CS 6530 Applied Cryptography

July-Nov 2025

Elliptic Curve Cryptography

25th August 2025_S2, 26th August 2025

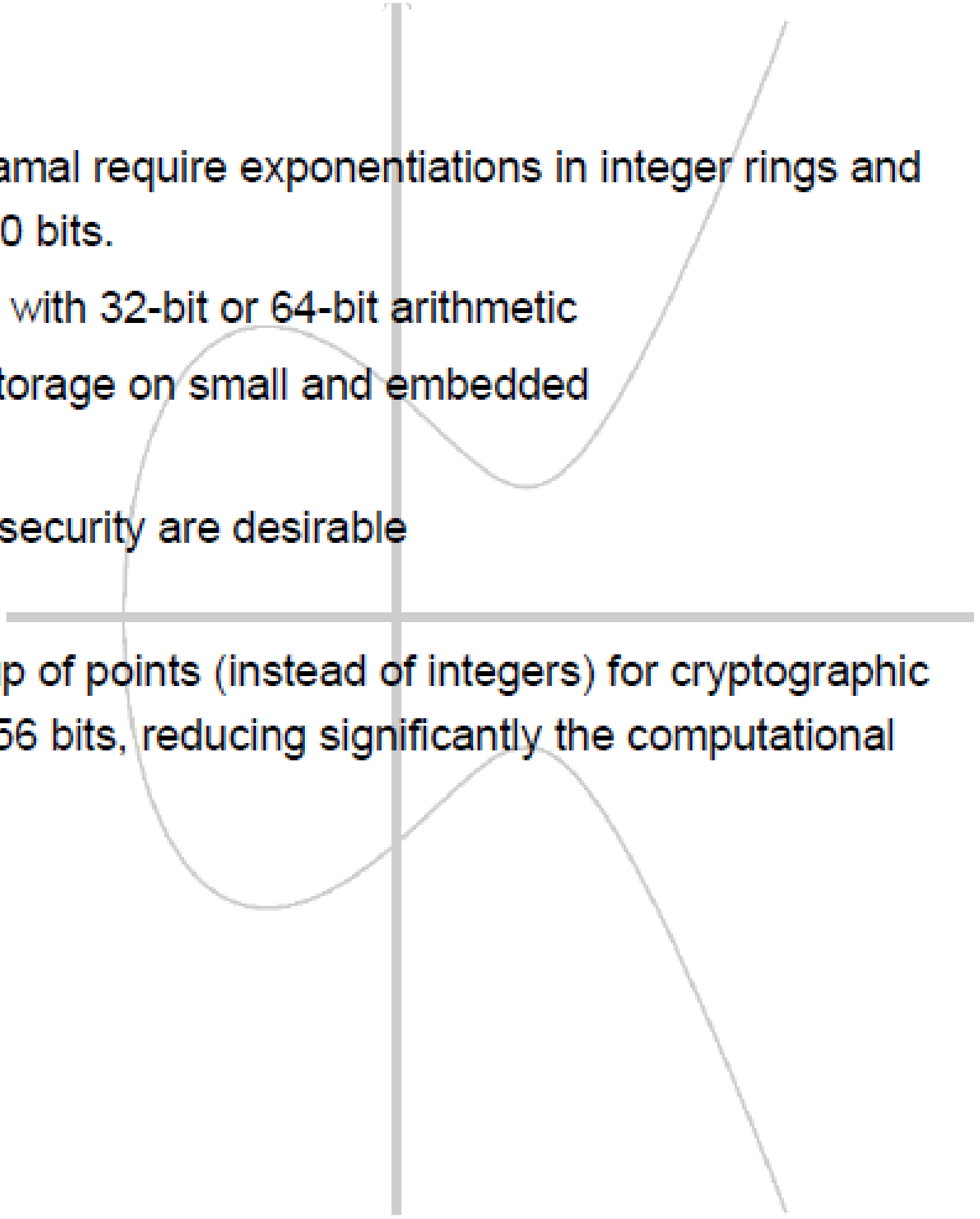
Dr. Manikantan Srinivasan

(Material covered is based on Chapter 9 of

Understanding Cryptography – Second Edition,

Chapter 12 of Serious Cryptography – Second Edition)

Motivation



- **Problem:**
  Asymmetric schemes like RSA and Elgamal require exponentiations in integer rings and fields with parameters of more than 1000 bits.
  - High computational effort on CPUs with 32-bit or 64-bit arithmetic
  - Large parameter sizes critical for storage on small and embedded
- **Motivation:**
  Smaller field sizes providing equivalent security are desirable
- **Solution:**
  Elliptic Curve Cryptography uses a group of points (instead of integers) for cryptographic schemes with coefficient sizes of 160-256 bits, reducing significantly the computational effort.

Computations on Elliptic Curves

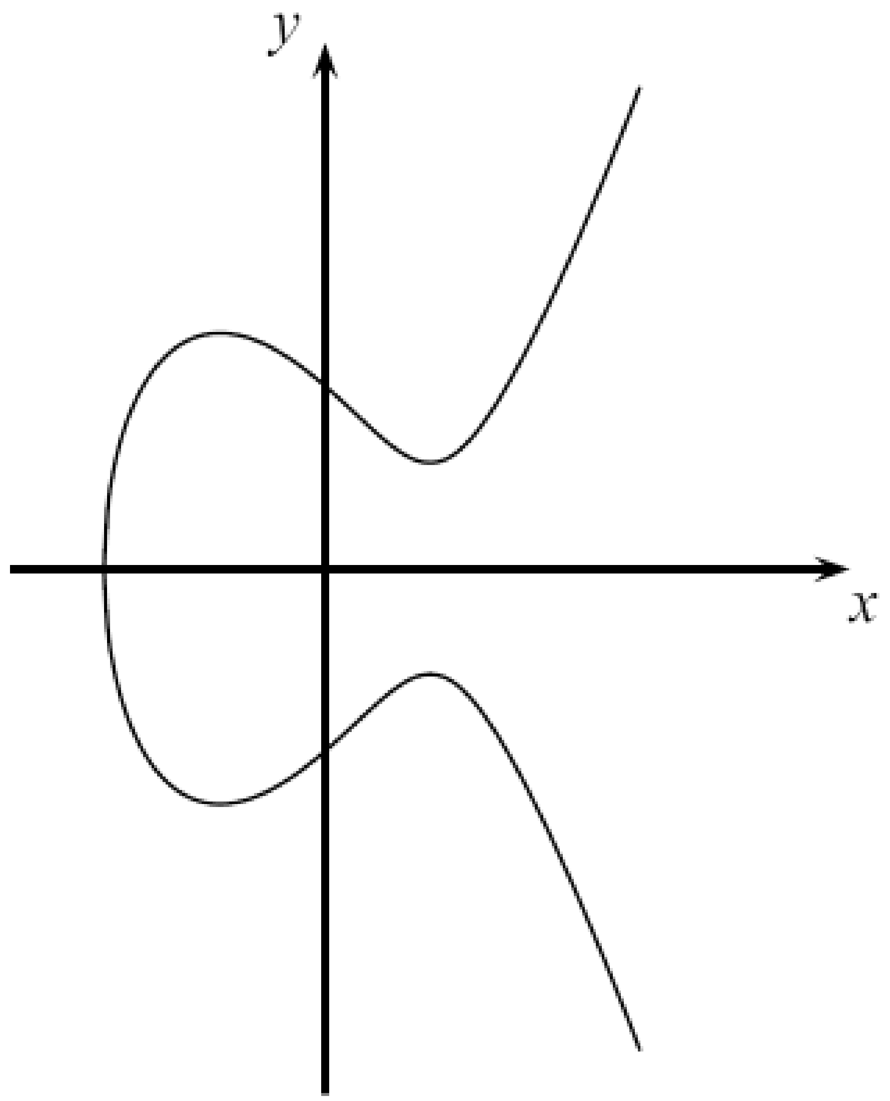
- Elliptic curves are polynomials that define points based on the (simplified) Weierstraß equation:

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

  for parameters a,b that specify the exact shape of the curve

- On the real numbers and with parameters a, b $\in$R, an elliptic curve looks like this →

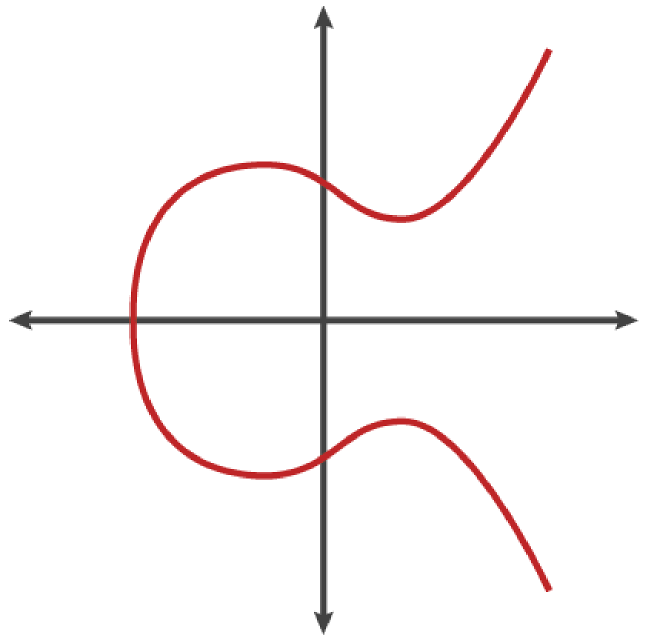
- Elliptic curves can not just be defined over the real numbers *R* but over many other types of finite fields.



