Cryptography

Lecture 4: Elliptic Curves II

Dr. Muhammad Hataba
Muhammad.Hataba@giu-uni.de

References and Legalities

This lecture makes use of the following resources:

- Understanding Cryptography Chapter 9
- https://www.youtube.com/watch?v=F3zzNa42-tQ
- Information Security Course, German International University, Amr ElMougy
- Cryptography Course, German International University, *Alia El Bolock*

Elliptic Curve Discrete Log Problem

Recall: We need a trapdoor function!

Elliptic Curves over Prime Fields $\mathbb{Z}_p$

- In cryptography, we are interested in elliptic curves modulo a prime p

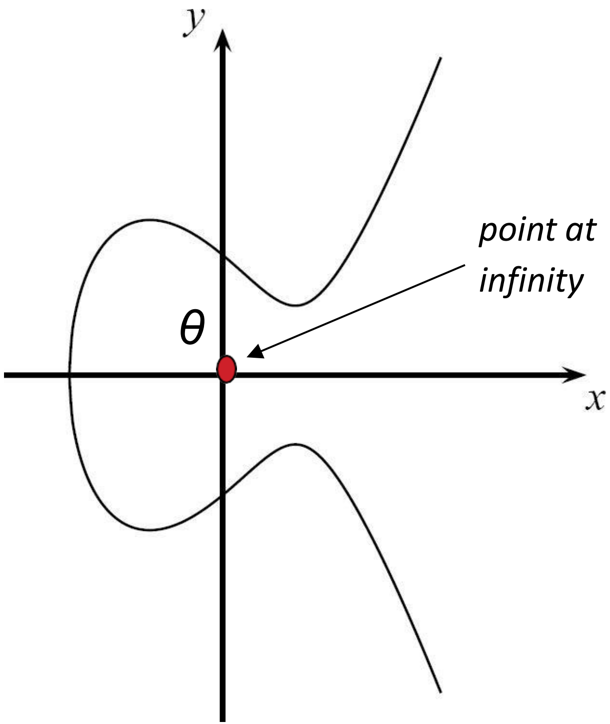
Definition: Elliptic Curves over prime fields

The elliptic curve over $\mathbb{Z}_p$, $p>3$ is the set of all pairs $(x,y) \in \mathbb{Z}_p$ which fulfill

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

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

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



Note;

$\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ is a set of integers with modulo p arithmetic

Solutions still forms an Abelian Group

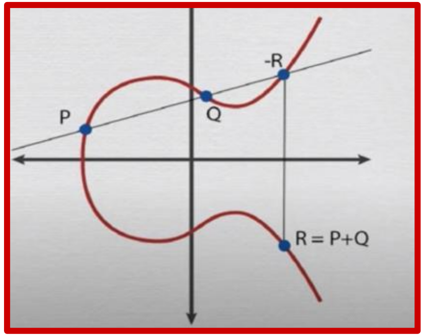
Point Addition and Doubling Formulas in $\mathbb{Z}_p$

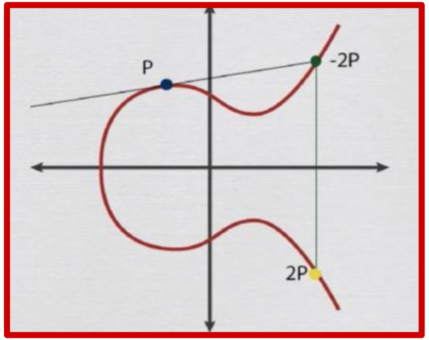
$$P+Q=R \text{ where } P=(x_1,y_1),\ Q=(x_2,y_2), \text{ and } R=(x_3,y_3)$$
$$\Rightarrow (x_1,y_1)+(x_2,y_2)=(x_3,y_3)$$

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

where

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





Elliptic Curve Discrete Log Problem

Given Elliptic Curves E over $\mathbb{Z}/p\mathbb{Z}$

- Scalar multiplication on Elliptic Curves E is a one way function, i.e., a trapdoor

$\Rightarrow$ **Discrete Log Problem:**

- Given P and $Q \in \mathbb{Z}/p\mathbb{Z}$, where Q is a multiple of P
- Find k such that $Q = kP$

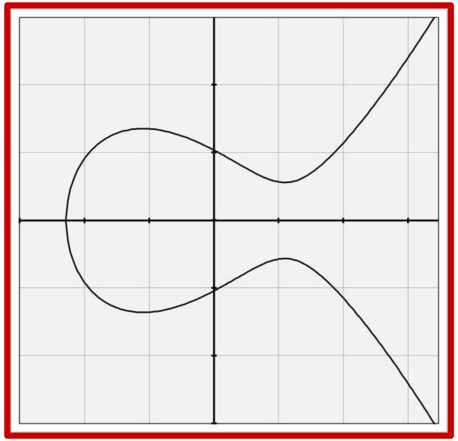
This problem is a very hard one!

