NT219-Cryptography

Week 08: Asymmetric Cryptography (P3)

PhD. Ngoc-Tu Nguyen

tunn@uit.edu.vn

Outline

- Why asymmetric cryptography?
- Factoring Based Cryptography (P1)
  - RSA
  - *Rabin*
- Logarithm Based Cryptography (P2)
- Elliptic Curve Cryptography (P3)
- Some advanced cryptography system (quantum resistance)

Warmup

- **Agreement a symmetric key (AES) with a server**

1. RSA, or DHE?
2. Setup system parameters?
3. Exchange public keys or send AES Key encapsulation (RSA-based Encryption)
4. Compute the AES session key?

Implementation the RSA Algorithm (review)

Key Generation by Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Fermat's little theorem

$\lambda(n) = \text{lcm}(p - 1, q - 1)$

$e.d = 1 \bmod \phi(n)$

Carmichael's theorem

$e.d = 1 \bmod \lambda(n)$

Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

$$C^d \bmod n = (M^e)^d \bmod n$$
$$= M^{ed} \bmod n =? M$$

Implementation The RSA Algorithm ((review))

Generalization of Fermat's little theorem

- Let $n = p.q,\ \lambda(n) = lcm(p-1, q-1)$?
- $for\ all\ a\ st.\ \gcd(a,n) = 1,$ $a^{\lambda(n)} mod\ n =?$

$$PU = \{n, e\}$$

$$PR = \{n, p, q, d, d_p, d_q, q_{inv}\}$$

$$d_p = d\ mod\ (p-1),$$
$$d_q = d\ mod\ (q-1),$$
$$q_{inv} = q^{-1} mod\ p$$

📖 Openssl_key_geneeration.txt

$$n = 15, \mathbb{Z}_{15} = \{0,1,2, \ldots 15\}$$

$$n = 3.5 = 15, \varphi(15) = 8,$$
$$\lambda(n) = lcm(2,4) = 4$$

a	$a^8 mod\ 15$	$a^4 mod\ 15$
1		
2		
4		
7		
8		
11		
13		
14		

Implementation The RSA Algorithm

$$PU = \{n, e\}$$

- Encrypt

$$C = m^e mod\, n$$

$$PR = \{n, p, q, d, d_p, d_q, q_{inv}\}$$

$$d_p = d\ mod\ (p - 1),$$
$$d_q = d\ mod\ (q - 1),$$
$$q_{inv} = q^{-1} mod\ p$$

- Decrypt

$$m = C^d mod\, n\, !$$

$$m_1 = C^{d_p}\ mod\ p$$
$$m_2 = C^{d_q}\ mod\ q$$
$$h = q_{inv}(m_1 - m_2) mod\ p$$
$$m = m_2 + h.q$$

📖 **Chinese remainder theorem**

ElGamal Cipher

Encryption message $\boldsymbol{m < p - 1}$ (using public key $\boldsymbol{h = g^x}$)

- Choose a random number: $r \in_R [1, p-1]$ why?
- Compute $C_1 = g^r \bmod p$;
- Computer $C_2 = m.h^r \; mod \; p = m.g^{x.r} \; mod \; p$
- Output cipher message (C_1, C_2)

Decryption (C_1, C_2) (using secret key $\boldsymbol{x}$)

- Computer $\dfrac{C_2}{(C_1)^x} \; mod \; p = \dfrac{m.g^{x.r}}{g^{r.x}} \; mod \; p$
  $= m$

Diffie-Hellman exchange Protocol (DHE)

$p = 1606938044258990275541962092341162602522202993782792835301301$

$g = 123456789$

g^a mod p =
78467374529422653579754596319852702575499692980085777948593

g^b mod p =
56004810429321812866744102134248313380262627139429941012879 8

$a =$
685408003627063
761059275919665
781694368639459
527871881531452

$b =$
362059131912941
987637880257325
269696682836735
524942246807440

$(g^b)^a \; mod \; p$

g^{ab} mod p =
437452857085801785219961443000
845969831329749878767465041215

$(g^a)^b mod \; p$

Computational hardness assumptions

- Integer factorization Problem ;

$$n(= p.q) \not\mapsto \phi(n) = (p-1)(q-1)$$

- Discrete Log Problem (DLP):

$$g, p, y = g^x mod\ p \not\mapsto x$$

- Diffie-Hellman Problem (DHP):

$$g, p, A = g^a mod\ p, B = g^b mod\ p \not\mapsto g^{ab}$$

Man-in-the middle attacks the DHE

(a, g^a)

$\xrightarrow{g^a \bmod p}$

$\xleftarrow{g^b \bmod p}$

(b, g^b)

A(Putin)

$$sk = (g^b)^a = (g^a)^b = g^{ab} \bmod p$$

B(Zelensky)

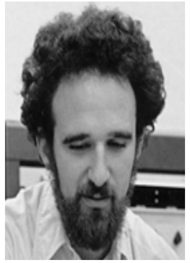
man-in-the-middle attack!

$\xrightarrow{g^a \bmod p}$

$\xleftarrow{g^m \bmod p}$

$\xrightarrow{g^m \bmod p}$

$\xleftarrow{g^b \bmod p}$



$$sk_1 = g^{am} \bmod p$$

$$sk_2 = g^{bm} \bmod p$$

Motivations

Group (G,+) can do + −

lightweight computational overhead?