Example: $y^2 = x^3 - 3x + 3$ over *R*

Computations on Elliptic Curves

- That graphs to something that looks a bit like this:



There are other representations of elliptic curves, but technically an elliptic curve is the set points satisfying an equation in two variables with degree two in one of the variables and three in the other. An elliptic curve is not just a pretty picture, it also has some properties that make it a good setting for cryptography.

Computations on Elliptic Curves

- In cryptography, we are interested in elliptic curves module a prime *p*:

Definition: Elliptic Curves over prime fields

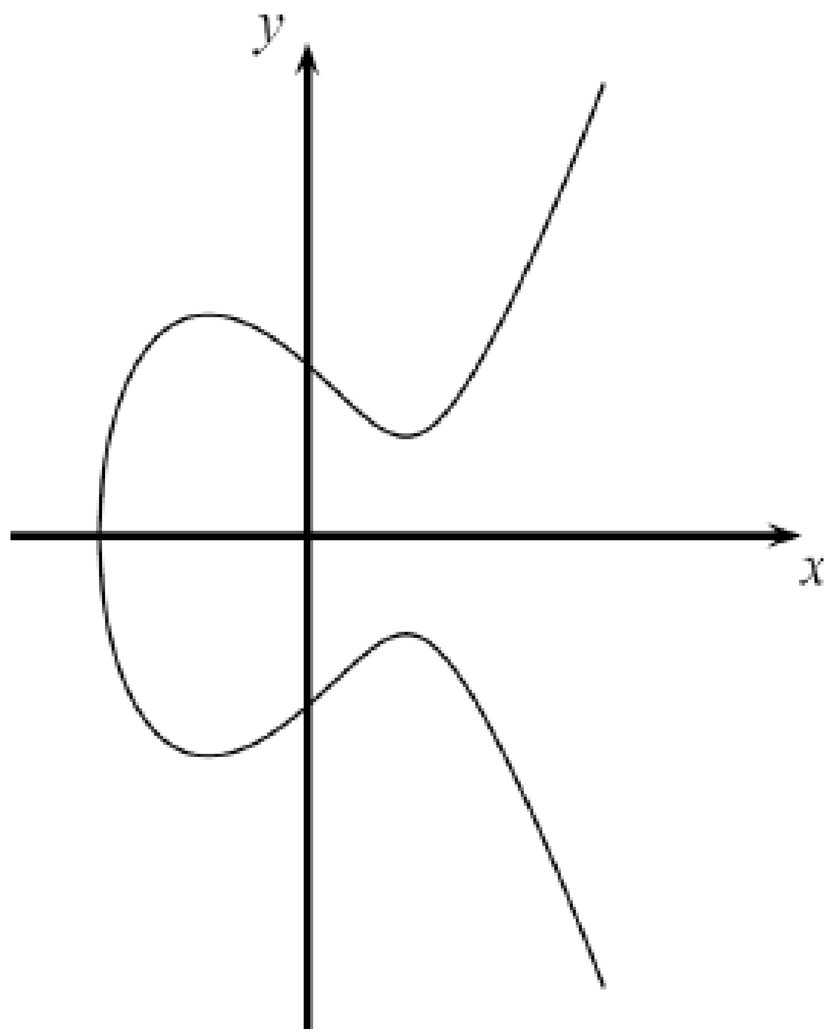
The elliptic curve over Z_p, $p>3$ is the set of all pairs $(x,y) \in Z_p$ which fulfill

$$y^2 = x^3 + ax + b \bmod p$$

together with an imaginary point of infinity θ, where $a,b \in Z_p$ and the condition

$$4a^3+27b^2 \neq 0 \bmod p.$$

$\in$

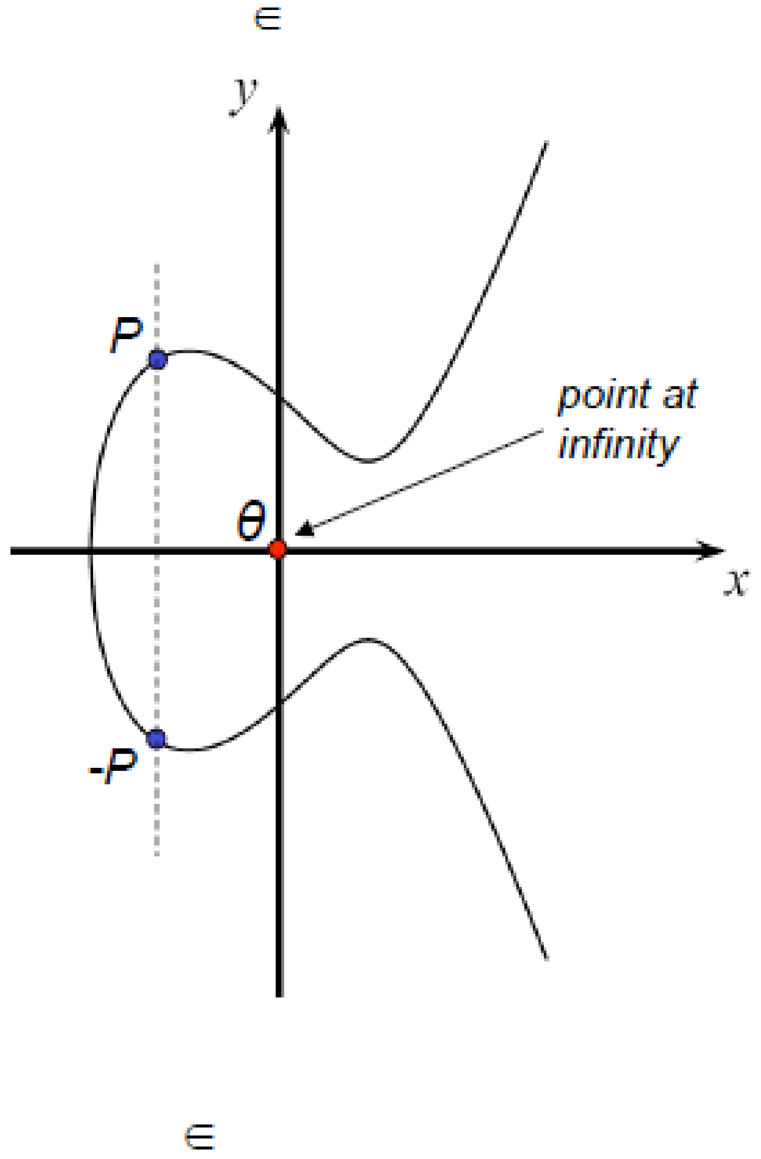


- Note that $Z_p = \{0,1,\ldots, p-1\}$ is a set of integers with modulo p arithmetic

Computations on Elliptic Curves

- Some special considerations are required to convert elliptic curves into a group of points
  - *In any group, a special element is required to allow for the identity operation, i.e., given P∈E: P + θ = P = θ + P*
  - *This identity point (which is not on the curve) is additionally added to the group definition*
  - *This (infinite) identity point is denoted by θ*

- Elliptic Curve are symmetric along the *x*-axis
  - Up to two solutions *y and -y* exist for each quadratic residue *x* of the elliptic curve
  - For each point *P =(x,y)*, the inverse or negative point is defined as *-P =(x,-y)*