Elliptic Curve Discrete Log Problem

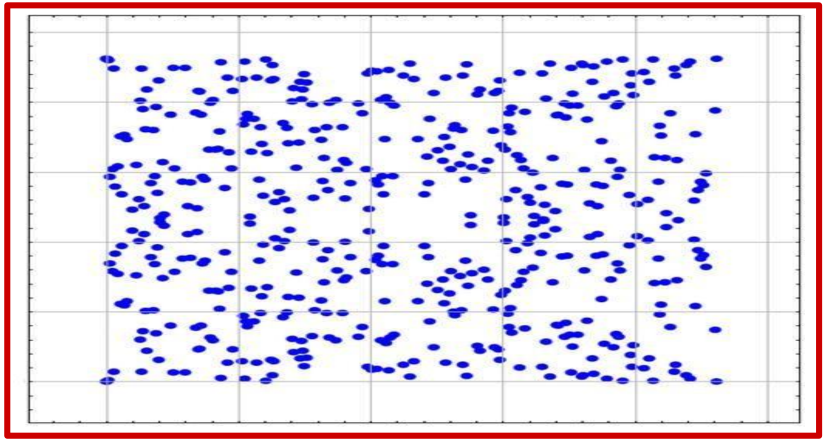
Elliptic Logarithm Problem

Given P and aP, there is no way to compute a, but given P and a, it is easy to compute aP.
This is more difficult than factorization → can use much smaller key sizes than with RSA for the same security level.
Shorter Key means less operations
With similar security, ECC offers significant computational advantages

Hint: Why is the Problem Hard?



Elliptic curve over R



Elliptic curve over Z_p

The Base Point (Generator)

- $G \in \mathbb{Z}/\mathbb{Z}p$ generates a cyclic group
  - i.e., every point in the group can be generated by repeated addition of the generator point
- $n = ord(G)$ is the order of G
  - i.e., number of points in the generated group
  - is the smallest pos. int k such that $kG = O$
- $|E(\mathbb{Z}/p\mathbb{Z})|$ is the number of points on the curve
- $h = |E(\mathbb{Z}/p\mathbb{Z})| / n$ is the cofactor, ideally $h=1$

Example - Point Multiplication Computation

$E: y^2 = x^3 + 2x + 2 \pmod{17}$ $\quad G = (5,1)$

Compute: $2G$

$$s = \frac{3x_G^2 + a}{2y_G} \qquad s \equiv \frac{3(5^2)+2}{2(1)} \equiv 77 \cdot 2^{-1} \equiv 9 \cdot 9 \equiv 13 \pmod{17}$$

$$x_{2G} = s^2 - 2x_G \qquad x_{2G} \equiv 13^2 - 2(5) \equiv 16 - 10 \equiv 6 \pmod{17}$$

$$y_{2G} = s(x_G - x_{2G}) - y_G \qquad y_{2G} \equiv 13(5-6) - 1 \equiv -13 - 1 \equiv -14 \equiv 3 \pmod{17}$$

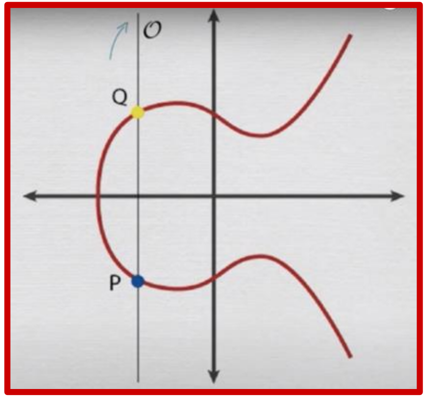
$$2G = (6,3)$$

Example - Full Cyclic Group of E

$$E: \quad y^2 \equiv x^3 + 2x + 2 \pmod{17}$$

$G = (5, 1)$
$2G = (6, 3)$
$3G = (10, 6)$
$4G = (3, 1)$
$5G = (9, 16)$
$6G = (16, 13)$
$7G = (0, 6)$
$8G = (13, 7)$
$9G = (7, 6)$
$10G = (7, 11)$
$11G = (13, 10)$
$12G = (0, 11)$
$13G = (16, 4)$
$14G = (9, 1)$
$15G = (3, 16)$
$16G = (10, 11)$
$17G = (6, 14)$
$18G = (5, 16)$
$19G = \mathcal{O}$

n=19 and h=1



Elliptic Curve cryptography (ECC) Security

Consider an elliptic curve

$$E : y^2 = x^3 + x + 6 \ \text{ over } \ GF(11)$$

To find all points (x, y) of E for each $x \in GF(11)$, compute $z = x^3 + x + 6 \mod 11$ and determine whether z is a **quadratic residue** (QR). If so, solve $y^2 = z$ in $GF(11)$. We can find there are totally 13 points on this curve.

x	$x^3 + x + 6$	QR?	y
0	6	no	–
1	8	no	–
2	5	yes	4, 7
3	3	yes	5, 6
4	8	no	–
5	4	yes	2, 9
6	8	no	–
7	4	yes	2, 9
8	9	yes	3, 8
9	7	no	–
10	4	yes	2, 9

An integer q is called a quadratic residue modulo n if there exists an integer x such that: $x^2 \equiv q \pmod{n}$.

GIU GERMAN INTERNATIONAL UNIVERSITY

Elliptic Curve cryptography (ECC) Security

Example (continued)

There are 13 points in the group.

So, it is **cyclic** and **any point** other $\mathcal{O}$ is generator.

Let $P = (2, 7)$. We can compute $2P = (x_2, y_2)$ as follows.

$$
\begin{aligned}
\lambda &= \frac{3x_1^2 + a}{2y_1} = \frac{3 \cdot 2^2 + 1}{2 \cdot 7} = \frac{13}{14} = 2 \cdot 3^{-1} = 2 \cdot 4 = 8 \pmod{11} \\
x_2 &= \lambda^2 - 2x_1 = 8^2 - 2 \cdot 2 = 5 \pmod{11} \\
y_2 &= (x_1 - x_2)\lambda - y_1 = (2 - 5) \cdot 8 - 7 = 2 \pmod{11}
\end{aligned}
$$

Therefore, we obtain $2P = (5, 2)$.