Ring (R, +, ×) can do + − ×

 $Z_{2^n} = \{0,1,...,2^n - 1\}$

Field (F, +, ×) can do + − × ÷

 $Z_p = \{0,1,..., p-1\}$

Elliptic curve

An elliptic curve is a group defined over a field K

elliptic curve group $(E, \oplus)$	can do $\oplus$ $\ominus$
underlying field $(K, +, \times)$	can do $+$ $-$ $\times$ $\div$

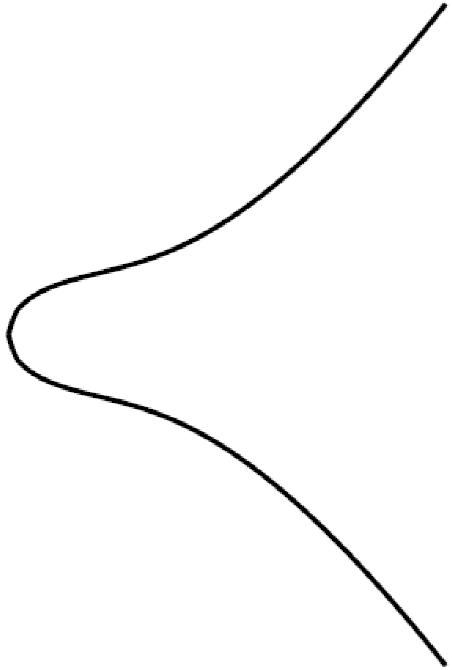
operations in underlying field are used and combined to compute the eliptic curve operation $\oplus$