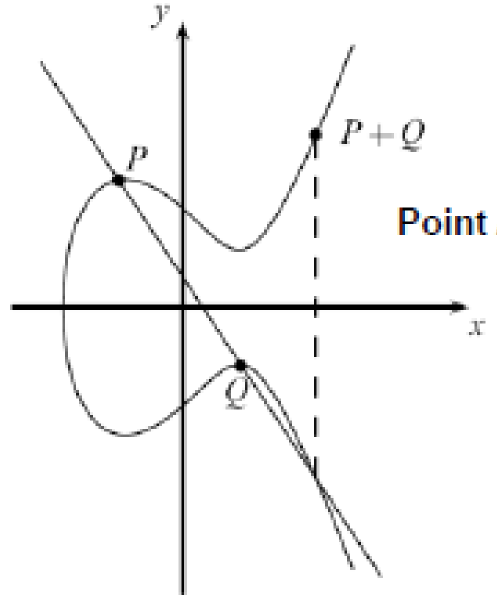
Computations on Elliptic Curves

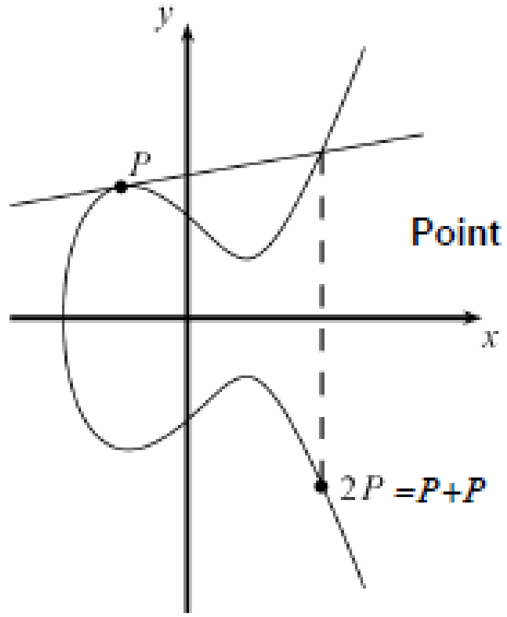
- Generating a *group of points* on elliptic curves based on point addition operation *P+Q = R, i.e.,* $(x_P,y_P)+(x_Q,y_Q) = (x_R,y_R)$
- Geometric Interpretation of point addition operation
  - *Draw straight line through P and Q; if P=Q use tangent line instead*
  - *Mirror third intersection point of drawn line with the elliptic curve along the x-axis*
- Elliptic Curve Point Addition and Doubling Formulas

$$x_3 = s^2 - x_1 - x_2 \bmod p \quad \text{and} \quad y_3 = s(x_1 - x_3) - y_1 \bmod p$$

where

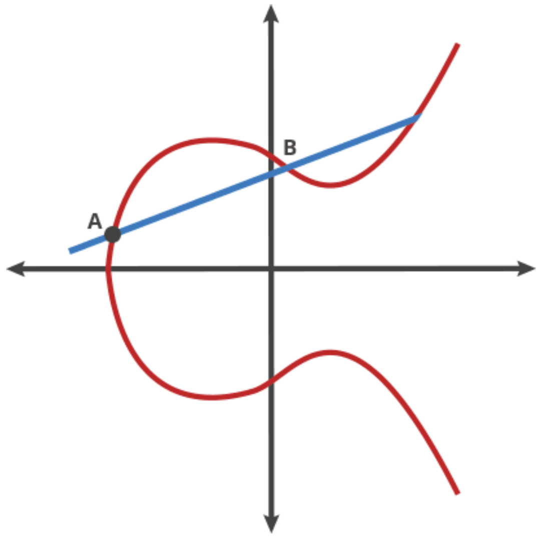
$$s = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \bmod p & \text{; if } P \neq Q \text{ (point addition)} \\ \dfrac{3x_1^2 + a}{2y_1} \bmod p & \text{; if } P = Q \text{ (point doubling)} \end{cases}$$





Horizontal Symmetry

- Take a closer look at the elliptic curve plotted above. It has several interesting properties.
- One of these is horizontal symmetry. Any point on the curve can be reflected over the x axis and remain the same curve. A more interesting property is that any non-vertical line will intersect the curve in at most three places.
- Let's imagine this curve as the setting for a bizarre game of billiards. Take any two points on the curve and draw a line through them, it will intersect the curve at exactly one more place. In this game of billiards, you take a ball at point A, shoot it towards point B. When it hits the curve, the ball bounces either straight up (if it's below the x-axis) or straight down (if it's above the x-axis) to the other side of the curve.



Computations on Elliptic Curves

- **Example**: Given E: $y^2 = x^3+2x+2 \ mod \ 17$ and point $P=(5,1)$

  Goal: Compute $2P = P+P = (5,1)+(5,1) = (x_3,y_3)$

$$s = \frac{3x_1^2+a}{2y_1} = (2 \cdot 1)^{-1}(3 \cdot 5^2 + 2) = 2^{-1} \cdot 9 \equiv 9 \cdot 9 \equiv 13 \ mod \ 17$$

$$x_3 = s^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6 \ mod \ 17$$

$$y_3 = s(x_1 - x_3) - y_1 = 13(5-6) - 1 = -14 \equiv 3 \ mod \ 17$$

Finally $2P = (5,1) + (5,1) = (6,3)$

Computations on Elliptic Curves

- The points on an elliptic curve and the point at infinity θ form cyclic subgroups