Elliptic Curve cryptography (ECC) Security

Example (continued)

Let $3P = P + 2P = (x_3, y_3)$. Then we can compute $3P$ as follows.

$$\begin{aligned}
\lambda &= \frac{y_2 - y_1}{x_2 - x_1} = \frac{2-7}{5-2} = 2 \pmod{11} \\
x_3 &= \lambda^2 - x_1 - x_2 = 2^2 - 2 - 5 = 8 \pmod{11} \\
y_3 &= (x_1 - x_3)\lambda - y_1 = (2-8) \cdot 2 - 7 = 3 \pmod{11}
\end{aligned}$$

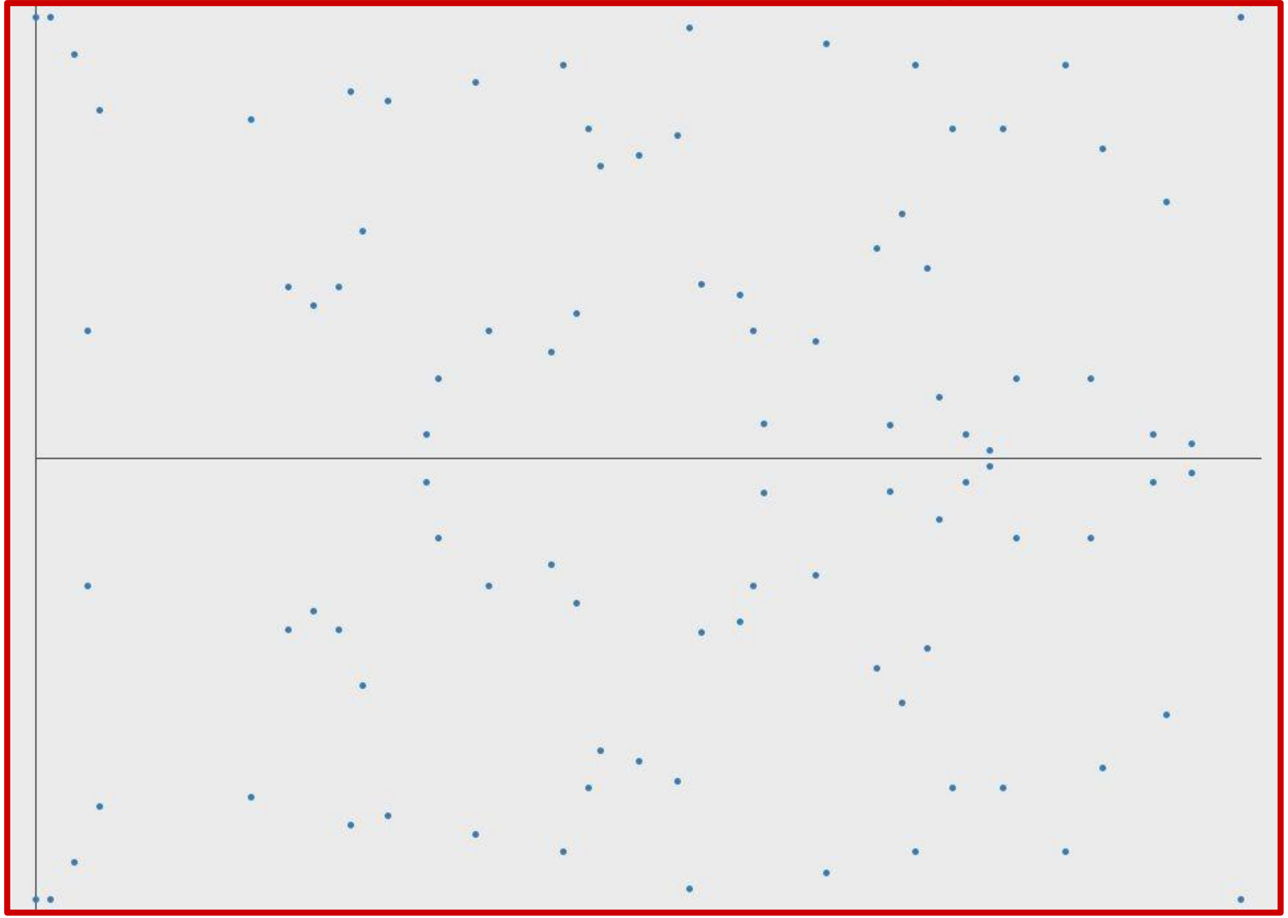
Hence, we obtain $3P = (8, 3)$.

Similarly, we can also compute the cyclic group generated by P.

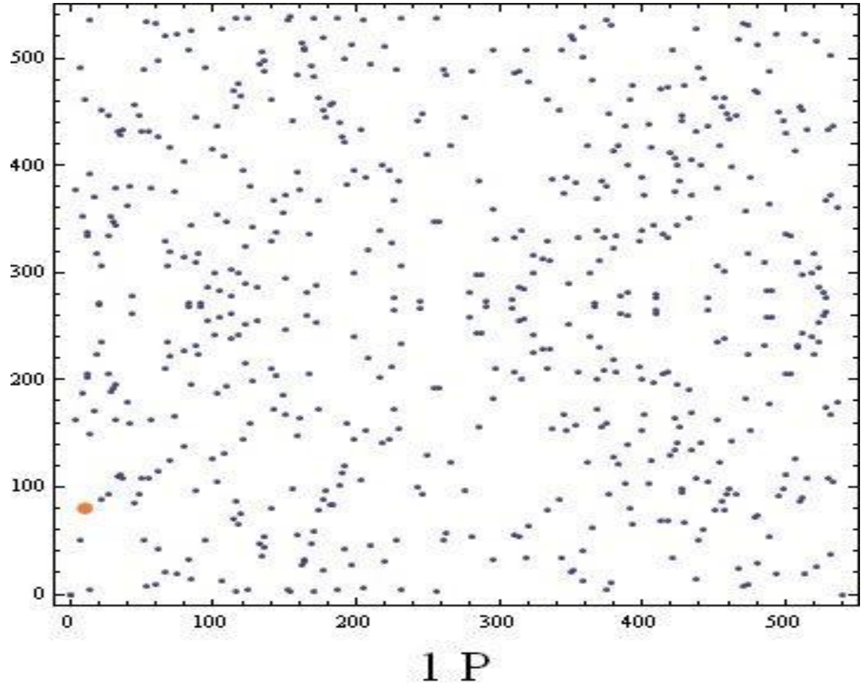
$$\begin{array}{cccc}
P = (2,7) & 2P = (5,2) & 3P = (8,3) & 4P = (10,2) \\
5P = (3,6) & 6P = (7,9) & 7P = (7,2) & 8P = (3,5) \\
9P = (10,9) & 10P = (8,8) & 11P = (5,9) & 12P = (2,4)
\end{array}$$

