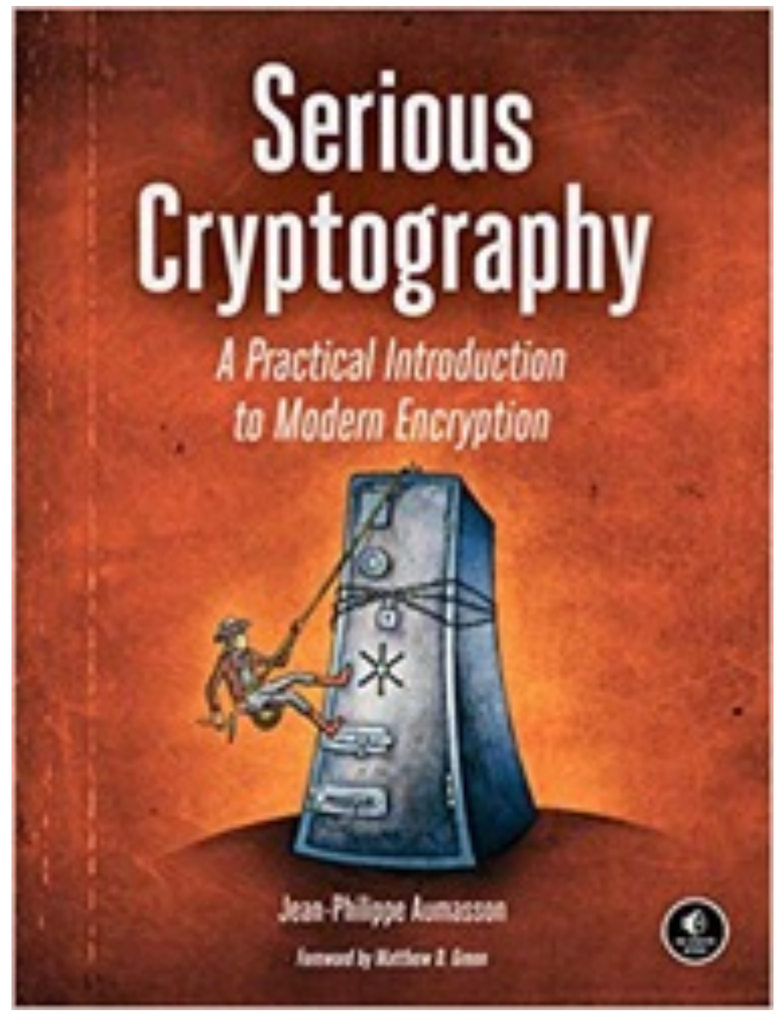
CNIT 141
Cryptography for Computer Networks



12. Elliptic Curves

Updated 11-30-22

Topics

- What is an Elliptic Curve?
- The ECDLP Problem
- Diffie-Hellman Key Agreement over Elliptic Curves
- Choosing a Curve
- How Things Can Go Wrong

New & Improved

- ECC was introduced in 1985
- More powerful and efficient than RSA and classical Diffie-Hellman
- ECC with as 256-bit key is stronger than RSA with a 4096-bit key

Slow Adoption

- OpenSSL added ECC in 2005
- OpenSSH added it in 2011
- Used in Bitcoin
- Most applications based on DLP can use ECC instead
  - Except Secure Remote Password
    - Link Ch 12a

What is an Elliptic Curve?