$2P = (5,1)+(5,1) = (6,3)$

$3P = 2P+P = (10,6)$

$4P = (3,1)$

$5P = (9,16)$

$6P = (16,13)$

$7P = (0,6)$

$8P = (13,7)$

$9P = (7,6)$

$10P = (7,11)$

$11P = (13,10)$

$12P = (0,11)$

$13P = (16,4)$

$14P = (9,1)$

$15P = (3,16)$

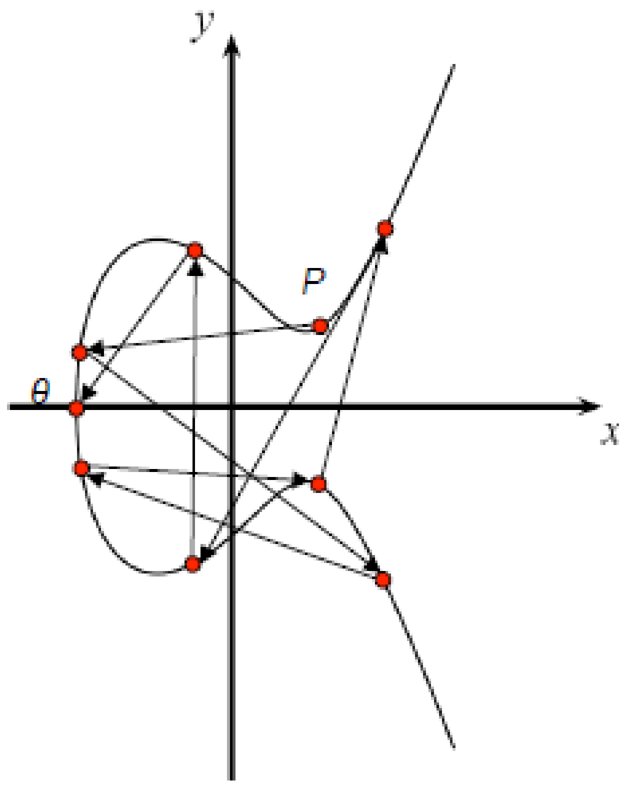
$16P = (10,11)$

$17P = (6,14)$

$18P = (5,16)$

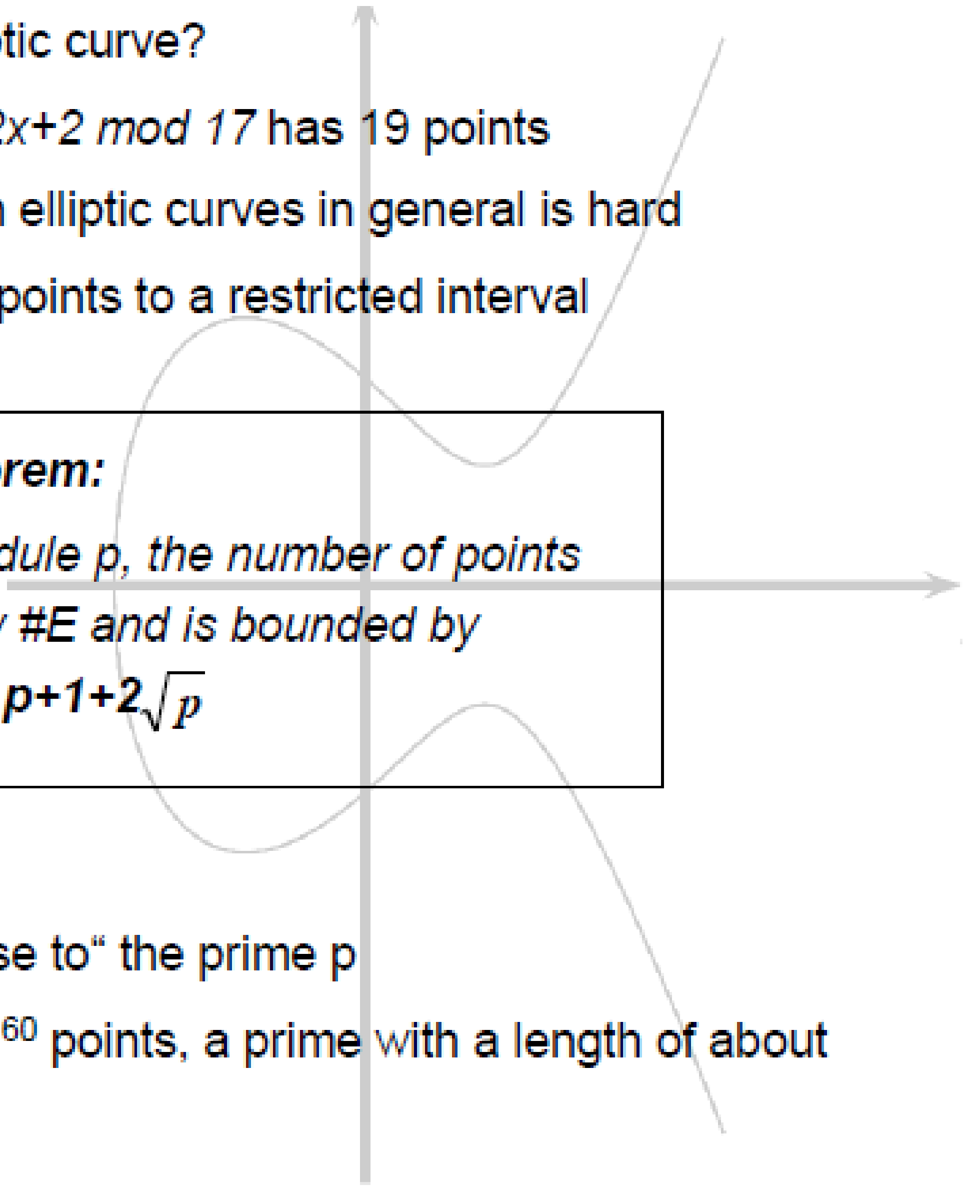
$19P = \theta$

This elliptic curve has order $\#E = |E| = 19$ since it contains 19 points in its cyclic group.



Number of Points on an Elliptic Curve

- How many points can be on an arbitrary elliptic curve?
  - Consider previous example: *E*: $y^2 = x^3+2x+2 \bmod 17$ has 19 points
  - However, determining the point count on elliptic curves in general is hard
- But Hasse's theorem bounds the number of points to a restricted interval



Definition: Hasse's Theorem:

Given an elliptic curve module p, the number of points on the curve is denoted by #E and is bounded by

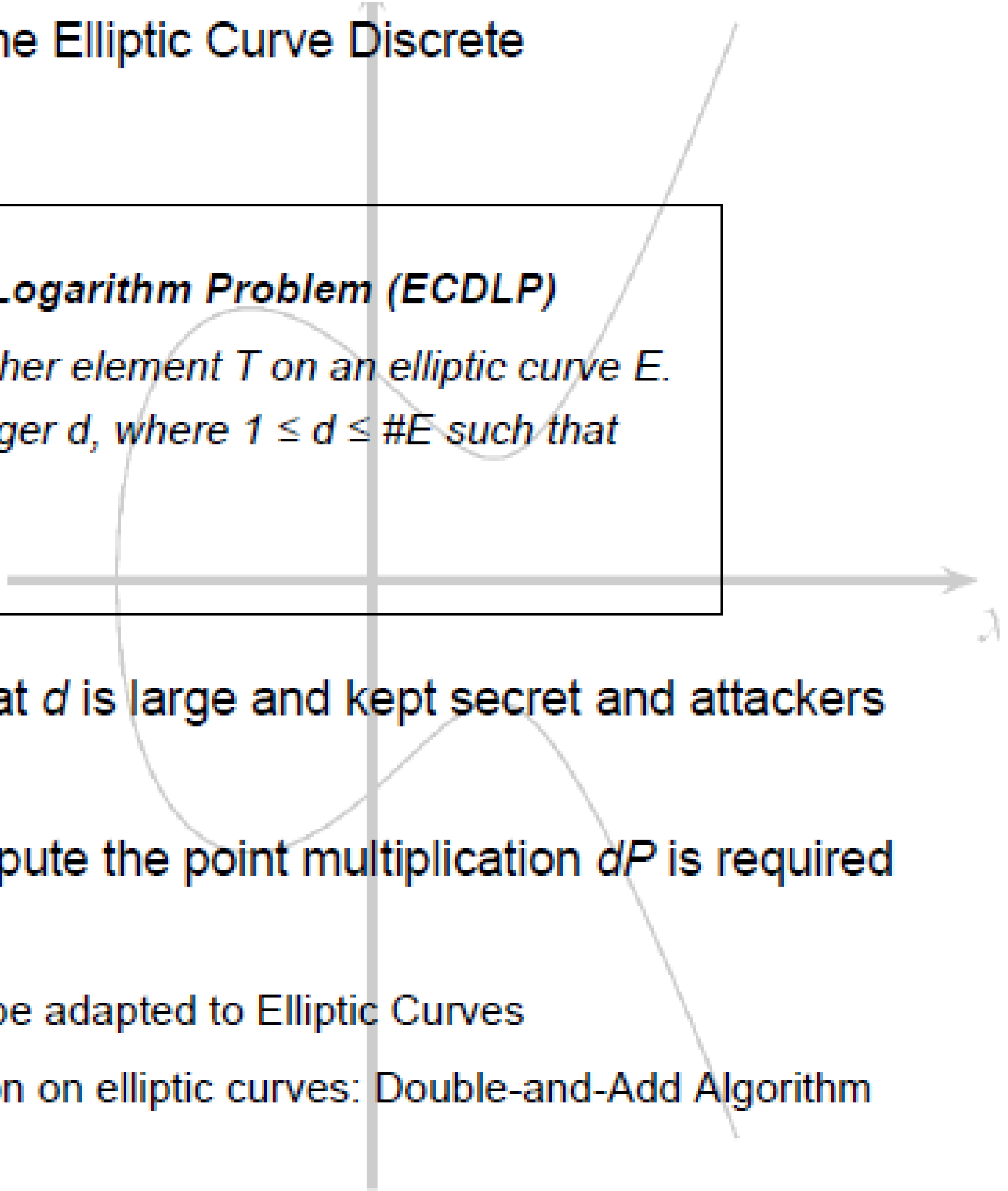
$$p+1-2\sqrt{p} \leq \#E \leq p+1+2\sqrt{p}$$

- **Interpretation:** The number of points is „close to" the prime p
- **Example:** To generate a curve with about 2^{160} points, a prime with a length of about 160 bits is required

Elliptic Curve Discrete Logarithm Problem

- Cryptosystems rely on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP)

> ***Definition: Elliptic Curve Discrete Logarithm Problem (ECDLP)***
>
> *Given a primitive element P and another element T on an elliptic curve E. The ECDL problem is finding the integer d, where 1 ≤ d ≤ #E such that*
>
> $$\underbrace{P + P + \ldots + P}_{d\ times} = dP = T.$$



- Cryptosystems are based on the idea that *d* is large and kept secret and attackers cannot compute it easily
- If *d* is known, an efficient method to compute the point multiplication *dP* is required to create a reasonable cryptosystem
  - Known Square-and-Multiply Method can be adapted to Elliptic Curves
  - The method for efficient point multiplication on elliptic curves: Double-and-Add Algorithm

Double-and-Add Algorithm for Point Multiplication

- **Double-and-Add Algorithm**

Input: Elliptic curve *E*, an elliptic curve point *P* and a scalar *d* with bits d_i

Output: $T = d\,P$

Initialization:

$T = P$

Algorithm:

```
FOR $i = t-1$ DOWNTO 0
    $T = T + T \bmod n$
    IF $d_i = 1$
        $T = T + P \bmod n$
RETURN ($T$)
```