$$13P = P + 12P = 2P + 11P = 3P + 10P = \cdots = \mathcal{O}$$

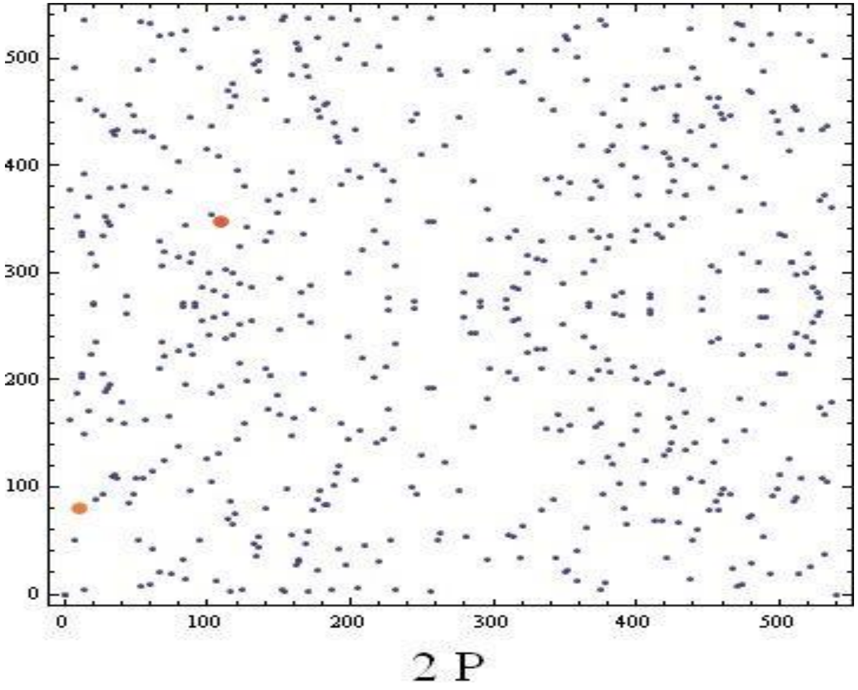
Elliptic Curves on $\mathbb{Z}_p$



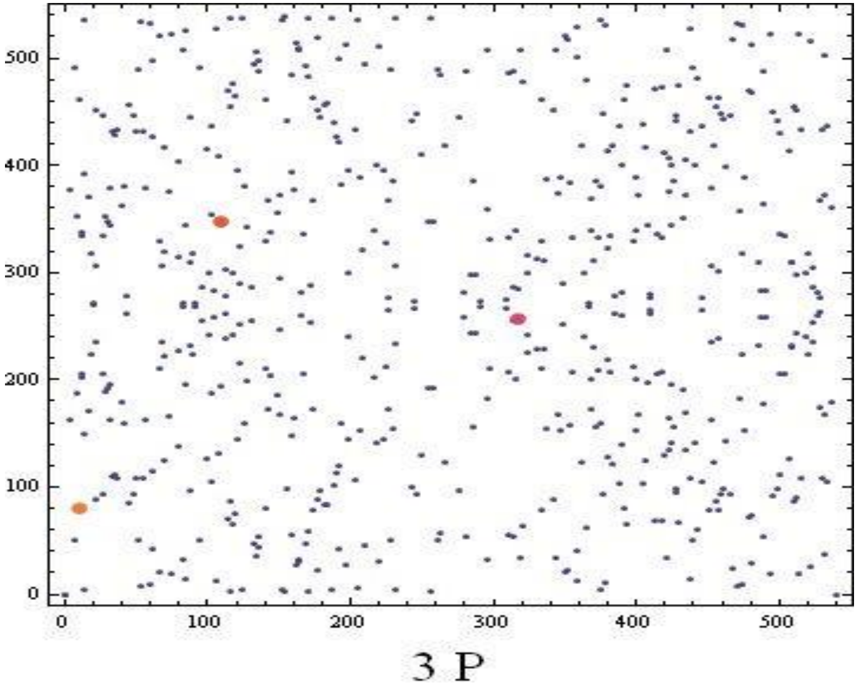
Example mod 541



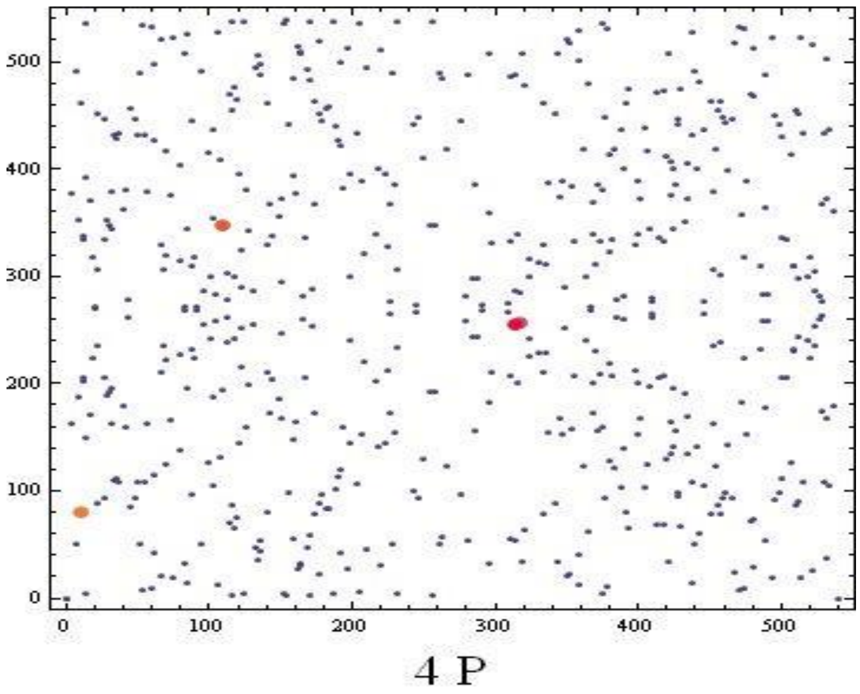
Example mod 541



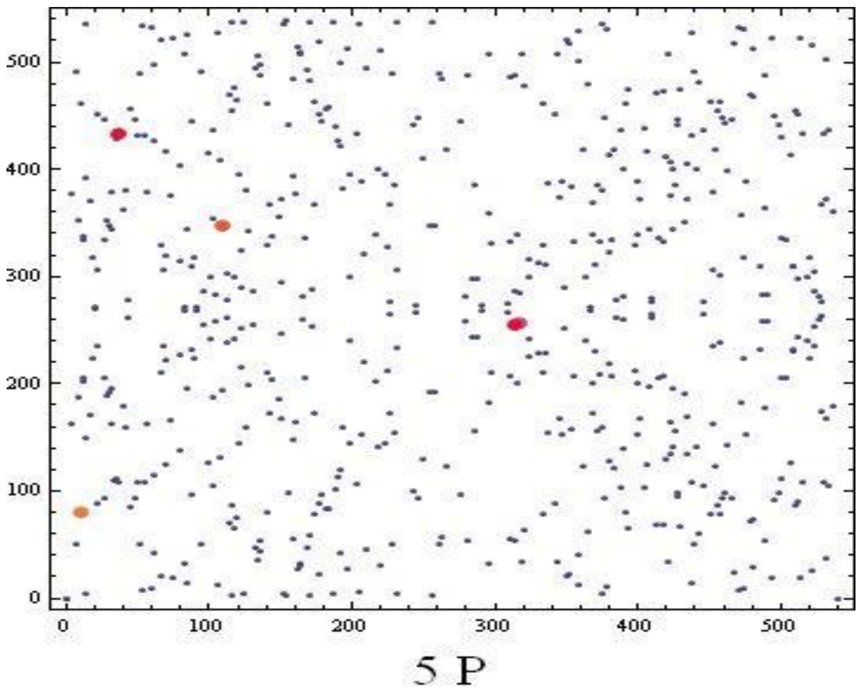