Elliptic Curve over Real Numbers

$$y^2 = x^3 + ax + b$$

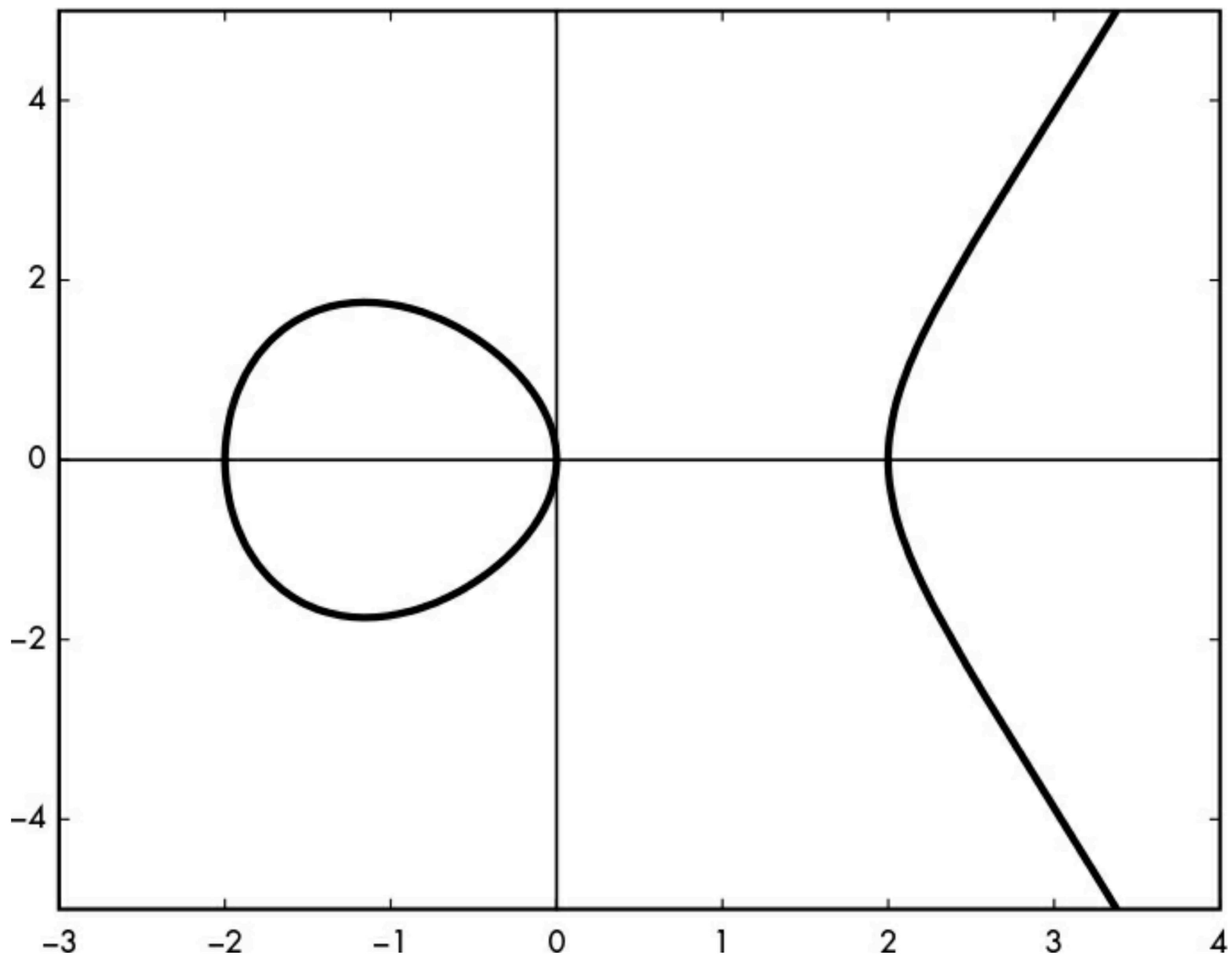


Figure 12-1: An elliptic curve with the equation $y^2 = x^3 - 4x$, shown over the real numbers