Elliptic curves

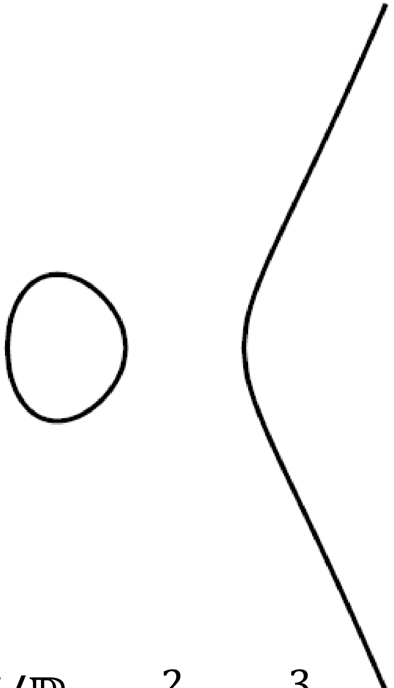
Weierstrass form

$$\boldsymbol{E/K: y^2 = x^3 + ax + b}$$

E specified by K, a, b



$E/\mathbb{R}: \ y^2 = x^3 + x + 1$



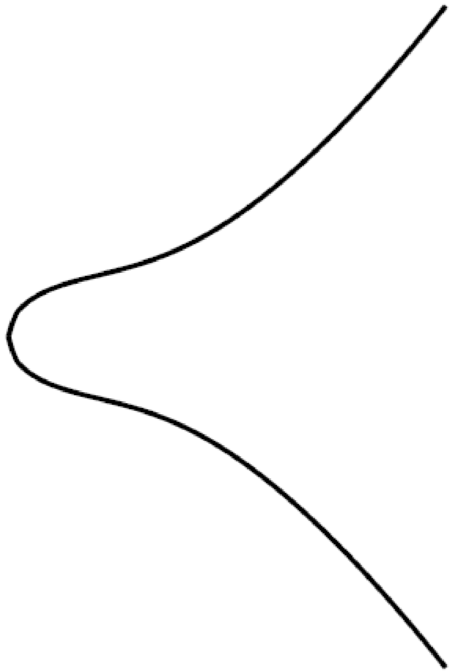
$E/\mathbb{R}: \ y^2 = x^3 - x$

Elliptic curves

Weierstrass form

$E/\mathbb{R}: \; y^2 = x^3 + x + 1$

$E/\mathbb{Z}_7: \; y^2 = x^3 + x + 1 (mod\ 7)$

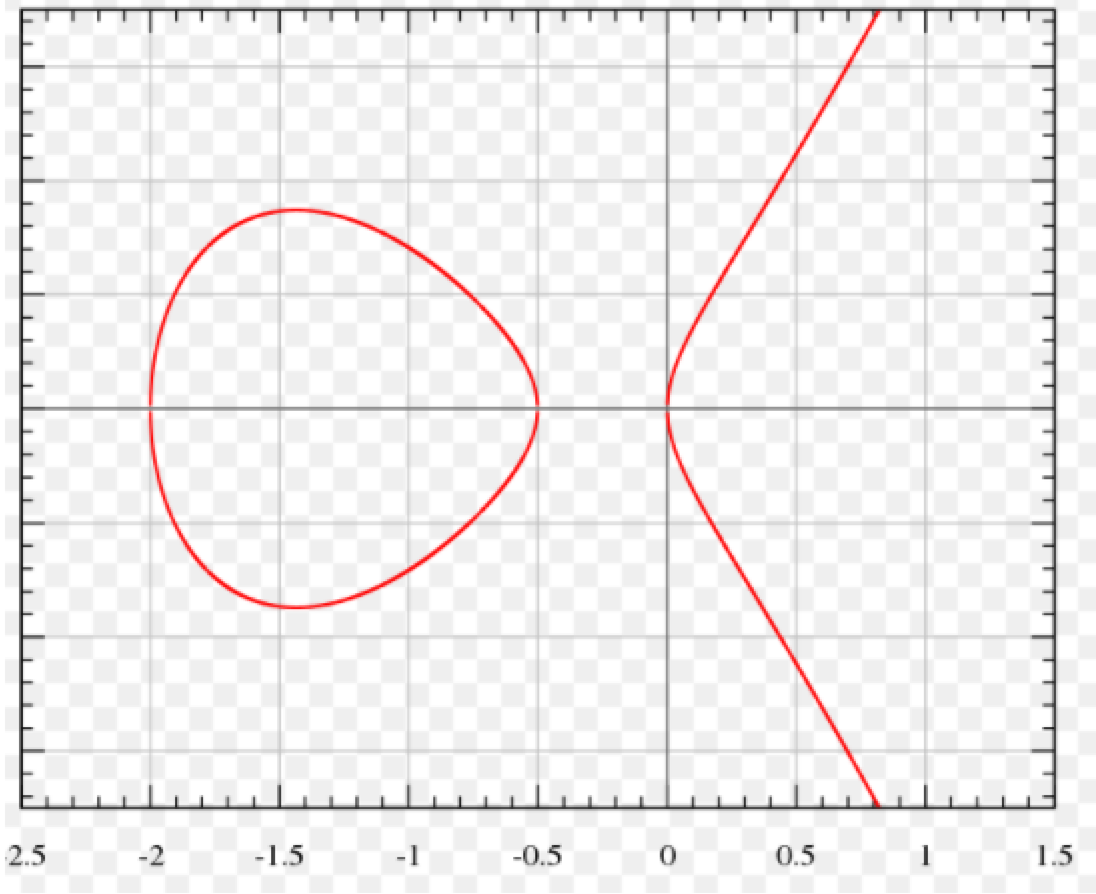


x	$\boldsymbol{y^2 = x^3 + x + 1}$	**y**
1	3	x
2	4	2,5
3	3	x
4	6	x
5	5	x
6	6	x

Montgomery form

$$E/K: By^2 = x^3 + Ax^2 + x$$

E specified by K, A, B