Example mod 541



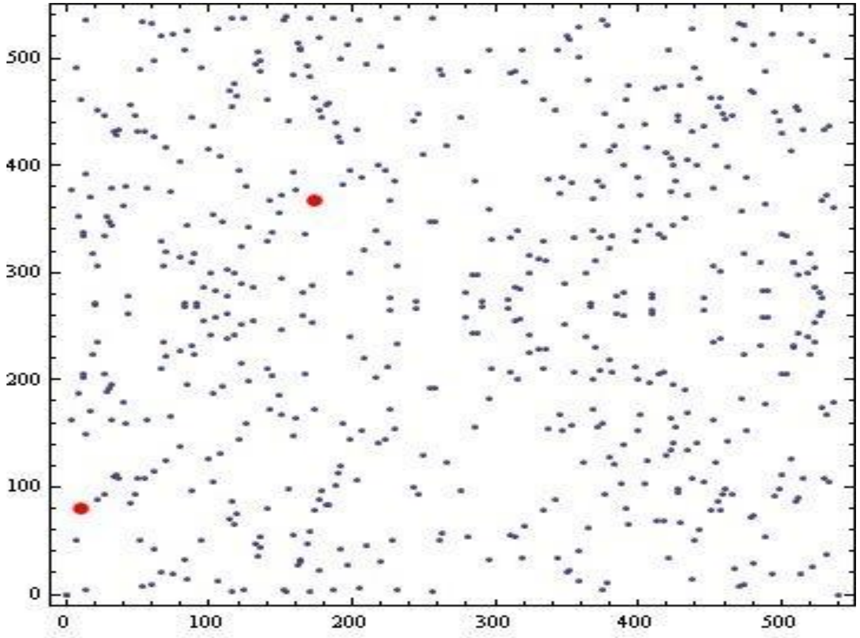
Example mod 541



Example mod 541



Elliptic Curve Discrete Log Problem



$$(173, 368) = m\,(10, 80)$$

Elliptic Curve Cryptography

ECC Algorithms

- Every user has a public and a private key.
  - 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