Elliptic Curves over Integers

- Mod 191
- From group $\boldsymbol{Z}_{191}$ = 0, 1, 2, ... 190

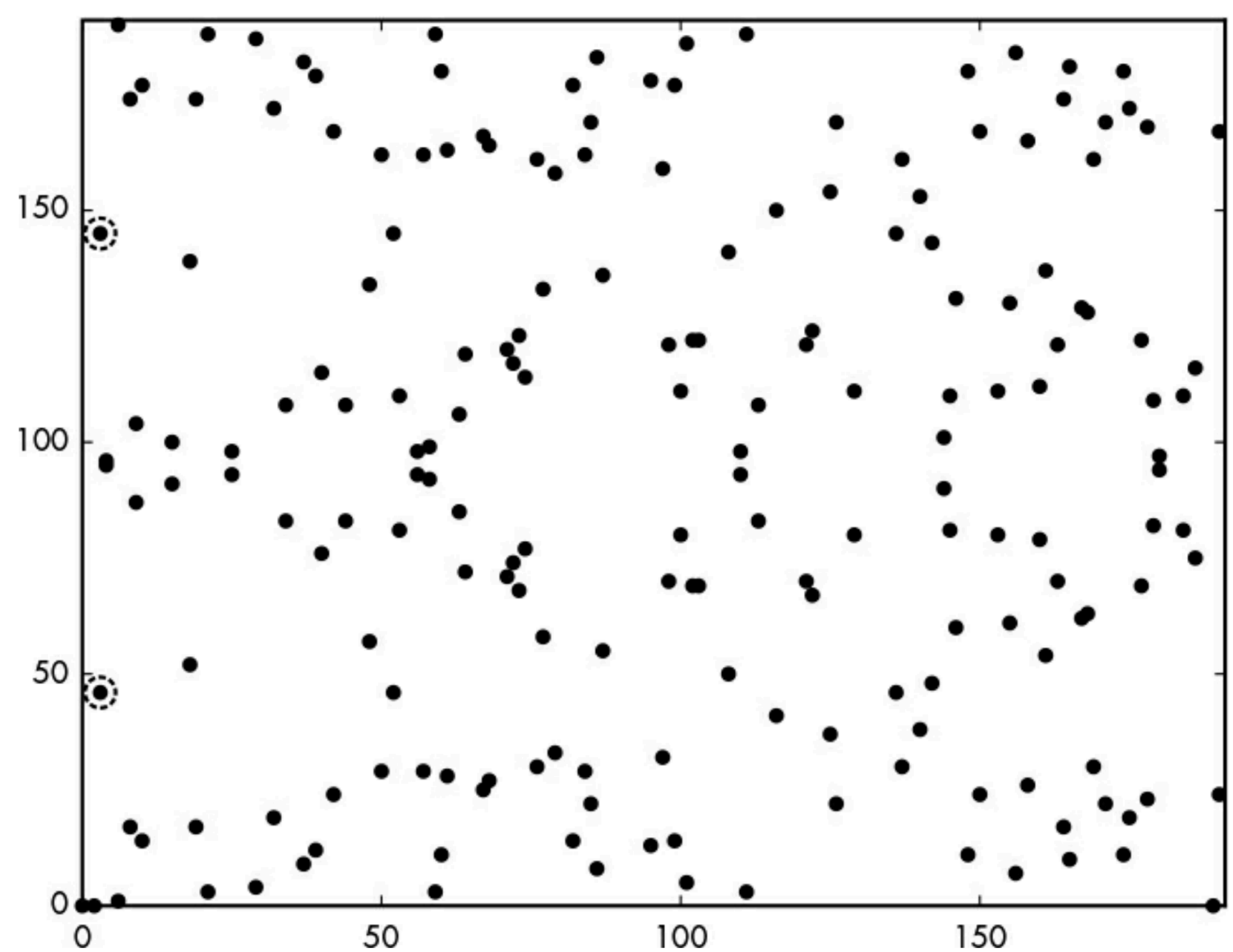


Figure 12-2: The elliptic curve with the equation $y^2 = x^3 - 4x$ *over* $\boldsymbol{Z}_{191}$*, the set of integers modulo 191*

The Field $\boldsymbol{Z_p}$

- We'll use both addition and multiplication
- We need 0 as the additive identity element
  - x + 0 = x
- There are inverses for addition (-x)
  - And for multiplication (denoted 1/x)
- Such a group is called a ***field***
- A finite number of elements: ***finite field***

Adding Points

- ***P*** + ***Q***: Draw line connecting P and Q
  - Find the point where it intersects with the elliptic curve
  - Reflect around X-axis

