COM307000 - Cryptography
Public Key Crypto_part 2:

Dr. Anca Jurcut
E-mail: `anca.jurcut@ucd.ie`

School of Computer Science and Informatics
University College Dublin,
Ireland

Public Key Algorithms

- ✓ **Knapsack**
- ❑ **RSA**
- ❑ **Diffie Hellman**
- ❑ **Elliptic Curve based Crypto (ECC)**

Others: Elgamal, Rabin, Goldwasser-Micali (probanilistic),Blum-Goldwasser(probalistic), Schnorr signature, Zero-Knowledge Algorithms (Fiat-Shamir, Ohta-Okamoto,…)

RSA

RSA

- Invented by Clifford Cocks (GCHQ) and **R**ivest, **S**hamir, and **A**dleman (MIT)
  - RSA is the ***gold standard*** in public key crypto
- Let **p** and **q** be two large prime numbers
- Let **N = pq** be the **modulus**
- Choose **e** relatively prime to (p−1)(q−1)
- Find **d** such that ed = 1 mod (p−1)(q−1)
- **Public key** is **(N,e)**
- **Private key** is **d**

RSA

- Message M is treated as a number
- **To encrypt M we compute**

  $C = M^e \bmod N$
- **To decrypt ciphertext C compute**

  $M = C^d \bmod N$
- Recall that e and N are public
- If Trudy can factor $N = pq$, she can use e to easily find d since $ed = 1 \bmod (p-1)(q-1)$
- So, **factoring the modulus breaks RSA**
  - Is factoring the only way to break RSA?

Does RSA Really Work?

- Given $C = M^e \bmod N$ we want to show that $M = C^d \bmod N = M^{ed} \bmod N$
- We'll need **Euler's Theorem:**
  If x is relatively prime to n then $x^{\varphi(n)} = 1 \bmod n$
- Facts:
  1) $ed = 1 \bmod (p-1)(q-1)$
  2) By definition of "mod", $ed = k(p-1)(q-1) + 1$
  3) $\varphi(N) = (p-1)(q-1)$
- Then $ed - 1 = k(p-1)(q-1) = k\varphi(N)$
- So, $\mathbf{M^{ed}} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\varphi(N)}$
  $= M \cdot (M^{\varphi(N)})^k \bmod N = M \cdot 1^k \bmod N = \mathbf{M \bmod N}$

Simple RSA Example

- Example of ***textbook*** RSA
  - Select "large" primes $p = 11$, $q = 3$
  - Then $N = pq = 33$ and $(p - 1)(q - 1) = 20$
  - Choose $e = 3$ (relatively prime to 20)
  - Find d such that $ed = 1 \bmod 20$
    - We find that $d = 7$ works
- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$

Simple RSA Example

- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$
- Suppose message to encrypt is $M = 8$
- Ciphertext C is computed as

  $C = M^e \bmod N = 8^3 = 512 = 17 \bmod 33$
- Decrypt C to recover the message M by

  $M = C^d \bmod N = 17^7 = 410{,}338{,}673$

  $= 12{,}434{,}505 * 33 + 8 = 8 \bmod 33$

More Efficient RSA (1)

- Modular exponentiation example
  - $5^{20} = 95367431640625 = 25 \bmod 35$
- A better way: **repeated squaring**
  - 20 = 10100 base 2
  - (1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)
  - Note that $2 = 1 \cdot 2$, $5 = 2 \cdot 2 + 1$, $10 = 2 \cdot 5$, $20 = 2 \cdot 10$
  - $5^1 = 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 = 25 \bmod 35$
  - $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 = 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 = 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 = 25 \bmod 35$
- No huge numbers and it's efficient!

More Efficient RSA (2)

- Use **e = 3** for all users (but not same N or d)
  - + Public key operations only require 2 multiplies
  - o Private key operations remain expensive
  - - If $M < N^{1/3}$ then $C = M^e = M^3$ and **cube root attack**
  - - For any M, if C_1, C_2, C_3 sent to 3 users, cube root attack works (uses Chinese Remainder Theorem)
- Can prevent cube root attack by padding message with random bits
- Note: $e = 2^{16} + 1$ also used ("better" than e = 3)