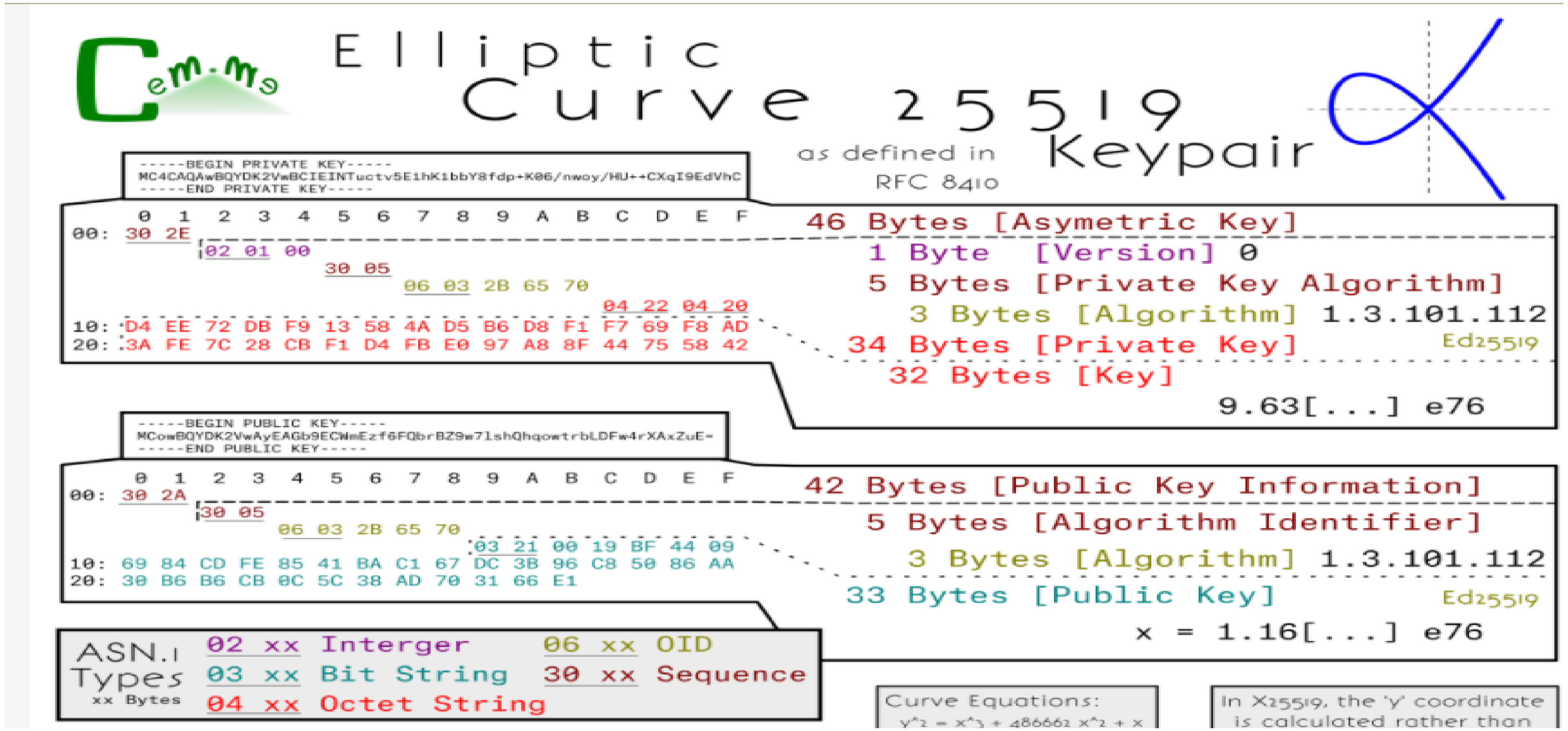
$E/\mathbb{R}:$

$$\frac{1}{4}y^2 = x^3 + \frac{5}{2}x^2 + x$$

Curve25519

$$E/K: y^2 = x^3 + 486662x^2 + x$$

where field $K = Z_{2^{255}-19}$

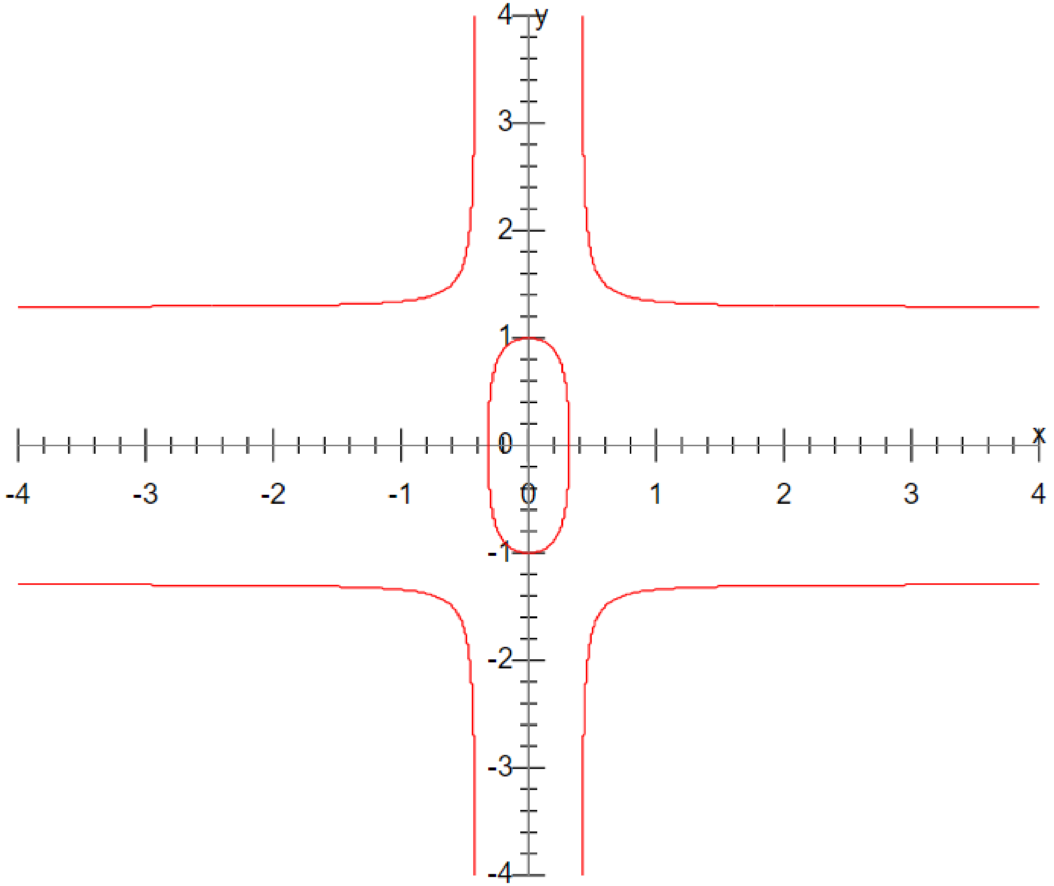


Elliptic curves

Twisted Edwards form

$$\boldsymbol{E/K}\text{: } ax^2 + y^2 = 1 + dx^2.y^2$$

E specified by: K, a, d

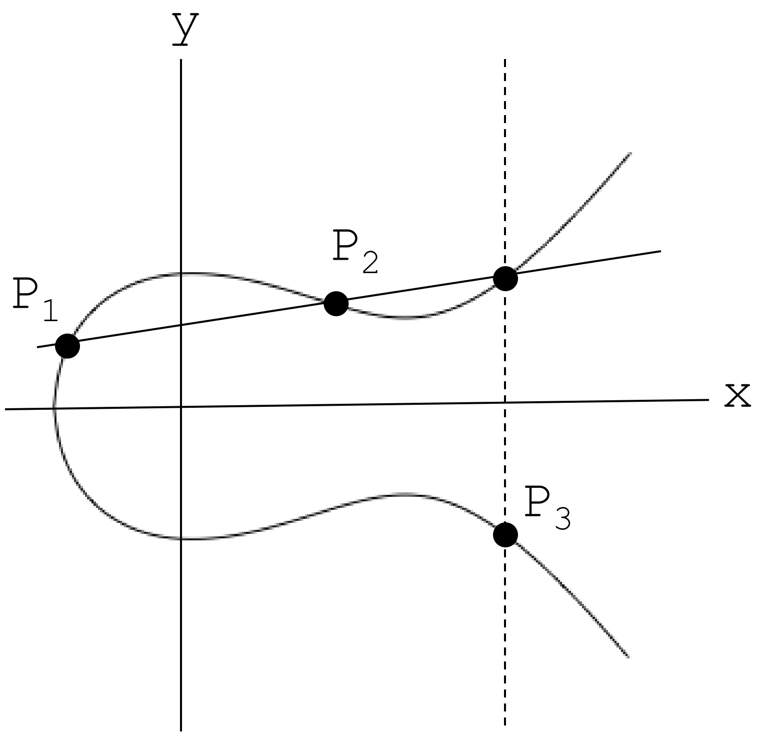


$E/\mathbb{R}$:

$$10x^2 + y^2 = 1 + 6x^2y^2$$

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

Elliptic group

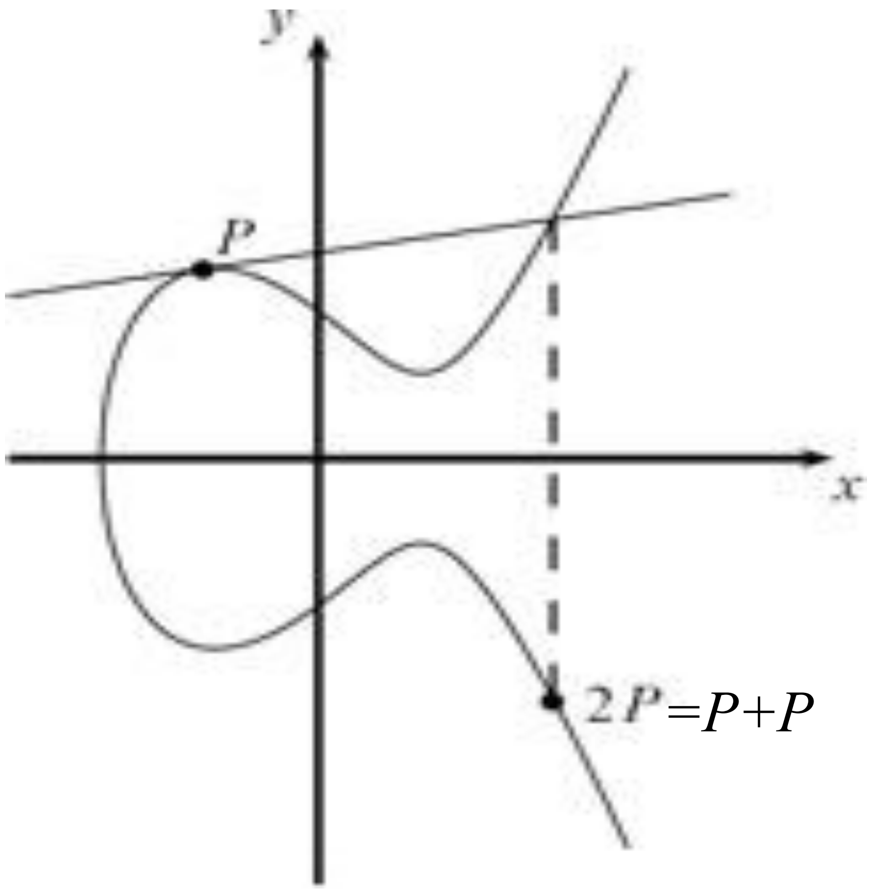


Addition of two points:

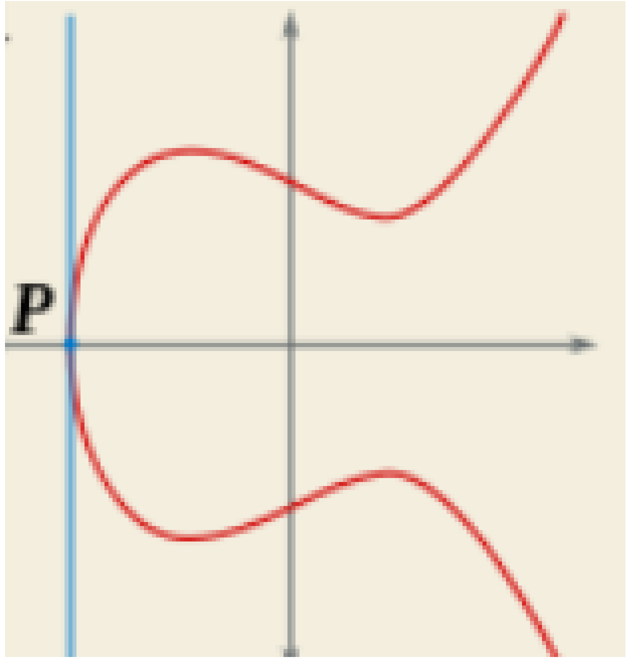
- If P_1 and P_2 are on E, we can define sum
  $P_1 + P_2 = P_2 + P_1 \overset{\text{def}}{=} P_3$
  as shown in picture

Elliptic group

Point Doubling

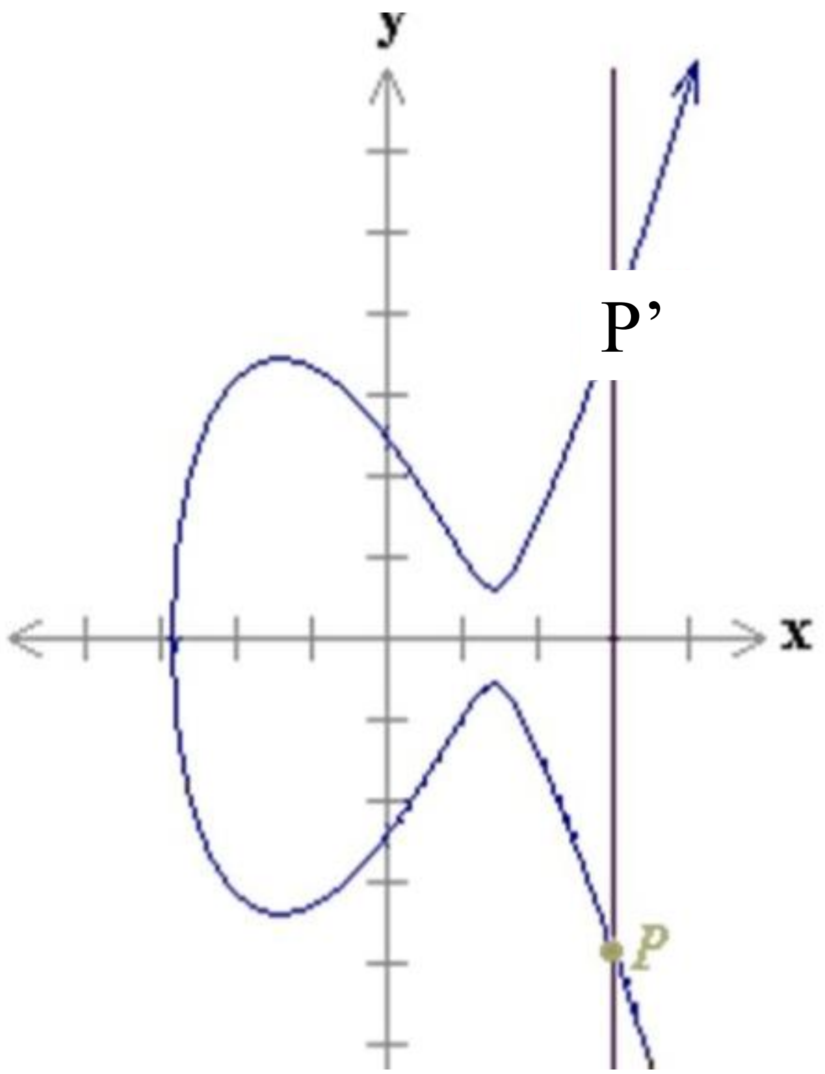


Special case



P+P= O (infinity)