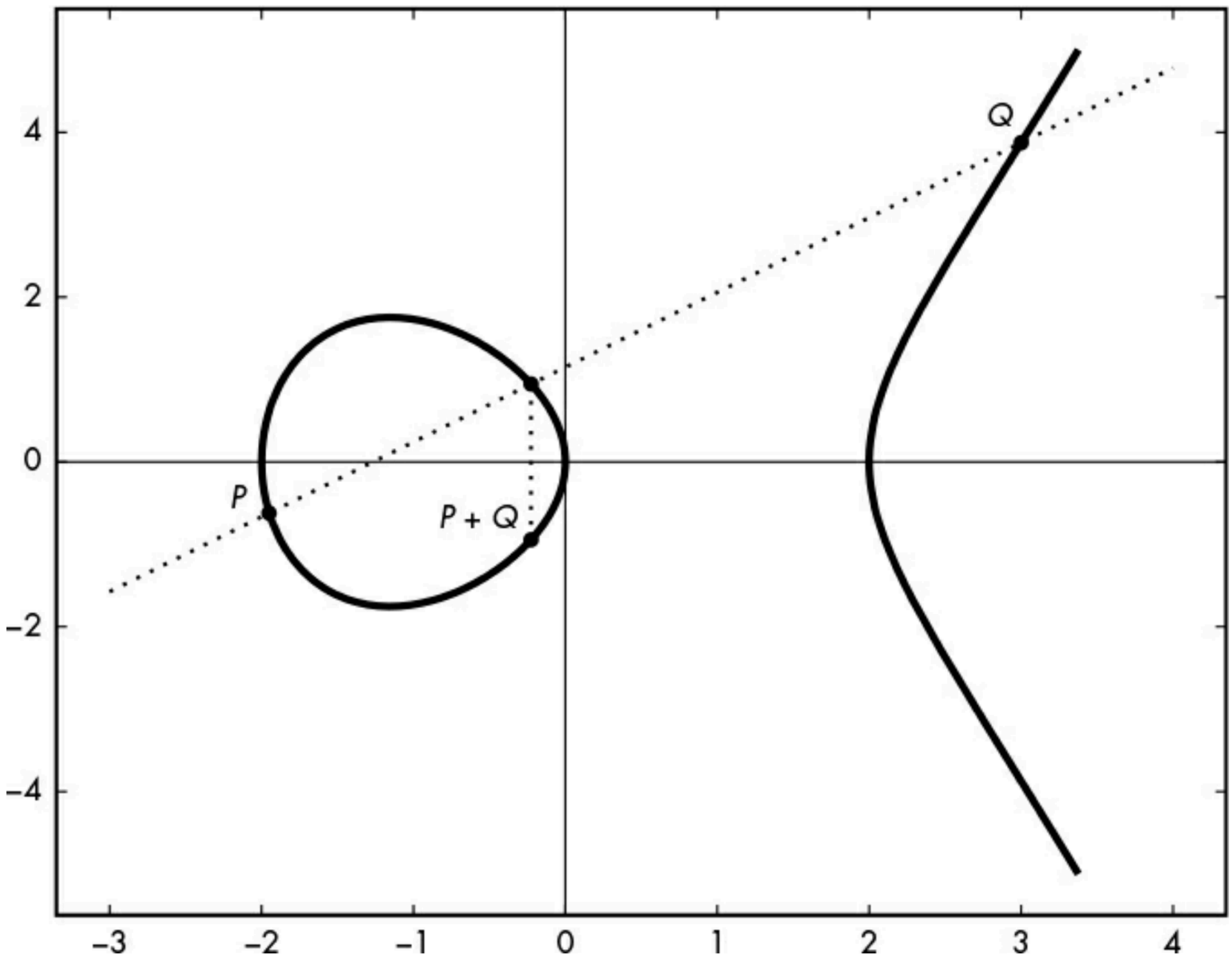


Figure 12-3: A general case of the geometric rule for adding points over an elliptic curve

P + (-P)

$\boldsymbol{P} = (\boldsymbol{x_P}, \boldsymbol{y_P})$

$\boldsymbol{-P} = (\boldsymbol{x_P}, -\boldsymbol{y_P})$

- Adding these points makes a vertical line
- Goes to the "point at infinity"
  - Which acts as zero for elliptic curves