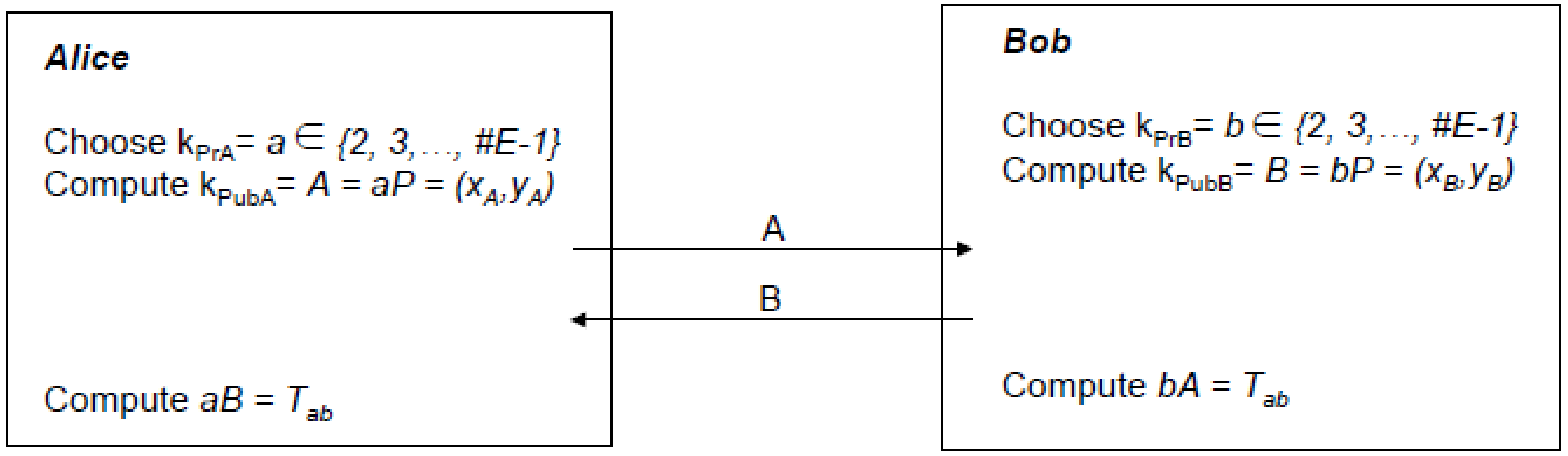
Example: $26P = (11010_2)P = (d_4 d_3 d_2 d_1 d_0)_2\, P$.

Step		
#0	$P = 1_2 P$	inital setting
#1*a*	$P+P = 2P = \mathbf{10_2} P$	DOUBLE (bit d_3)
#1*b*	$2P+P = 3P = 10^2\, P + 1_2 P = \mathbf{11_2} P$	ADD (bit d_3=1)
#2*a*	$3P+3P = 6P = 2(11_2 P) = \mathbf{110_2} P$	DOUBLE (bit d_2)
#2*b*		no ADD ($d_2 = 0$)
#3*a*	$6P+6P = 12P = 2(110_2 P) = \mathbf{1100_2} P$	DOUBLE (bit d_1)
#3*b*	$12P+P = 13P = 1100_2 P + 1_2\, P = \mathbf{1101_2} P$	ADD (bit d_1=1)
#4*a*	$13P+13P = 26P = 2(1101_2 P) = \mathbf{11010_2} P$	DOUBLE (bit d_0)
#4*b*		no ADD ($d_0 = 0$)

Elliptic Curve Diffie Helman Protocol

The Elliptic Curve Diffie-Hellman Key Exchange (ECDH)

- Given a prime p, a suitable elliptic curve E and a point $P=(x_P,y_P)$
- The Elliptic Curve Diffie-Hellman Key Exchange is defined by the following protocol:

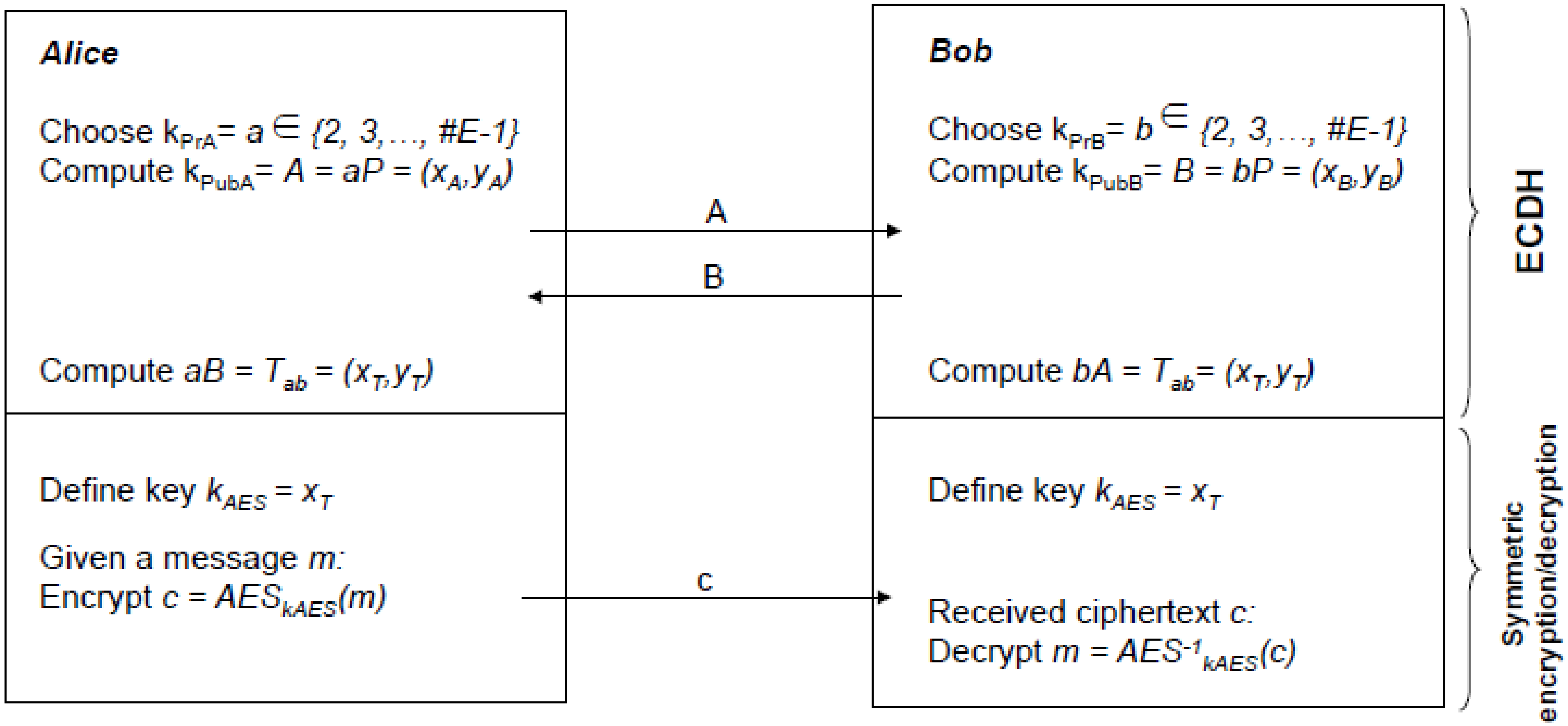


Alice
Choose $k_{PrA} = a \in \{2, 3, \ldots, \#E-1\}$
Compute $k_{PubA} = A = aP = (x_A, y_A)$
Compute $aB = T_{ab}$
A
B
Bob
Choose $k_{PrB} = b \in \{2, 3, \ldots, \#E-1\}$
Compute $k_{PubB} = B = bP = (x_B, y_B)$
Compute $bA = T_{ab}$

- Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$
- Proof for correctness:
  - Alice computes aB=a(bP)=abP
  - Bob computes bA=b(aP)=abP since group is associative
- One of the coordinates of the point T_{AB} (usually the x-coordinate) can be used as session key (often after applying a hash function)

The Elliptic Curve Diffie-Hellman Key Exchange (ECDH) - Contd

- The ECDH is often used to derive session keys for (symmetric) encryption
- One of the coordinates of the point T_{AB} (usually the x-coordinate) is taken as session key