Domain Parameters

{p,a,b,G,n,h}

- p: field (mod p)
- a,b: curve parameters

⇒ Curve

- G: generator point
- n: ord(G)
- h: cofactor

Generic Procedures of ECC

- Both parties agree to some publicly-known data items
  - The **<u>elliptic curve equation</u>**
    - values of ***a*** and ***b***
    - prime, ***p***
  - The **<u>elliptic group</u>** computed from the elliptic curve equation
  - A **<u>base point or generator</u>**, G, taken from the elliptic group
    - Similar to the generator used in other current cryptosystems
- Each user generates their public/private key pair

$$Q = nG = \underbrace{G + G + \ldots + G}_{\text{n times}}$$

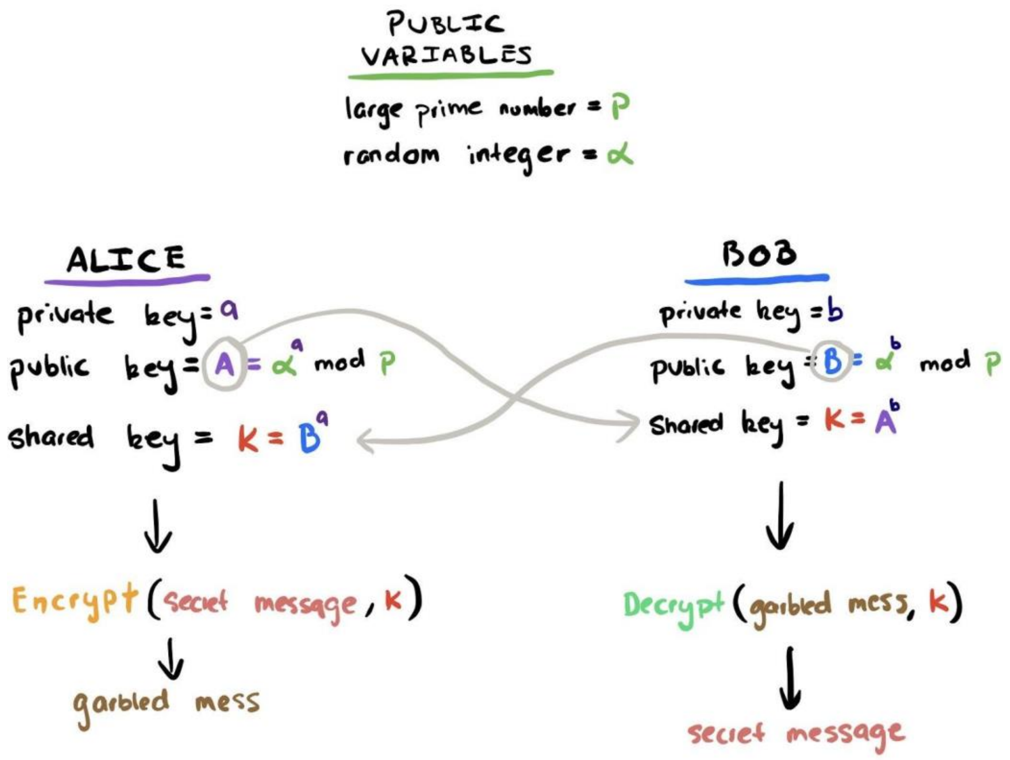
Private Key = integer, n, selected from the interval [1,p-1]

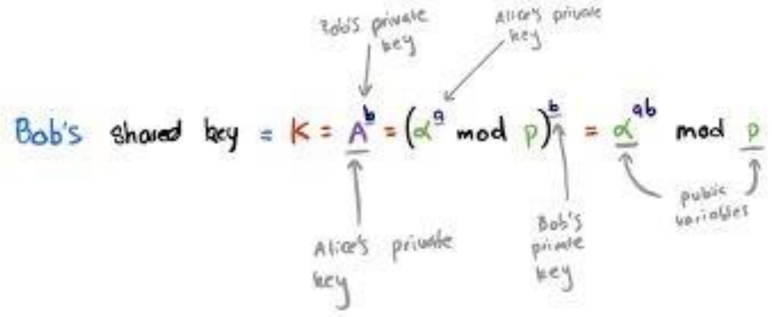
Public Key = product, Q, of private key and base point G

Elliptic Curve Cryptography

Diffie-Hellmann Key Exchange

Recall: Diffie-Hellman Key Exchange





EC Diffie-Hellman Key Exchange

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

a(x,y) →

← b(x,y)

Alice, A

Bob, B

- Alice computes a(b(x,y))
- Bob computes b(a(x,y))
- These are the same since ab=ba