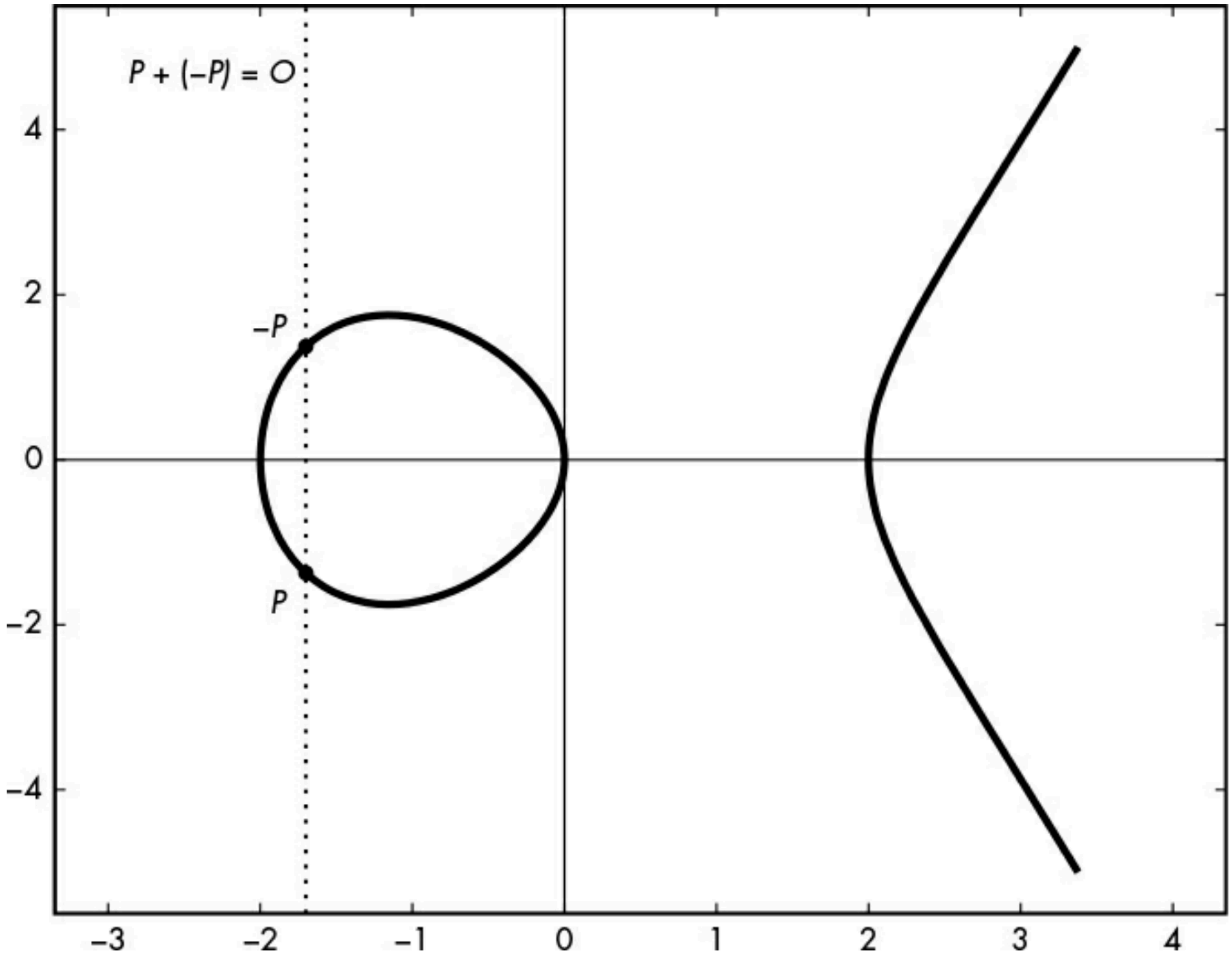
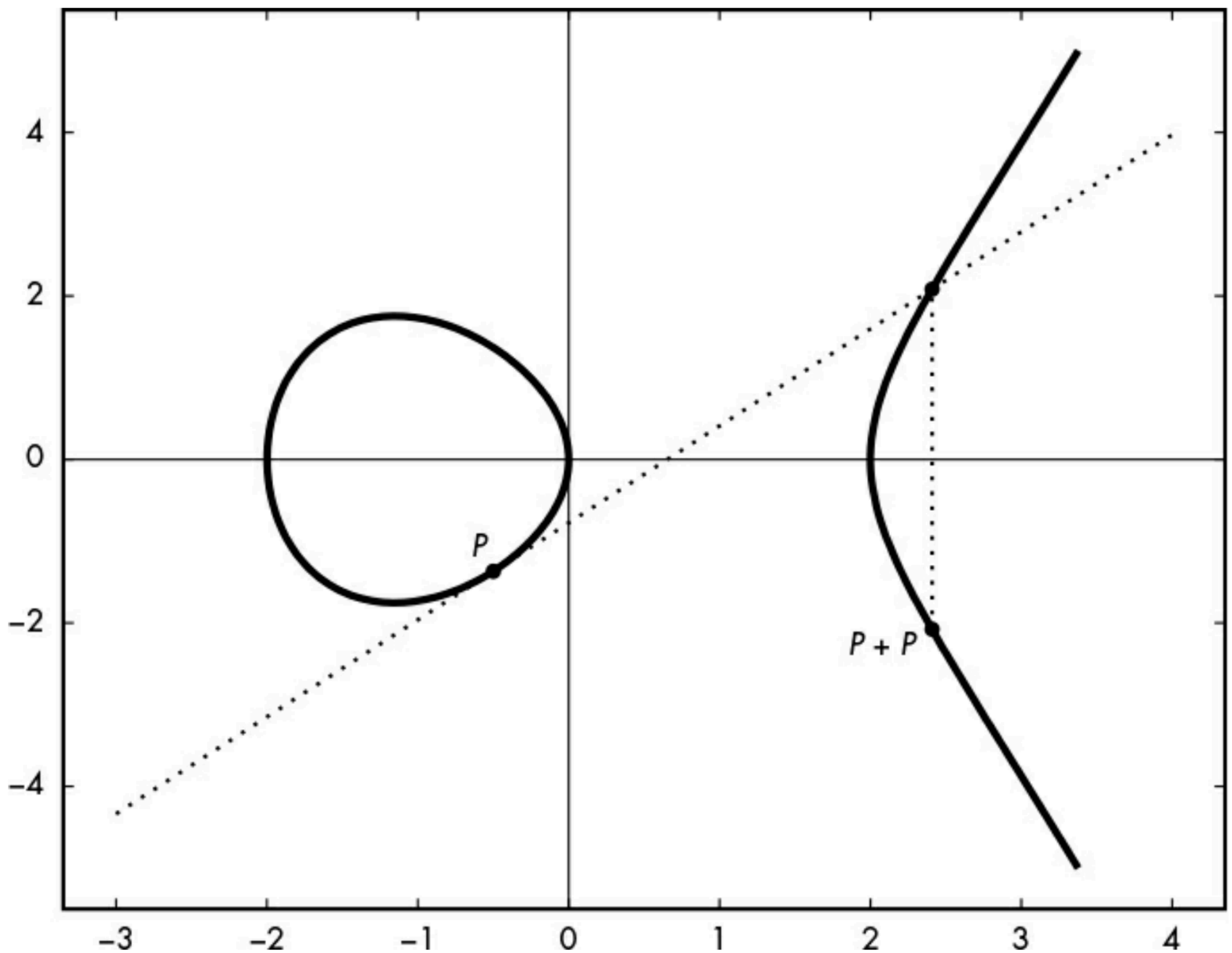


