - 1$  method, only one group  $\mathbb{Z}_p^*$  of order  $p - 1$ .

For ECM, can keep choosing random curves with random orders until  $|E(\mathbb{F}_p)|$  has small factors.

Running time:  $O(e^{\sqrt{2+o(1)}\sqrt{\ln p \ln \ln p}})$ . Feasible for 80-bit  $p$ .

## Extra properties of elliptic curves: Pairings

Some elliptic curve groups have efficiently computable pairings.

We switch to representing the group operation with  $\times$  because that is the convention for this application.

A pairing is a map  $e : G_0 \times G_1 \rightarrow G_T$  that satisfies:

- Bilinear:  $e(u \cdot u', v) = e(u, v) \cdot e(u', v)$  and  $e(u, v \cdot v') = e(u, v) \cdot e(u, v')$
- Non-degenerate:  $e(g_0, g_1) = g_T$  where  $g_0, g_1, g_T$  are generators of  $G_0, G_1, G_T$ .
- $G_0$  is an order  $q$  subgroup of  $E(\mathbb{F}_p)$ .
- $G_1$  is an order  $q$  subgroup of  $E(\mathbb{F}_{p^d})$ .
- $G_T$  is an order  $q$  multiplicative subgroup of  $\mathbb{F}_{p^d}$ .

# Neat pairing facts

## DDH is false

Let  $(u, v, w)$  be our DDH problem:  $(g_0^a, g_0^b, g_0^c)$  or  $(g_0^a, g_0^b, g_0^{ab})$

If  $e(u, v) = e(g_0, w)$  then

$$e(g_0^a, g_0^b) = e(g_0, g_0)^{ab} \stackrel{?}{=} e(g_0, g_0^c) = e(g_0, g_0)^c.$$

## Discrete log in $G_0$ or $G_1$ is no harder than in $G_T$

1. Input  $u_0 = g_0^a \in G_0$ .
2. Compute  $u = e(u_0, g_1)$ ,  $g_T = e(g_0, g_1)$ , so  $u = g_T^a$ .
3. Compute discrete log in  $G_T$ , get  $a$ .

## Application: 3-way Diffie-Hellman

We know how 2-way Diffie-Hellman works:

Alice and Bob exchange messages  $g^a$ ,  $g^b$  and compute  $g^{ab}$ .

What about a 3-way exchange?

## Application: 3-way Diffie-Hellman

We know how 2-way Diffie-Hellman works:

Alice and Bob exchange messages  $g^a$ ,  $g^b$  and compute  $g^{ab}$ .

What about a 3-way exchange?

- Alice, Bob, Charlie exchange  $g^a$ ,  $g^b$ ,  $g^c$ .
- They want to compute  $g^{abc}$
- Discrete log should remain hard.

If they use pairings:

- Alice computes  $e(g^b, g^c)^a = g_T^{bca} = g_T^{abc}$
- Bob computes  $e(g^a, g^c)^b = g_T^{acb} = g_T^{abc}$
- Charlie computes  $e(g^a, g^b)^c = g_T^{abc}$

## Application: 3-way Diffie-Hellman

We know how 2-way Diffie-Hellman works:

Alice and Bob exchange messages  $g^a, g^b$  and compute  $g^{ab}$ .

What about a 3-way exchange?

- Alice, Bob, Charlie exchange  $g^a, g^b, g^c$ .
- They want to compute  $g^{abc}$
- Discrete log should remain hard.

If they use pairings:

- Alice computes  $e(g^b, g^c)^a = g_T^{bca} = g_T^{abc}$
- Bob computes  $e(g^a, g^c)^b = g_T^{acb} = g_T^{abc}$
- Charlie computes  $e(g^a, g^b)^c = g_T^{abc}$

Unsolved: multilinear map/group key exchange for  $m$  parties.

# Elliptic curves and quantum computers

NSA has been strongly pro elliptic curve cryptography for decades.

In 2015, NSA recommended against transitioning to elliptic curves for people who hadn't already.

Official explanation: Quantum computers are coming, so people should wait for post-quantum crypto.

Plausible explanation:

- Smaller key sizes for ECC mean fewer qubits required to break than 3072-bit factoring.

# Dual EC DRBG, this time with elliptic curves

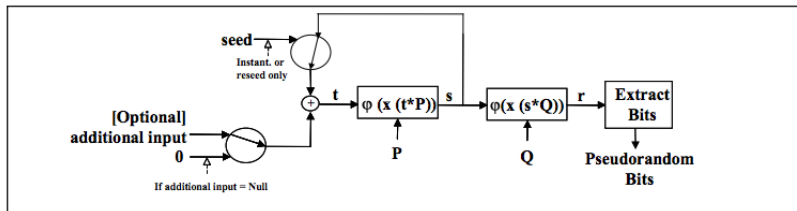


Figure 13: Dual\_EC\_DRBG

- Parameters: Pre-specified elliptic curve points  $P$  and  $Q$ .
- Seed: 32-byte integer  $s$
- State:  $x$ -coordinate of point  $sP$ . ( $\phi(x(sP))$  above.)
- Update:  $t = s \oplus$  optional additional input. State  $s = x(tP)$ .
- Output: At state  $s$ , compute  $x$ -coordinate of point  $x(sQ)$ , discard top 2 bytes, output 30 bytes.

# Shumow and Ferguson 2007, redux

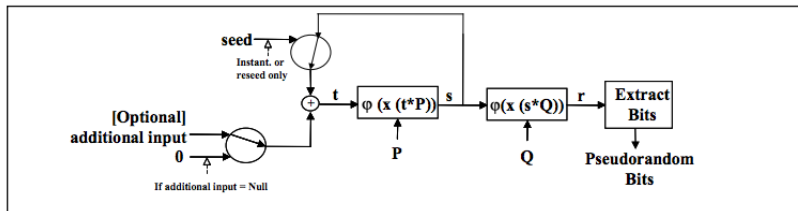


Figure 13: Dual\_EC\_DRBG

1. Assume attacker controls standard and constructs points with known relationship  $P = dQ$ .
2. Attacker gets 30 bytes of  $x$ -coordinate of  $sQ$ . Attacker brute forces  $2^{16}$  MSBs, gets  $2^{17}$  possible  $y$ -coordinates, ends up with  $2^{15}$  candidates for  $sQ$ .
3. For each candidate  $sQ$  attacker computes  $dsQ = sP$  and compares to next output.

# Reminders

1. HW 6 is due next lecture.
2. HW 7 will be online shortly.