Elliptic group



$P + O = P$

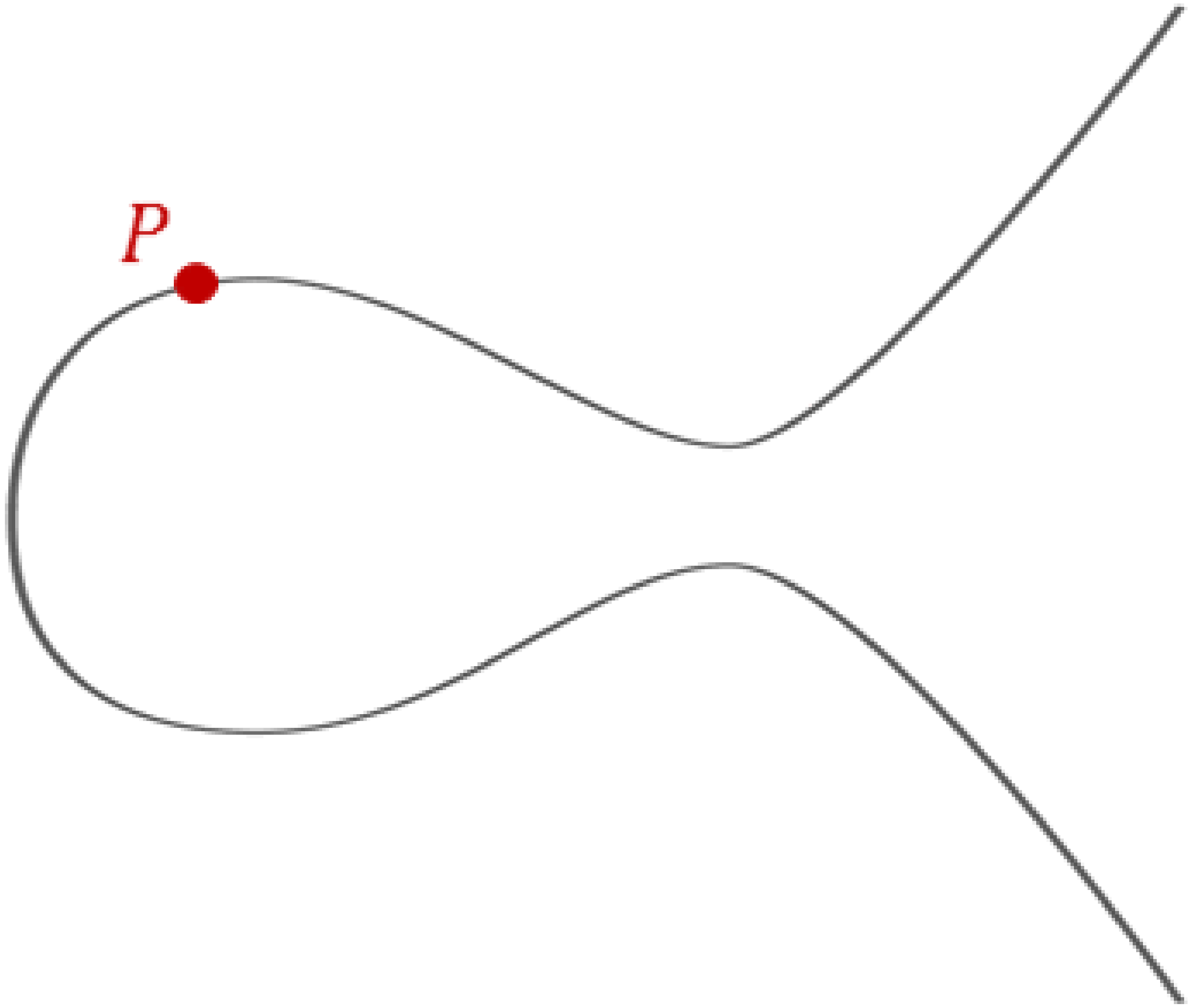
$O + P = P$

P'=-P

$P+(-P)= O$ *(infinity)*

Elliptic group

Example



Elliptic group

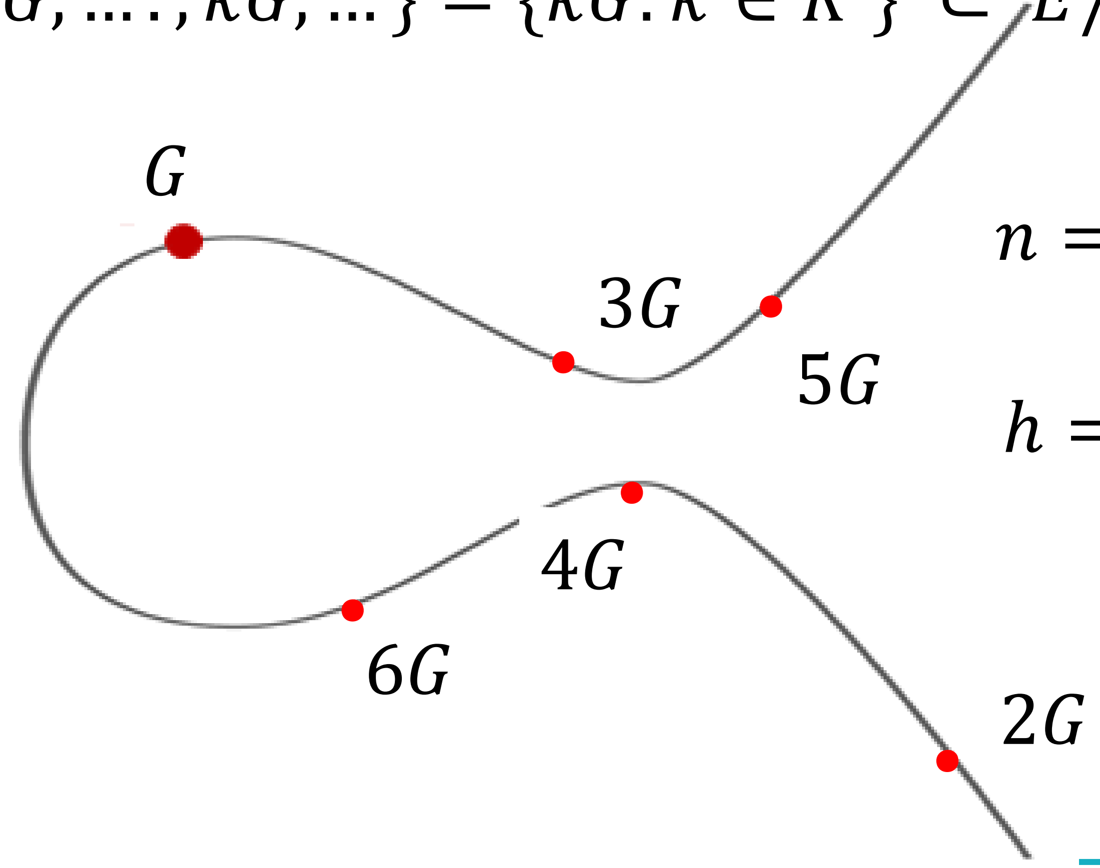
Given two points P, Q in E/K, there is a third point, denoted by $P + Q$ on E/K, and the following relations hold for all P, Q, R in E/K, where K be a finite field

- $P + Q = Q + P$ (*commutativity*)
- $(P + Q) + R = P + (Q + R)$ (*associativity*)
- $P + O = O + P = P$ (*existence of an identity element*)
- there exists $(-P)$ such that $-P + P = P + (-P) = O$ (*existence of inverses*)