Figure 12-4: The geometric rule for adding points on an elliptic curve with the operation P + (–P) = O *when the line between the points never intersects the curve*

P + P

- Use tangent line



Multiplication

- $\boldsymbol{kP} = \boldsymbol{P} + \boldsymbol{P} + \boldsymbol{P} + \ldots$ ($\boldsymbol{k}$ times)
- To calculate it faster, calculate:
  - $\boldsymbol{P_2} = \boldsymbol{P} + \boldsymbol{P}$
  - $\boldsymbol{P_4} = \boldsymbol{P_2} + \boldsymbol{P_2}$
  - $\boldsymbol{P_8} = \boldsymbol{P_4} + \boldsymbol{P_4}$
  - etc.

What is a Group?

- A set of elements (denoted $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ below)
- An operation (denoted $\times$ below)
- With these properties
  - **Closure**
    - If $\boldsymbol{x}$ and $\boldsymbol{y}$ are in the group, $\boldsymbol{x} \times \boldsymbol{y}$ is too
  - **Associativity** $(\boldsymbol{x} \times \boldsymbol{y}) \times \boldsymbol{z} = \boldsymbol{x} \times (\boldsymbol{y} \times \boldsymbol{z})$
  - **Identity** element **e** such that $\boldsymbol{e} \times \boldsymbol{x} = \boldsymbol{x}$
  - **Inverse** For every $\boldsymbol{x}$, there is a $\boldsymbol{y}$ with $\boldsymbol{x} \times \boldsymbol{y} = \boldsymbol{e}$

Elliptic Curve Groups

- Points ***P***, ***Q***, ***R*** and "addition" form a group
  - **Closure**
    - If ***P*** and ***Q*** are in the group, ***P*** + ***Q*** is too
  - **Associativity** (***P*** + ***Q***) + ***R*** = ***P*** + (***Q*** + ***R***)
  - **Identity** element **O** is the point at infinity
    - Such that ***P*** + ***O*** = ***P***
  - **Inverse**
    - For every ***P*** = (***x***, ***y***) , ***-P*** = (***x***, -***y***)
    - ***P*** + (-***P***) = ***O***

The ECDLP Problem

ECDLP

- All elliptic curve cryptography is based in this problem

  Given $\boldsymbol{P}$ and $\boldsymbol{Q}$

  Find $\boldsymbol{k}$ such that $\boldsymbol{Q} = \boldsymbol{kP}$

- Believed to be hard, like Discrete Logratim Problem
- Has withstood cryptanalysis since its introduction in 1985