Diffie-Hellman

Diffie-Hellman Key Exchange

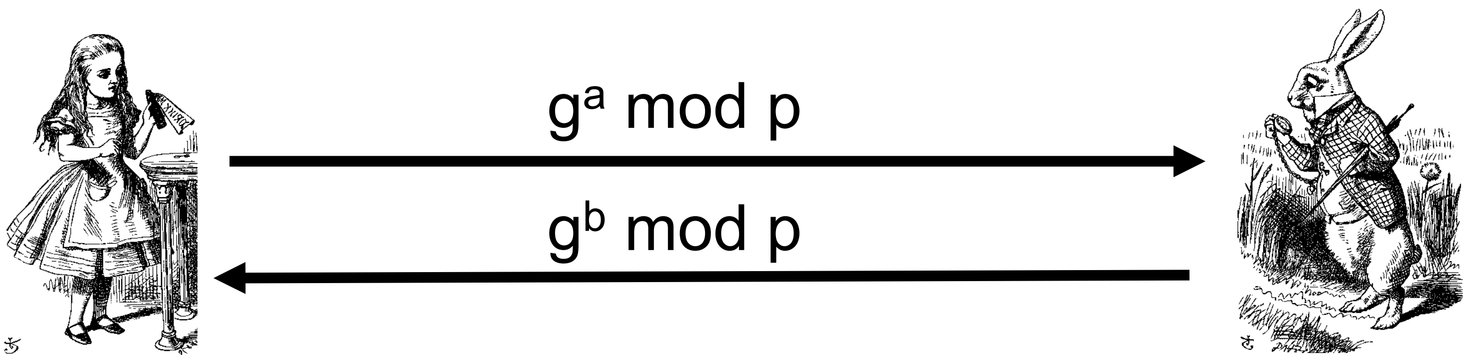
- Invented by Williamson (GCHQ) and, independently, by D and H (Stanford)
- A "key exchange" algorithm
  - Used to establish a shared symmetric key
  - ***Not*** for encrypting or signing
- Based on **discrete log** problem
  - **Given:** g, p, and g^k mod p
  - **Find:** exponent k

Diffie-Hellman

- Let p be prime, let g be a **generator**
  - For any $x \in \{1,2,\ldots,p-1\}$ there is n s.t. $x = g^n \bmod p$
- Alice selects her private value a
- Bob selects his private value b
- Alice sends $g^a \bmod p$ to Bob
- Bob sends $g^b \bmod p$ to Alice
- Both compute shared secret, $g^{ab} \bmod p$
- Shared secret can be used as symmetric key

Diffie-Hellman

- **Public:** g and p
- **Private:** Alice's exponent a, Bob's exponent b



Alice, a   Bob, b

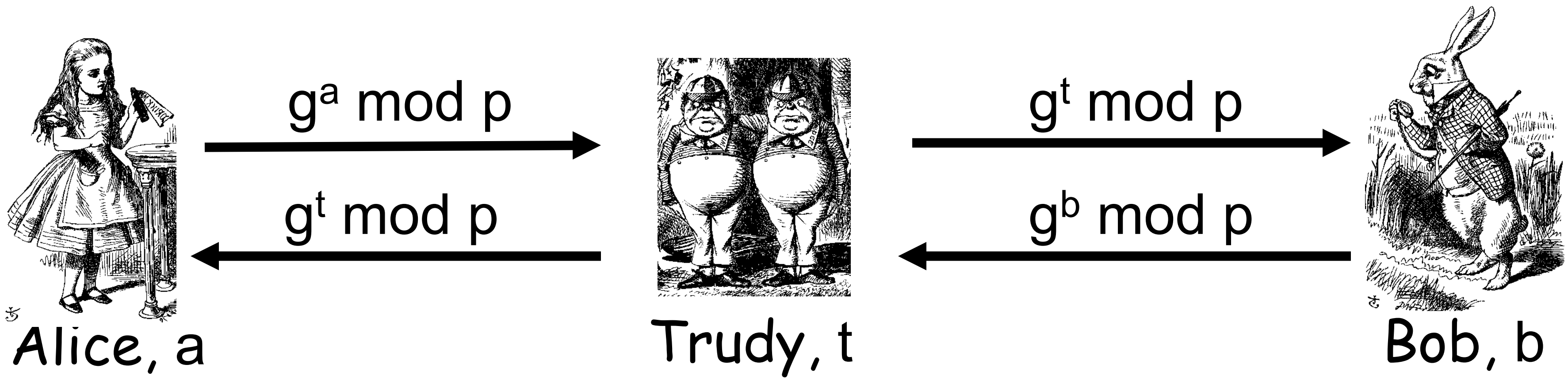
- Alice computes $(g^b)^a = g^{ba} = g^{ab}$ mod p
- Bob computes $(g^a)^b = g^{ab}$ mod p
- They can use **$K = g^{ab}$** mod p as **symmetric key**

Diffie-Hellman

- Suppose Bob and Alice use Diffie-Hellman to determine symmetric key $K = g^{ab} \bmod p$
- Trudy can see $g^a \bmod p$ and $g^b \bmod p$
  - But… $g^a g^b \bmod p = g^{a+b} \bmod p \neq g^{ab} \bmod p$
- If Trudy can find a or b, she gets K
- If Trudy can solve **discrete log** problem, she can find a or b

Diffie-Hellman

- Subject to man-in-the-middle (MiM) attack



- Trudy shares secret g^{at} mod p with Alice
- Trudy shares secret g^{bt} mod p with Bob
- Alice and Bob don't know Trudy is MiM

Diffie-Hellman

- **How to prevent MiM attack?**
  - Encrypt DH exchange with symmetric key
  - Encrypt DH exchange with public key
  - Sign DH values with private key
  - Other?
- **At this point, DH may look pointless…**
  - …but it's not (more on this later)
- You **must** be aware of MiM attack on Diffie-Hellman

Elliptic Curve Cryptography

Elliptic Curve Crypto (ECC)