Elliptic group

- **Group points E/K**
- Subgroup generated by a point G

$$< G >= \{G, 2G, \ldots., kG, \ldots\} = \{kG : k \in K\} \subset E/K$$



$$n = |< G >| = |\{kG : k \in K\}|$$

$$h = \frac{|\boldsymbol{E/K}|}{|< G >|} \textit{ is small!}$$

Elliptic group

- **ECC group parameters:**

✓ ECC equation (type, coeffection): $a, b, \ldots$

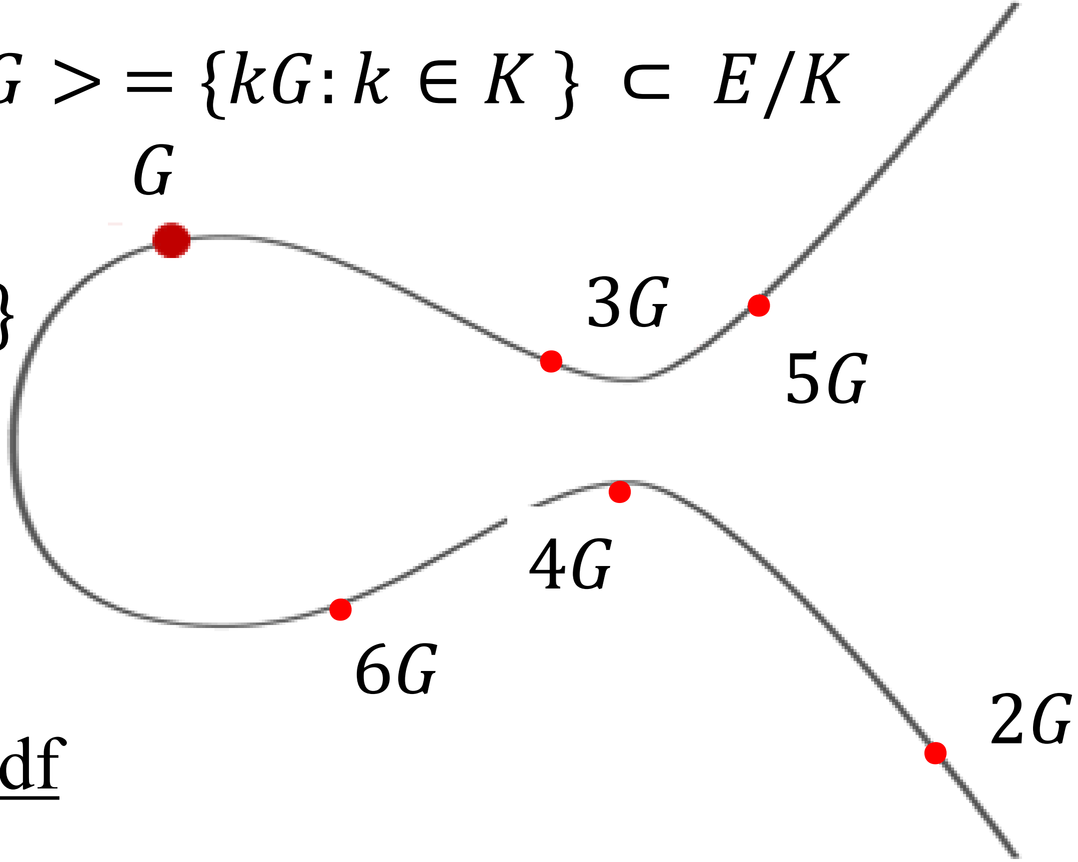
✓ Modulo: $p\ or\ f(x)$

✓ Generator point: G

✓ Order of $G: n\ = <G> = \{kG: k \in K\}$

✓ Cofactor: $h = \dfrac{|\boldsymbol{E/K}|}{|<G>|}$

$$<G> = \{kG: k \in K\} \subset E/K$$



http://www.secg.org/sec2-v2.pdf

Using Elliptic Curves In Cryptography

- Hardness assumption.
  - $\boldsymbol{d} \rightarrow \boldsymbol{Q} = \boldsymbol{dG} = \boldsymbol{G} + \boldsymbol{G} + \cdots . + \boldsymbol{G}$
  - Elliptic curve discrete logarithm problem (ECDLP)

$$G, Q(= dG) \not\mapsto d$$

where $\boldsymbol{dG} = \underbrace{\boldsymbol{G} + \boldsymbol{G} + \cdots . + \boldsymbol{G}}_{d \text{ times}}$

- Discrete Log Problem (DLP):

$$g, p, y = g^x mod\ p \not\mapsto x$$

Elliptic Curve Cryptosystems (ECC)

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie–Hellman	No	No	Yes
DSS	No	Yes	No

What Is ECC?

- Elliptic curve cryptography [ECC] is a **public-key** cryptosystem just like RSA, Rabin, and El Gamal.
- Every user has a **public key $\boldsymbol{Q} (= \boldsymbol{dG})$** and a **private** key $\boldsymbol{d}$.
  - Public key is used for encryption/signature verification.
  - Private key is used for decryption/signature generation.
- Elliptic curves are used as an extension to other current cryptosystems.
  - Elliptic Curve Diffie-Hellman Key Exchange
  - Elliptic Curve Digital Signature Algorithm