Smaller Numbers

- Using the field $\mathbf{Z}_p$
- Where p is n bits long
- The security is n / 2 bits
- A p 256 bits long provides 128 bits of security
- That would take more than 4096 bits with RSA
- ECC is much faster

Diffie-Hellman Key Agreement over Elliptic Curves

RSA-Based Diffie-Hellman (DH)

- They can both calculate g^{ab} by combining public and secret information

Keep a secret
Transmit g^a
Calculate $g^{ab} = B^a$

Keep b secret
Transmit g^b
Calculate $g^{ab} = A^b$

ECDH

- Choose a fixed point $\boldsymbol{G}$ (not secret)
- Shared secret is $\boldsymbol{d_A P_B} = \boldsymbol{d_B P_A} = \boldsymbol{d_A d_B G}$

Pick a random secret $\boldsymbol{d_A}$
Transmit $\boldsymbol{P_A} = \boldsymbol{d_A G}$
Calculate $\boldsymbol{d_A d_B G} = \boldsymbol{d_A P_B}$

Pick a random secret $\boldsymbol{d_B}$
Transmit $\boldsymbol{P_B} = \boldsymbol{d_B G}$
Calculate $\boldsymbol{d_A d_B G} = \boldsymbol{d_B P_A}$

Signing with Elliptic Curves

- ECDSA (Elliptic Curve Digital Signature Algorithm)
  - Replaces RSA and classical DSA
  - The only signature used in Bitcoin
  - Supported by many TLS and SSH implementations
- Consists of ***signature generation*** and ***verification*** algorithms

Signing with Elliptic Curves

- Signer holds a private key ***d***
- Verifiers hold the public key ***P*** = ***dG***
- Both know:
  - What elliptic curve to use
  - Its order ***n*** (the number of points in the curve)
  - Coordinates of a base point ***G***

ECDSA Signature Generation

- Signer hashes the message to form ***h***
  - With a function such as SHA-256 or BLAKE2
- Signer picks a random number ***k***
  - Calculates ***kG***, with coordinates (***x***, ***y***)
- Signature is (***r***, ***s***):

$$\boldsymbol{r = x \bmod n} \qquad \boldsymbol{s = (h + rd) / k \bmod n}$$

Signature Length

- If coordinates are 256-bit numbers
  - ***r*** and ***s*** are both 256 bits long
  - Signature is 512 bits long

ECDSA vs. RSA Signatures

- RSA is used only for encryption and signatures
- ECC is a family of algorithms
  - Encryption and signatures
  - Key agreement
  - Advanced functions such as identity-based encryption

ECDSA vs. RSA Signatures

- RSA's signature and verification algorithms are simpler than ECDSA
  - RSA's verification process is often faster because of the small public key ***e***
- ECC has two major advantages
  - Shorter signatures
  - Faster signing speed

Speed Comparison

- ECDSA is 150x faster at signing than RSA
- Slightly faster at verifying

```
$ openssl speed ecdsap256 rsa4096
                              sign      verify      sign/s    verify/s
rsa 4096 bits           0.007267s   0.000116s      137.6      8648.0
                              sign      verify      sign/s    verify/s
256 bit ecdsa (nistp256)   0.0000s     0.0001s    21074.6      9675.7
```

Encrypting with Elliptic Curves

- Rarely used
  - Message size can't exceed about 100 bits
- RSA can use 4000 bits at same security level

Integrated Encryption Scheme (IES)

- Generate a Diffie-Hellman shared secret
  - Derive a symmetric key from the shared secret
  - Use symmetric encryption

Elliptic Curve Integrated Encryption Scheme (ECIES)

1. **Pick a random number, d, and compute the point $Q = dG$, where the base point G is a fixed parameter. Here, (d, Q) acts as an ephemeral key pair, used only for encrypting M.**
2. **Compute an ECDH shared secret by computing $S = dP$.**
3. **Use a key derivation scheme (KDF) to derive a symmetric key, K, from S.**
4. **Encrypt M using K and a symmetric authenticated cipher, obtaining a ciphertext, C, and an authentication tag, T.**

Choosing a Curve

Parameters

- **Order** (number of points), also called **modulus**
- ***a*** and ***b*** in

$$y^2 = x^3 + ax + b$$

- Origin of the chosen ***a*** and ***b***

Criteria for Security

- **Order** (number of points) must not be product of small numbers
- Curves that treat ***P***+***Q*** and ***P***+***P*** the same way are safer, to avoid information leaks
  - **Unified addition law**
- Creators of curve should explain how ***a*** and ***b*** were chosen, or people will be suspicious

NIST Curves

- Standardized in 2000
- Five ***prime curves*** (modulus is prime)
  - The most commonly used ones
- Ten others use "binary polynomials"
  - Make hardware implementation more efficient
  - Rarely used with elliptic curves

P-256

- The most common NIST curve
- Modulus is ***p***
- ***b*** is a 256-bit number

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

$$y^2 = x^3 - 3x + b$$

Where Did b Come From?