EC Diffie-Hellman Key Exchange

Bob

Bob picks private key β

$1 \leq \beta \leq n-1$

Computes

$B = \beta G$

Receives

$A = (x_A, y_A)$

Computes

$P = \beta\alpha G$

Eve

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

p

a

b

G

n

h

A

B

Alice

Alice picks private key α

$1 \leq \alpha \leq n-1$

Computes

$A = \alpha G$

Receives

$B = (x_B, y_B)$

Computes

$P = \alpha\beta G$

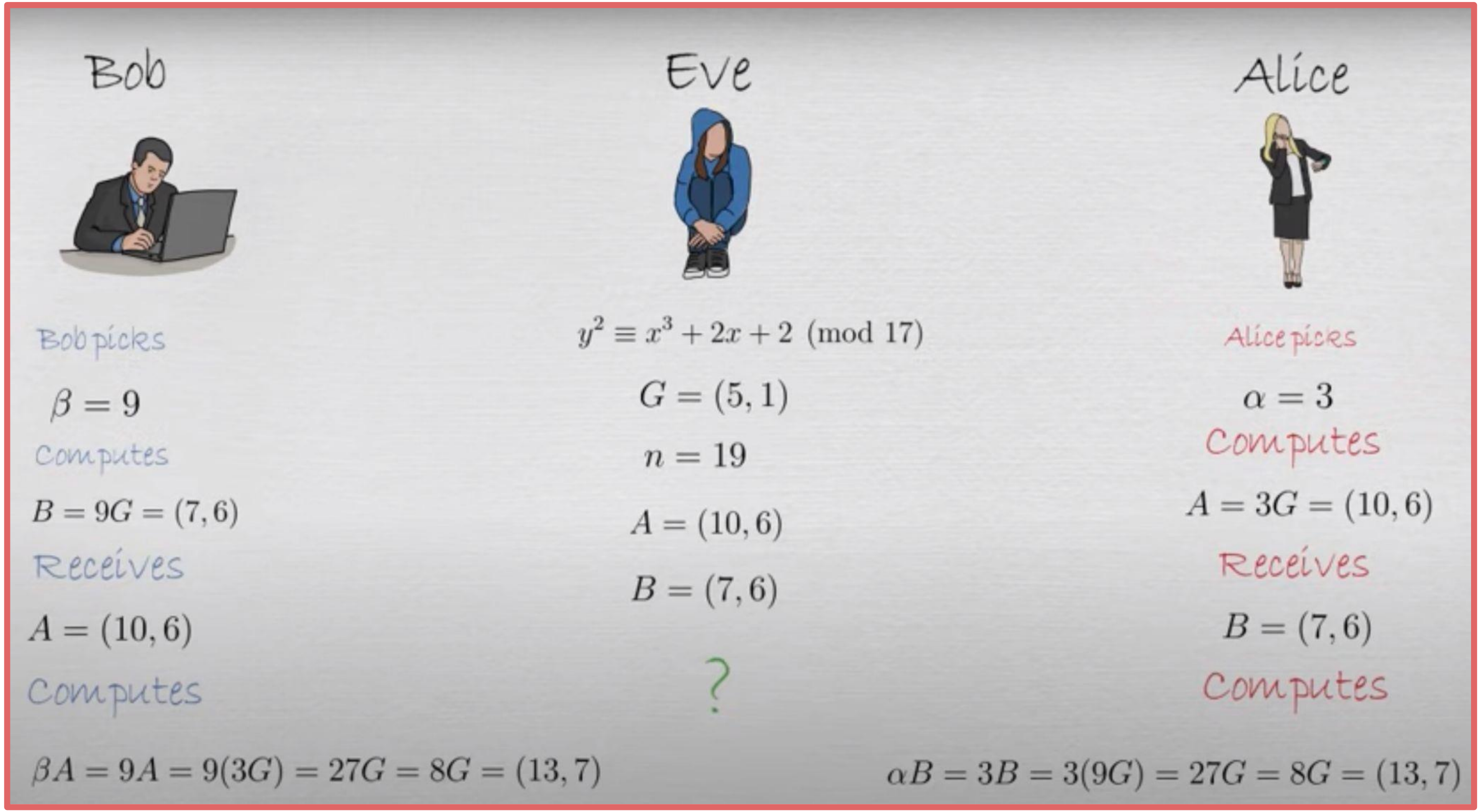
Recall: Example - Full Cyclic Group of E

$$E: \; y^2 \equiv x^3 + 2x + 2 \pmod{17}$$

$G = (5,1)$
$2G = (6,3)$
$3G = (10,6)$
$4G = (3,1)$
$5G = (9,16)$
$6G = (16,13)$
$7G = (0,6)$
$8G = (13,7)$
$9G = (7,6)$
$10G = (7,11)$
$11G = (13,10)$
$12G = (0,11)$
$13G = (16,4)$
$14G = (9,1)$
$15G = (3,16)$
$16G = (10,11)$
$17G = (6,14)$
$18G = (5,16)$
$19G = \mathcal{O}$

n=19 and h=1

Example - EC Diffie-Hellman Key Exchange



EC Diffie-Hellman Key Exchange

Alice and Bob want to agree on a shared key

- Alice and Bob compute their public and private keys.
  - Alice
    - Private Key = a
    - Public Key = P_A = a* G
  - Bob
    - Private Key = b
    - Public Key = P_B = b* G
- Alice and Bob send each other their public keys.
- Both take the product of their private key and the other user's public key.
  - Alice $\longrightarrow K_{AB} = a(bG)$
  - Bob $\longrightarrow K_{AB} = b(aG)$
  - **Shared Secret Key =** $K_{AB} = abG$

Elliptic Curve Cryptography

Encryption and Decryption

ECC Encryption

Suppose Alice wants to send to Bob an encrypted message

- Both agree on a base point G.
- Alice and Bob create public/private key pairs.
  - Alice
    - Private Key = a
    - Public Key = P_A =a* G
  - Bob
    - Private Key = b
    - Public Key = P_B =b* G
- Alice takes plaintext message, M, and encodes it onto a point, P_M, from the elliptic group

ECC Encryption

- Alice chooses another random integer, k from the interval $[1, p-1]$
- The ciphertext is a pair of points

$$P_C = [\ (kG),\ (P_M + kP_B)\ ]$$

ECC Decryption

- To decrypt, Bob computes the product of the first point from P_C and his private key, b
  - $b * (kG)$
- Bob then takes this product and subtracts it from the second point from P_C
  - $(P_M + kP_B) - [b(kG)] = P_M + k(bG) - b(kG) = P_M$
- Bob then decodes P_M to get the message, M.