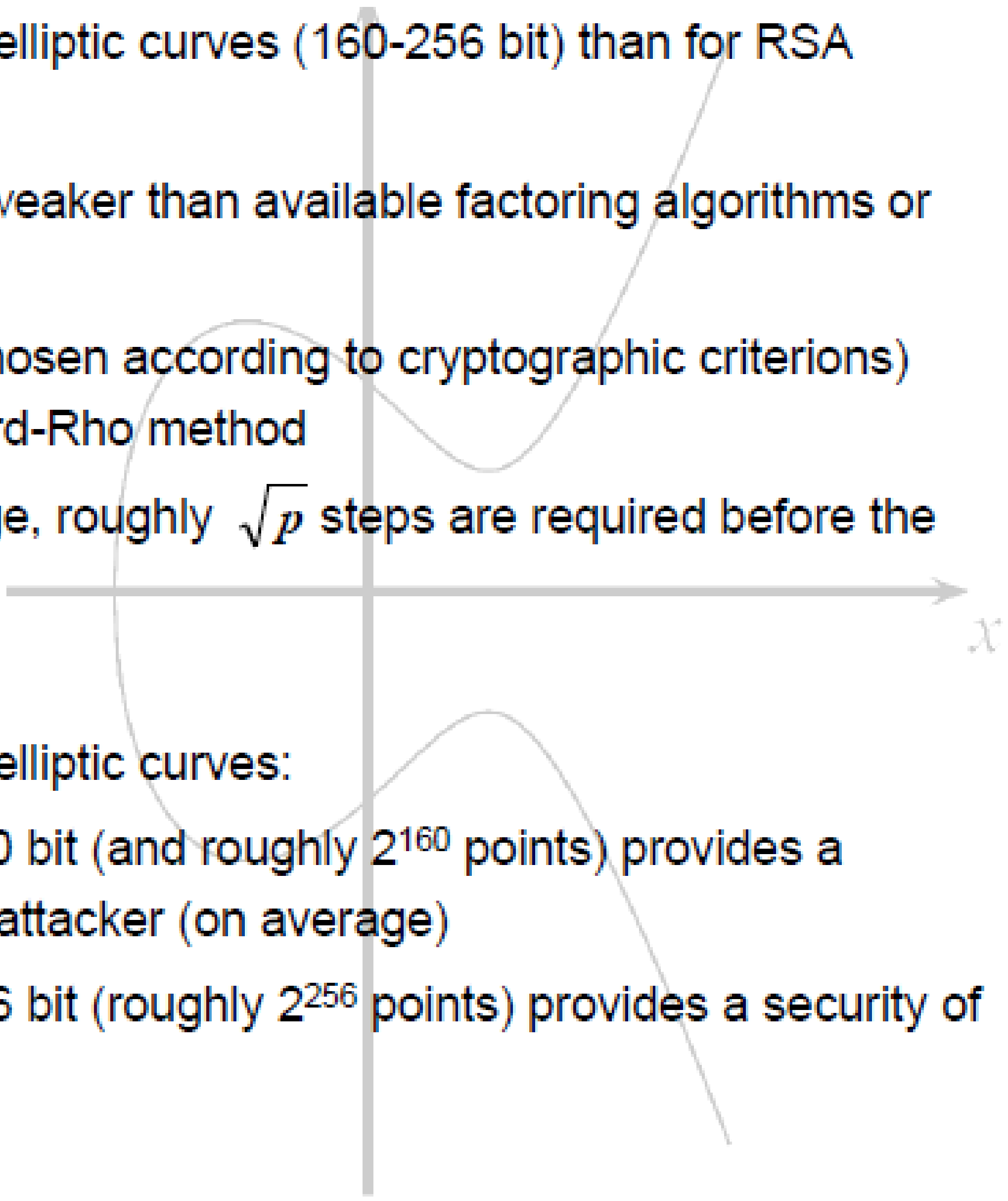


Alice
Choose $k_{PrA} = a \in \{2, 3, \ldots, \#E-1\}$
Compute $k_{PubA} = A = aP = (x_A, y_A)$
Compute $aB = T_{ab} = (x_T, y_T)$
Define key $k_{AES} = x_T$
Given a message m:
Encrypt $c = AES_{kAES}(m)$
Bob
Choose $k_{PrB} = b \in \{2, 3, \ldots, \#E-1\}$
Compute $k_{PubB} = B = bP = (x_B, y_B)$
Compute $bA = T_{ab} = (x_T, y_T)$
Define key $k_{AES} = x_T$
Received ciphertext c:
Decrypt $m = AES^{-1}_{kAES}(c)$
A
B
c
ECDH
Symmetric encryption/decryption

- In some cases, a hash function (see next chapters) is used to derive the session key

Security Aspects

Security Aspects



- Why are parameters signficantly smaller for elliptic curves (160-256 bit) than for RSA (1024-3076 bit)?
  - Attacks on groups of elliptic curves are weaker than available factoring algorithms or integer DL attacks
  - Best known attacks on elliptic curves (chosen according to cryptographic criterions) are the Baby-Step Giant-Step and Pollard-Rho method
  - Complexity of these methods: on average, roughly $\sqrt{p}$ steps are required before the ECDLP can be successfully solved

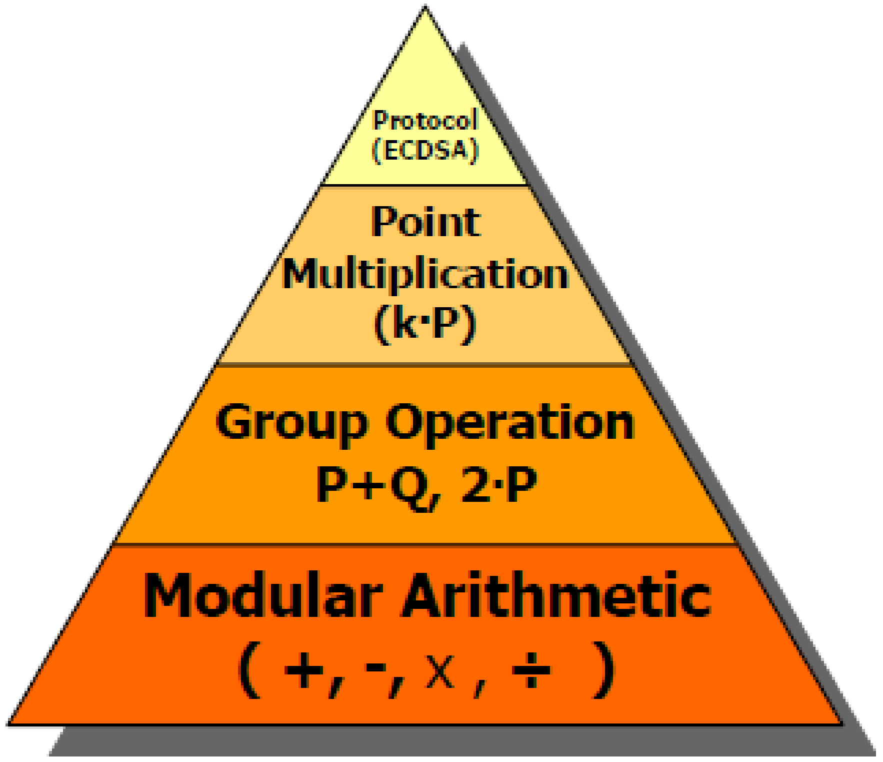
- Implications to practical parameter sizes for elliptic curves:
  - An elliptic curve using a prime p with 160 bit (and roughly 2^{160} points) provides a security of 2^{80} steps that required by an attacker (on average)
  - An elliptic curve using a prime p with 256 bit (roughly 2^{256} points) provides a security of 2^{128} steps on average

Implementation in Hardware and Software

Implementations in Hardware and Software

- Elliptic curve computations usually regarded as consisting of four layers:
  - Basic modular arithmetic operations are computationally most expensive
  - Group operation implements point doubling and point addition
  - Point multiplication can be implemented using the Double-and-Add method
  - Upper layer protocols like ECDH and ECDSA

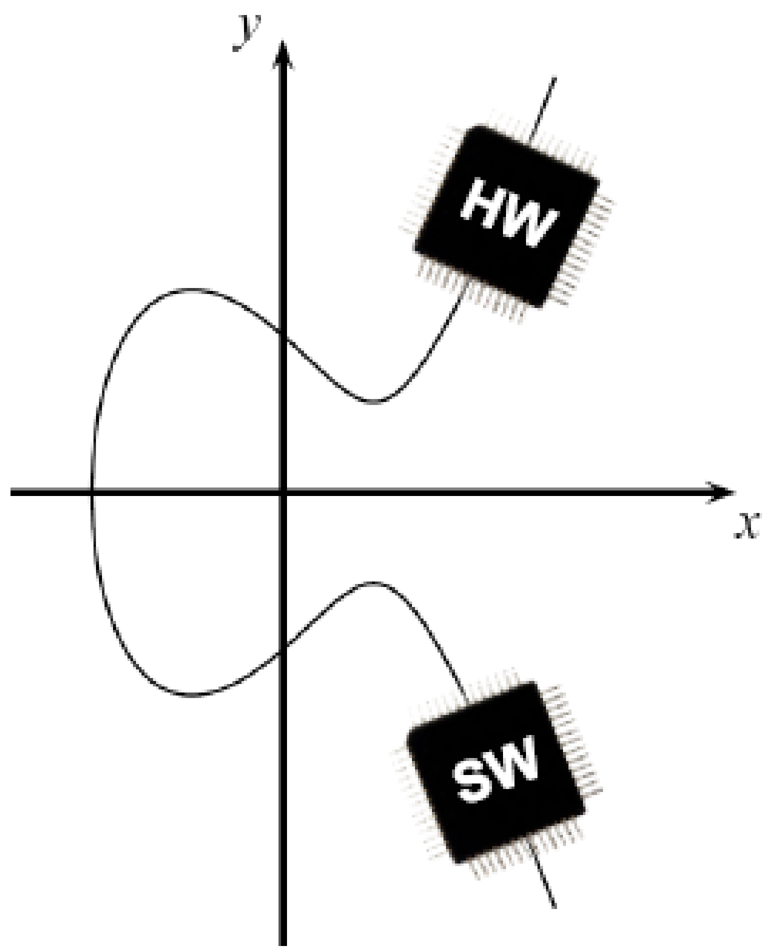
- Most efforts should go in optimizations of the modular arithmetic operations, such as
  - Modular addition and subtraction
  - Modular multiplication
  - Modular inversion