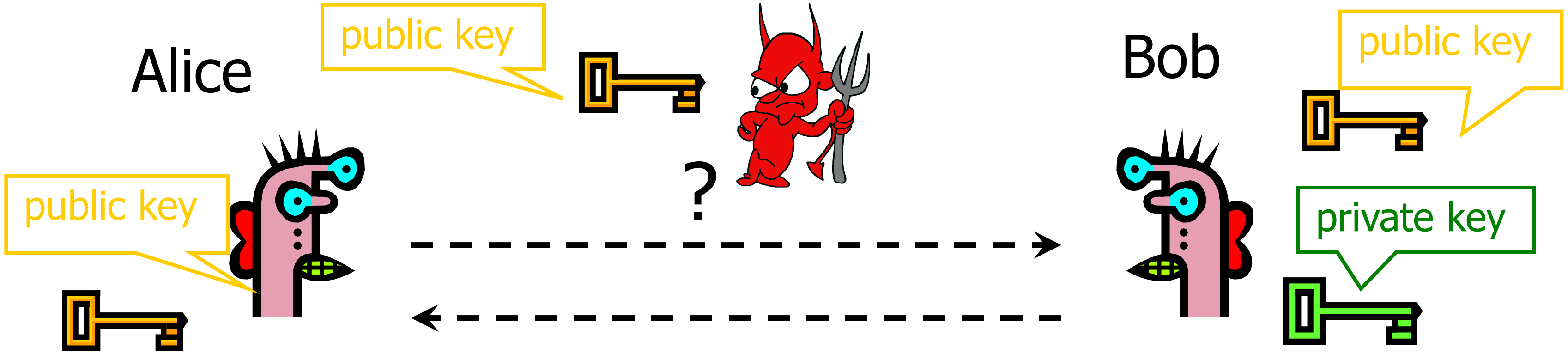
Generic Procedures of ECC

- Both parties agree to some publicly-known data items
  - The **elliptic curve equation**
    - Type, values of ***a*** and ***b*** (or others)
    - Modulo: prime ***p or f(x)***
  - A **base point**, $\boldsymbol{G}$**,** taken from the elliptic group;
  - Others parameters (assure security)
- Each user generates their public/private key pair
  - Private Key: $d \in [1, p-1]$
  - Public Key: $Q = d.G = \underbrace{G + G + \cdots + G}_{d \text{ times}}$

ECC Cipher

- Suppose Alice wants to send to Bob an encrypted message.
  - Both agree on a ECC curver and a base point $\boldsymbol{G}$.
  - Alice and Bob create public/private keys.
    - Alice
      - Private Key = a
      - Public Key = $Q_A = a\,.\boldsymbol{G}$
    - Bob
      - Private Key = b
      - Public Key = $Q_B = b.\boldsymbol{G}$

ECC Cipher



$$(G, PK_B = bG, SK = b)$$

- Input: $M \in E/K$

- Select a random integer $k \in [1, p-1]$, compute

- **Encrypt**

$$R = kG, C = M + k\,.\,PK_B \xrightarrow{(R, C)}$$

- **Decrypt**

$$C - b\,R$$
$$= M + kb\,G - b\,kG$$
$$= M$$

ECC Diffie-Hellman (ECDHE)

- **Public:** Elliptic curve and a point ***G*** =(x,y) on curve
- **Secret:** Alice's a and Bob's b

(a, Q_A)

(b, Q_B)

Alice, A

$Q_A(= a.G)$ →

← $Q_B(= b.G)$

Bob, B

- Alice computes $K_A = a.Q_B = ab.G$
- Bob computes $K_B = b.Q_A = ba.G$
- These are the same since $K_A = K_B = ab\ G$

Diffie-Hellman key exchange attack

$g^a \bmod p \longrightarrow$

(a, g^a)

(b, g^b)

$\longleftarrow g^b \bmod p$

Bob

Alice

$$sk = (g^b)^a = (g^a)^b = g^{ab} \bmod p$$

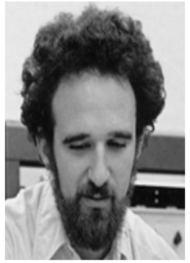
man-in-the-middle attack!

$g^a \bmod p \longrightarrow$

$\longleftarrow g^m \bmod p$

$g^m \bmod p \longrightarrow$

$\longleftarrow g^b \bmod p$



$$sk_1 = g^{am} \bmod p$$

$$sk_2 = g^{bm} \bmod p$$

Diffie-Hellman key exchange attack

$\boldsymbol{Q_A}$

$(a, Q_A = aG)$

$\boldsymbol{Q_B}$

$(b, Q_B = bG)$

Bob

$sk = aQ_B = bQ_A = abG$

Alice

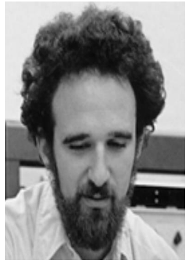
man-in-the-middle attack!

aG

mG

mG

bG



$(a, Q_A = aG)$

$sk_1 = amG$

$(m, Q_M = mG)$

$sk_2 = bmG$

$(b, Q_B = bG)$

Why use ECC?

- How do we analyze Cryptosystems?
  - How difficult is the underlying problem that it is based upon
    - RSA – Integer Factorization
    - DH – Discrete Logarithms
    - ECC - Elliptic Curve Discrete Logarithm problem
  - How do we measure difficulty?
    - We examine the algorithms used to solve these problems

Applications of ECC

- Many devices are small and have limited storage and computational power
- Where can we apply ECC?
  - **Wireless communication devices**
  - Smart cards
  - Web servers that need to handle many encryption sessions
  - **Any application where security is needed but lacks the power, storage and computational power that is necessary for our current cryptosystems**

Benefits of ECC

- Same benefits of the other cryptosystems: confidentiality, integrity, authentication and non-repudiation but…
- Shorter key lengths
  - Encryption, Decryption and Signature Verification speed up
  - Storage and bandwidth savings

Security of ECC

- To **protect** a 128 bit AES key it would take a:
  - RSA Key Size: 3072 bits
  - ECC Key Size: 256 bits
- How do we strengthen RSA?
  - Increase the key length
- **Impractical?**

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

Symmetric Encryption (Key Size in bits)	RSA and Diffie-Hellman (modulus size in bits)	ECC Key Size in bits	
56	512	112	
80	1024	160	
112	2048	224	
128	3072	256	3
192	7680	384	4
256	15360	512	5

Summary of ECC

- "**Hard problem**" analogous to discrete log
  - **Q=kG, where Q,G belong to a prime curve**

    given k,G → "easy" to compute Q

    given Q,G → "hard" to find k
  - **known as the elliptic curve logarithm problem**
    - **k must be large enough**
- ECC security relies on elliptic curve logarithm problem
  - compared to factoring, can use much smaller key sizes than with RSA etc

    → **for similar security ECC offers significant computational advantages**