- NSA never explained it well
- Most experts don't believe there's a backdoor
- ***b*** is the SHA-1 hash of this random-looking value

```
c49d3608 86e70493 6a6678e1 139d26b7 819f7e90
```

But few people trust the NIST curves because of Snowden. Bruce Schneier wrote:

> I no longer trust the constants. I believe the NSA has manipulated them through their relationships with industry.

and that:

> I looked at the random seed values for the P-xxxr curves. For example, P-256r's seed is c49d360886e704936a6678e1139d26b7819f7e90. No justification is given for that value.

and:

> I now personally consider this to be smoking evidence that the parameters are cooked.

And so, Bitcoin, Tor and so many applications avoid the NIST derived curves, and instead focus on **secp256k1** and **Curve 25519**. Bitcoin uses secp256k1 which has a prime of 2^{256}-$2^{32}-977$, and Tor uses Curve 25519 which has prime of 2^{255}-19.

- https://medium.com/asecuritysite-when-bob-met-alice/ding-ding-its-secp256k1-and-curve-25519-in-red-corner-and-nist-in-the-blue-corner-f49c4ad1c3a8

Curve 25519

- Published by Daniel J. Bernstein in 2006
- Faster than NIST standard curves
- Uses shorter keys
- No suspicious constants
- Uses same formula for ***P***+***Q*** and ***P***+***P***

Curve 25519

$$y^2 = x^3 + 486662x^2 + x$$

$$p = 2^{255} - 19$$

- Used everywhere
  - Chrome, Apple, OpenSSH
- But not a NIST standard

Curve secp256k1

$$y^2 = x^3 + 7$$

$$p = 2^{256} - 2^{32} - 977$$

- Used in Bitcoin

https://en.bitcoin.it/wiki/Secp256k1

Other Curves

- Old national standards
  - France: ANSSI curves
  - Germany: Brainpool curves
    - Use constants of unknown origin

Other Curves

- Newer curves, rarely used
  - **Curve41417**
    - More secure variant of Curve25519
  - **Ed448-Goldilocks**
    - 448-bit curve from 2014

How Things Can Go Wrong

Large Attack Surface

- Elliptic curves have
  - More parameters than classic Diffie-Hellman
  - More opportunities for mistakes
  - Possible vulnerabilities to side-channel attacks
    - Timing of calculations on large numbers

ECDSA with Bad Randomness

- Signing uses a secret random ***k***

  $\boldsymbol{s} = (\boldsymbol{h} + \boldsymbol{rd}) / \boldsymbol{k} \bmod \boldsymbol{n}$

- If ***k*** is re-used, attacker can calculate ***k***

  $\boldsymbol{k} = (\boldsymbol{h_1} - \boldsymbol{h_2}) / (\boldsymbol{s_1} - \boldsymbol{s_2})$

- This happened on the PlayStation 3 in 2010
  - Presented at CCC by failOverflow team

Invalid Curve Attack

- Attacker sends $\boldsymbol{P_A}$ that is not on the same curve
  - From a weak curve instead
- Target fails to verify that $\boldsymbol{P_A}$ is on the curve, and uses a secret $\boldsymbol{d_B P_A}$ on a weak curve, so ECDLP can be solved

Doesn't know $\boldsymbol{d_A}$
Transmit malicious $\boldsymbol{P}$
Calculates secret $\boldsymbol{d_B}$

Pick a random secret $\boldsymbol{d_B}$
Transmit $\boldsymbol{P_B = d_B G}$
Calculate $\boldsymbol{d_B P}$

Invalid Curve Attack

- Malicious client could trick server into using the wrong curve
- Exposing the server's secret key
- Some TLS implementations were shown to be vulnerable in 2015
  - https://threatpost.com/json-libraries-patched-against-invalid-curve-crypto-attack/124336/

JSON Libraries Patched Against Invalid Curve Crypto Attack

Kahoot!