Protocol (ECDSA)
Point Multiplication (k·P)
Group Operation P+Q, 2·P
Modular Arithmetic (+, -, x , ÷)

Implementations in Hardware and Software

- Software implementations
  - Optimized 256-bit ECC implementation on 3GHz 64-bit CPU requires about *2 ms* per point multiplication
  - Less powerful microprocessors (e.g, on SmartCards or cell phones) even take significantly longer (>*10 ms*)

- Hardware implementations
  - High-performance implementations with 256-bit special primes can compute a point multiplication in a few hundred microseconds on reconfigurable hardware
  - Dedicated chips for ECC can compute a point multiplication even in a few ten microseconds



ECDSA

Signing with Elliptic Curves

- The main standard algorithm you can use for signing with ECC is elliptic curve digital signature algorithm (ECDSA).
- This algorithm has replaced RSA signatures and classical DSA signatures in many applications.
- ECDSA is a NIST standard, is supported in the TLS and SSH protocols, and is the main signature algorithm in many blockchain platforms, including Bitcoin and Ethereum.

- As with all signature schemes, ECDSA consists of a signature generation algorithm that the signer uses to create a signature using their private key and a verification algorithm that a verifier uses to check a signature's correctness given the signer's public key.
- The signer holds a number, d, as a private key, and verifiers hold the public key, P = dG.
- Both know in advance what elliptic curve to use, its order (n, the number of points in the curve), and the coordinates of a base point, G.

ECDSA Signature Generation

- To sign a message, the signer first hashes the message with a cryptographic hash function such as SHA-256 or BLAKE2 to generate a hash value, h, that you interpret as a number between 0 and n – 1.
- Next, the signer picks a random number, k, between 1 and n – 1 and computes kG, a point with the coordinates (x, y).
- The signer now sets r = x mod n and computes s = (h + rd) / k mod n and then uses these values as the signature (r, s).

- The length of the signature depends on the coordinate lengths you're using.
- For example, when you're working with a curve where coordinates are 256-bit numbers, r and s are both 256 bits long, yielding a 512-bit-long signature.

ECDSA Signature Verification

- The ECDSA verification algorithm uses a signer's public key to verify the validity of a signature.
- To verify an ECDSA signature (r, s) and a message's hash, h, the verifier first computes w = 1 / s, the inverse of s in the signature, which is equal to k / (h + rd) mod n, since s is defined as s = (h + rd) / k. Next, the verifier multiplies w with h to find u according to the following formula:

$$wh = hk / (h + rd) = u$$

- The verifier then multiplies w with r to find v:

$$wr = rk / (h + rd) = v$$

- Given u and v, the verifier computes the point Q according to the following formula:

$$Q = uG + vP$$

- Here, P is the signer's public key, which is equal to dG, and the verifier accepts the signature only if the x-coordinate of Q is equal to the value r from the signature.

ECDSA Signature Verification (Contd)

- This process works because, as a last step, you compute the point Q by substituting the public key P with its actual value dG:

$$uG + vdG = (u + vd)G$$

- When you replace u and v with their actual values, you obtain the following:

$$u + vd = hk \,/\, (h + rd) + drk \,/\, (h + rd) = (hk + drk) \,/\, (h + rd) = k(h + dr) \,/\, (h + rd) = k$$

- This tells you that (u + vd) is equal to the value k, chosen during signature generation, and that uG + vdG is equal to the point kG.
- In other words, the verification algorithm succeeds in computing point kG, the same point computed during signature generation.
- Validation is complete once a verifier confirms that kG’s x-coordinate is equal to the r received; otherwise, the signature is rejected as invalid.

ECC based Encryption and Decryption.

ECC and Encryption

1. **ECIES (Elliptic Curve Integrated Encryption Scheme)**

- Uses ECC for key exchange and a symmetric cipher (like AES) for actual message encryption.
- Steps:
    1. Generate an ephemeral ECC key pair.
    2. Compute a shared secret using ECDH (Elliptic Curve Diffie-Hellman).
    3. Derive a symmetric key from the shared secret (using KDF).
    4. Encrypt the message with AES (or similar).
    5. Send the ephemeral public key + ciphertext.
- **Why simpler?**
    - The heavy lifting (encryption) is done by AES, which is efficient.
    - ECC is only used for key exchange.

ECC and Encryption (Continued...)

2. ECDH + Symmetric Encryption

- Even simpler than ECIES:
- - Both parties agree on a shared secret using ECDH.
- - Use that secret as a key for AES or ChaCha20.
- **No padding or point encoding issues** because the message is not directly mapped to the curve.

3. Direct Scalar Masking (Not Recommended for Production)

- Conceptually simple:
  - Represent message as an integer "m" .
  - Compute:
    $C = (kG, m + k . Q \bmod p)$
  - Decrypt:
    $m = (m + k . Q) - d . (kG)$
- Problem in this approach: Requires mapping message to integer and is vulnerable to attacks if not padded properly.
- This is only for enabling an understanding.

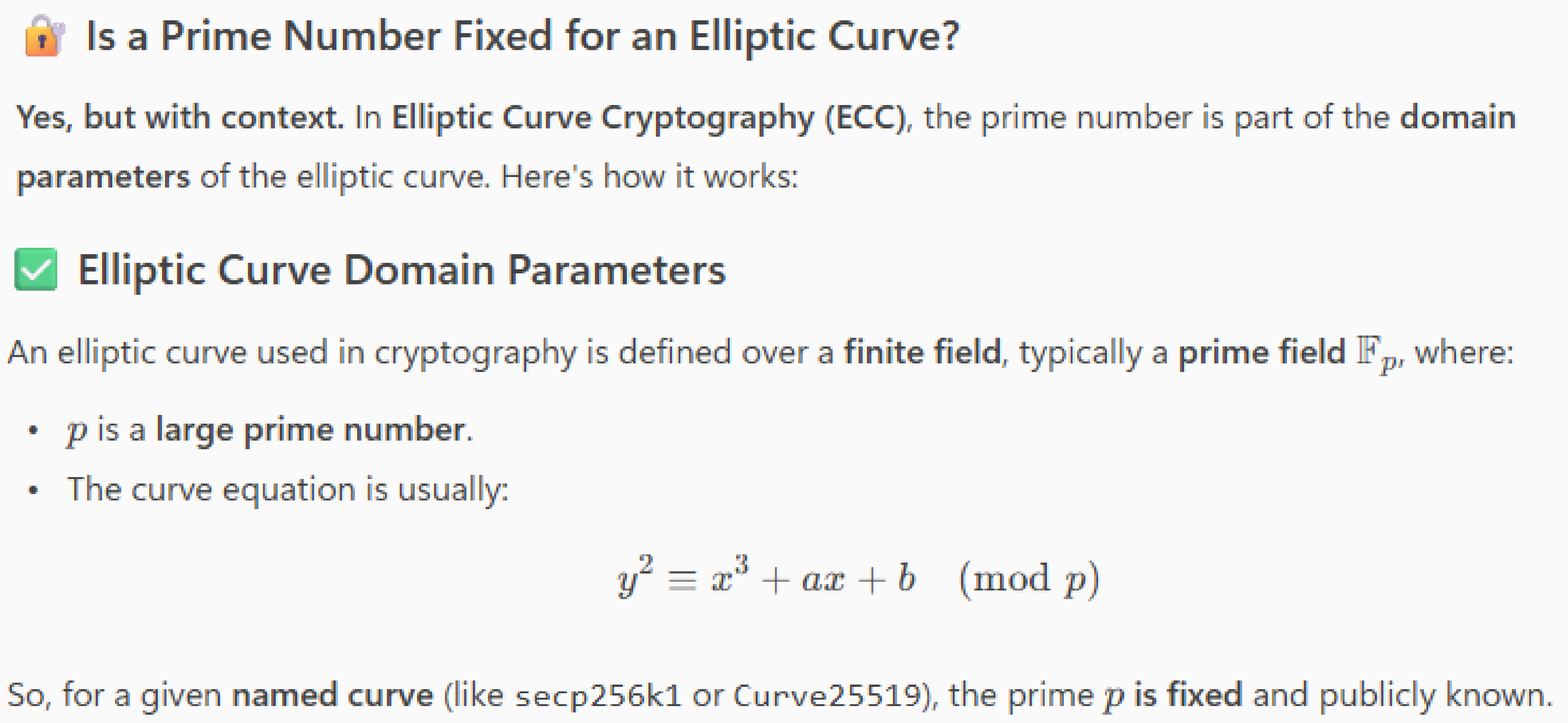
◆ ---

ECC and Encryption (Continued...)