ECC vs ElGamal (DH-based)

ECC

- The ciphertext is a pair of points
  - $P_C = [(kB), (P_M + kP_B)]$
- Bob takes this product and subtracts it from the second point from P_C
  - $(P_M + kP_B) - [b(kB)]$
    $= P_M + k(bB) - b(kB) = P_M$

ElGamal

- The ciphertext is a pair of numbers
  - $C = (g^k \bmod p, mP_B^k \bmod p)$
- Bob takes the quotient of the second value and the first value raised to Bob's private value
  - $m = mP_B^k \;/\; (g^k)^b$
    $= mg^{k*b} / g^{k*b} = m$

Elliptic Curve Cryptography

Digital Signature Algorithm (ECDSA)

ECDSA - Signature Generation

- Alice has a private key d and her public key is Q= dG
- Once we have the domain parameters (the generator G, the base p) and have decided on the keys to be used, the signature is generated by the following steps:

1. Alice generate a random number k, such that $1 \leq k \leq p-1$
2. kG = (x_1, y_1) is computed. x_1 is converted to its corresponding integer x_1'
3. Next, $r = x_1 \bmod n$ is computed
4. We then compute $k^{-1} \bmod p$
5. $e = HASH(m)$ where m is the message to be signed
6. $s = k^{-1}(e + dr) \bmod p$

$\Rightarrow$ We get the signature as (r, s)

GIU GERMAN INTERNATIONAL UNIVERSITY

ECDSA - Signature Verification

At the receiver's end the signature is verified as follows:

1. Verify whether r and s belong to the interval $[1, p-1]$ for the signature to be valid.
2. Compute $e = HASH(m)$. The hash function should be the same as the one used for signature generation.
3. Compute $w = s^{-1} \bmod p$.
4. Compute $u_1 = ew \bmod p$ and $u_2 = rw \bmod p$.
5. Compute $(x_1, y_1) = u_1 G + u_2 Q$
6. The signature is valid if $r = x_1 \bmod p$, invalid otherwise.

This is how we know that the verification works the way we want it to:

We have, $s = k^{-1}(e + dr) \bmod p$ which we can rearrange to obtain, $k = s^{-1}(e + dr)$ which is $s^{-1}e + s^{-1}rd$

This is nothing but $we + wrd = (u_1 + u_2 d) \pmod p$

We have $u_1 G + u_2 Q = (u_1 + u_2 d)G = kG$ which translates to $v = r$.

Other Considerations

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

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?**

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

Summary of ECC

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

    given k, P: "easy" to compute Q

    given Q, P: "hard" to find k
  - known as the elliptic curve logarithm problem
    - k must be large enough
- ECC security relies on the elliptic curve logarithm problem
  - compared to factoring, we can use much smaller key sizes
  - **for similar security ECC offers significant computational advantages**

ECC vs RSA vs DSA

System Comparison

Attribute	Elliptic Curve (ECC)	DSA	RSA
Key Size (Current use)	160 bits	1024 bits	1024 bits
Encryption/Decryption	-Encryption takes approximately twice as long as decryption -Some message expansion	-Encryption not possible	-Decryption much longer than encryption -No message expansion
Digital Signatures	-Signature generation fast -Can be implemented in low power/compact environment -Very compact signatures (1/5 size of RSA) -Signature appended to message	-Signature generation fast -Signatures longer than RSA -Signatures appended to message	-Signing much longer than verification -Same hardware/software required for signing and verification -Message recovered from signature
Hardware	-Very fast, efficient and small implementations -Suitable for low power applications	-Fast hardware implementations although much larger than Elliptic Curve	-Difficult to build fast hardware

Real example from the NSA

- **Curve P-192**

$p = 62771017353866807638578942320766641608390870039024961279$

$r = 627710173538668076385789423176059013767194773182842284081$

$a = 3099d2bb\ bfcb2538\ 542dcd5f\ b078b6ef\ 5f3d6fe2\ c745de65$

$b = 64210519\ e59c80e7\ 0fa7e9ab\ 72243049\ feb8deec\ c146b9b1$

$G_x = 188da89e\ b03090f6\ 7cbf20eb\ 43a18800\ f4ff0afd\ 82ff1012$

$G_y = 07192b95\ ffc8da78\ 631011ed\ 6b24cdd5\ 73f977a1\ 1e794811$

Pros

- Shorter Key Length

  Same level of security as RSA achieved at a much shorter key length

- Better Security

  Secure because of the ECDLP

  Higher security per key-bit than RSA

- Higher Performance

  Shorter key-length ensures lesser power requirement – suitable in wireless sensor applications and low power devices

  More computation per bit but overall lesser computational expense or complexity due to lesser number of key bits

Cons

- Relatively newer field

  Idea prevails that all the aspects of the topic may not have been explored yet – possibly unknown vulnerabilities

  Doesn't have widespread usage

- Not perfect

  Attacks still exist that can solve ECC (112 bit key length has been publicly broken)

  Well known attacks are the Pollard's Rho attack (complexity $O(\sqrt{n})$), Pohlig's attack, Baby Step, Giant Step, etc.

Using Elliptic Curves In Cryptography

- The central part of any cryptosystem involving elliptic curves is the **<u>elliptic group</u>**.
- All public-key cryptosystems have some underlying mathematical operation.
  - RSA has exponentiation (raising the message or ciphertext to the public or private values)
  - ECC has point multiplication (repeated addition of two points).

Crypto’s Dirty Secret

- Every form of public key cryptography or key exchange relies on our inability to solve a certain math problem quickly (factoring, DLP, ECDLP, SVP, etc).
- It is still possible that these “hard math problems” have quick solutions. All we know is that no one has found a quick solution yet (or at least has admitted to this publicly).
- Research Problem: Find a quick solution to the ECDLP (thus making ECC useless) OR prove that no quick solution exists (thus making every other form of crypto useless).

We Learned...

- The basic pros and cons of ECC vs. RSA and DL schemes
- What an elliptic curve is and how to compute with it
- How to build a DL problem with an elliptic curve
- Protocols that can be realized with elliptic curves
- Current security estimations of cryptosystems based on elliptic curves