4. **Hybrid ECC + Stream Cipher**
   - Use ECC to generate a one-time pad (via shared secret).
   - XOR the message with the derived keystream.
   - Very lightweight for IoT or constrained devices.

Additions

Additional information/points



🔐 Is a Prime Number Fixed for an Elliptic Curve?

Yes, but with context. In **Elliptic Curve Cryptography (ECC)**, the prime number is part of the **domain parameters** of the elliptic curve. Here's how it works:

✅ Elliptic Curve Domain Parameters

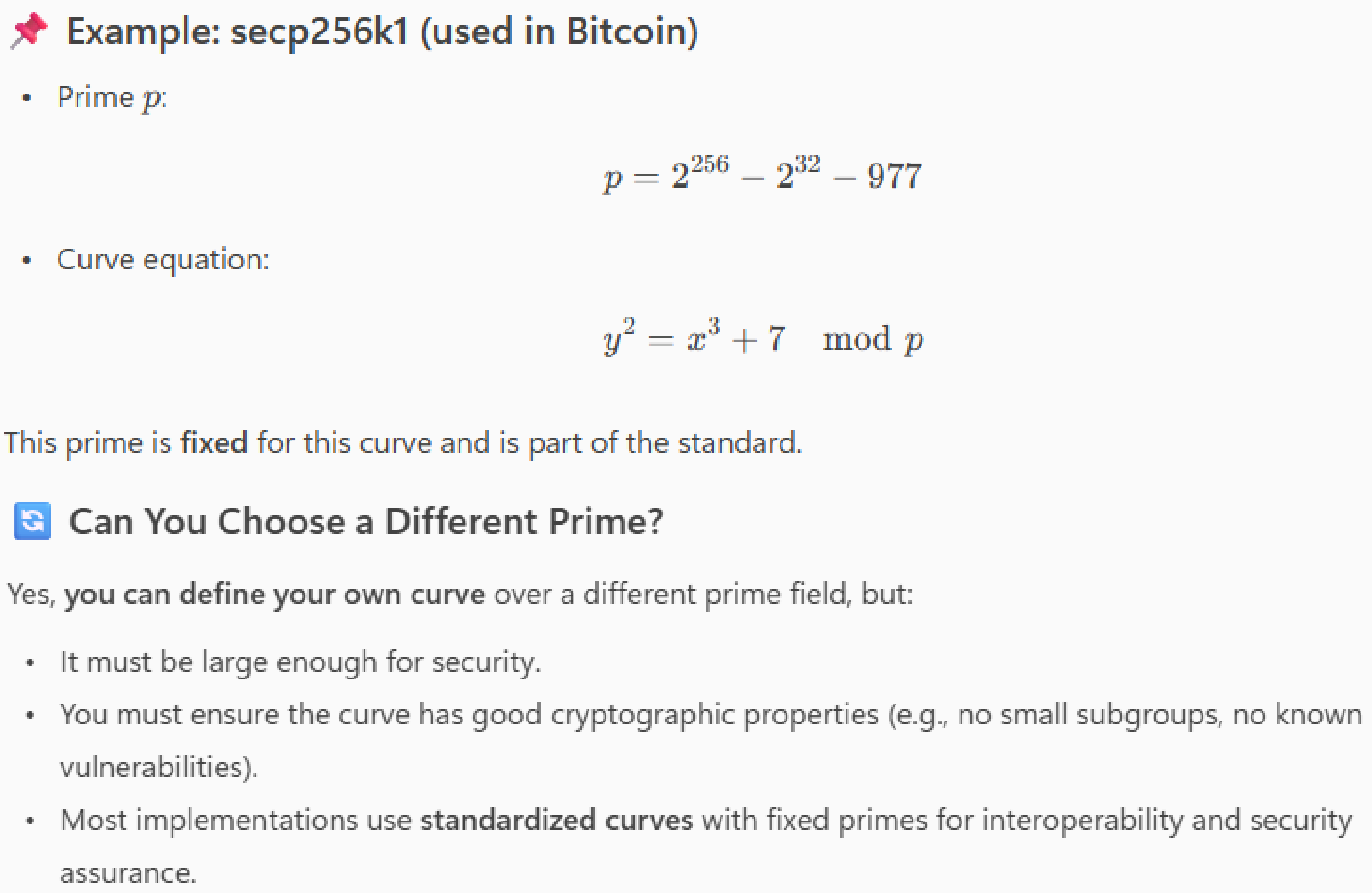
An elliptic curve used in cryptography is defined over a **finite field**, typically a **prime field** $\mathbb{F}_p$, where:

- p is a **large prime number**.
- The curve equation is usually:

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

So, for a given **named curve** (like `secp256k1` or `Curve25519`), the prime p **is fixed** and publicly known.

Additional information/points



📌 Example: secp256k1 (used in Bitcoin)

- Prime p:

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

- Curve equation:

$$y^2 = x^3 + 7 \mod p$$

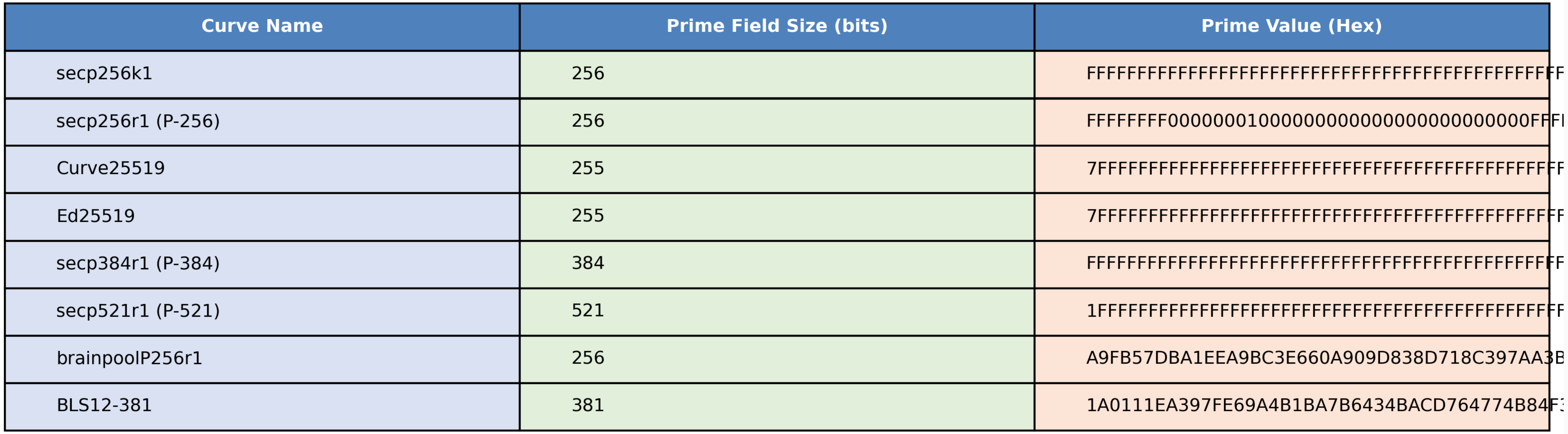
This prime is **fixed** for this curve and is part of the standard.

🔄 Can You Choose a Different Prime?

Yes, **you can define your own curve** over a different prime field, but:

- It must be large enough for security.
- You must ensure the curve has good cryptographic properties (e.g., no small subgroups, no known vulnerabilities).
- Most implementations use **standardized curves** with fixed primes for interoperability and security assurance.

Additional information/points



Curve Name	Prime Field Size (bits)	Prime Value (Hex)
secp256k1	256	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
secp256r1 (P-256)	256	FFFFFFFF00000001000000000000000000000000FFF
Curve25519	255	7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Ed25519	255	7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
secp384r1 (P-384)	384	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
secp521r1 (P-521)	521	1FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
brainpoolP256r1	256	A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B
BLS12-381	381	1A0111EA397FE69A4B1BA7B6434BACD764774B84F3

Conclusion

Lessons Learnt

- Elliptic Curve Cryptography (ECC) is based on the discrete logarithm problem. It requires, for instance, arithmetic modulo a prime.
- ECC can be used for key exchange, for digital signatures and for encryption.
- ECC provides the same level of security as RSA or discrete logarithm systems over Zp with considerably shorter operands (approximately 160–256 bit vs. 1024–3072 bit), which results in shorter ciphertexts and signatures.
- In many cases ECC has performance advantages over other public-key algorithms.
- ECC is slowly gaining popularity in applications, compared to other public-key schemes, i.e., many new applications, especially on embedded platforms, make use of elliptic curve cryptography.

Thank you – Q&A