- "Elliptic curve" is **not** a cryptosystem
- Elliptic curves provide different way to do the math in public key system
- Elliptic curve versions of DH, RSA, …
- Elliptic curves are more efficient
  - Fewer bits needed for same security
  - But the operations are more complex

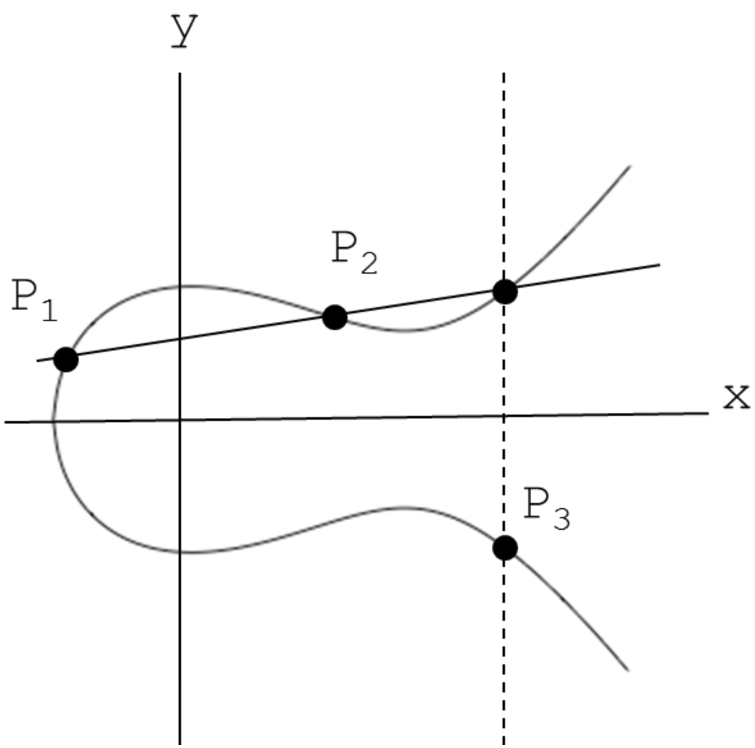
What is an Elliptic Curve?

- An elliptic curve E is the graph of an equation of the form

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

- Also includes a "point at infinity"
- What do elliptic curves look like?
- See the next slide!

Elliptic Curve Picture



- Consider elliptic curve

$$\mathbf{E}: \quad y^2 = x^3 - x + 1$$

- If P_1 and P_2 are on E, we can define

$$P_3 = P_1 + P_2$$

  as shown in picture
- Addition is all we need

Points on Elliptic Curve

- Consider $y^2 = x^3 + 2x + 3 \pmod 5$

  $x = 0 \Rightarrow y^2 = 3 \Rightarrow \text{no solution} \pmod 5$

  $x = 1 \Rightarrow y^2 = 6 = 1 \Rightarrow y = 1,4 \pmod 5$

  $x = 2 \Rightarrow y^2 = 15 = 0 \Rightarrow y = 0 \pmod 5$

  $x = 3 \Rightarrow y^2 = 36 = 1 \Rightarrow y = 1,4 \pmod 5$

  $x = 4 \Rightarrow y^2 = 75 = 0 \Rightarrow y = 0 \pmod 5$

- Then points on the elliptic curve are:

  (1,1) (1,4) (2,0) (3,1) (3,4) (4,0)

  and the point at infinity: ∞

Elliptic Curve Math

- **Addition on:** $y^2 = x^3 + ax + b \pmod p$

  $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$

  $P_1 + P_2 = P_3 = (x_3, y_3)$ **where**

  $x_3 = m^2 - x_1 - x_2 \pmod p$

  $y_3 = m(x_1 - x_3) - y_1 \pmod p$

  And $m = (y_2 - y_1) * (x_2 - x_1)^{-1} \bmod p$, **if** $P_1 \neq P_2$

  $m = (3x_1^2 + a) * (2y_1)^{-1} \bmod p$, **if** $P_1 = P_2$

  Special cases: **If** m **is infinite**, $P_3 = \infty$, **and**

  $\infty + P = P$ **for all** P

Elliptic Curve Addition

- **Consider** $y^2 = x^3 + 2x + 3 \pmod 5$. **Points on the curve are** $(1,1)$ $(1,4)$ $(2,0)$ $(3,1)$ $(3,4)$ $(4,0)$ and ∞
- **What is** $(1,4) + (3,1) = P_3 = (x_3, y_3)$**?**

$$m = (1-4)*(3-1)^{-1} = -3*2^{-1}$$
$$= 2(3) = 6 = 1 \pmod 5$$
$$x_3 = 1 - 1 - 3 = 2 \pmod 5$$
$$y_3 = 1(1-2) - 4 = 0 \pmod 5$$

- **On this curve,** $(1,4) + (3,1) = (2,0)$

ECC Diffie-Hellman

- **Public:** Elliptic curve and point (x,y) on curve
- **Private:** 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

ECC Diffie-Hellman

- **Public: Curve** $y^2 = x^3 + 7x + b \pmod{37}$ and point $(2,5) \Rightarrow b = 3$
- **Alice's private:** $A = 4$
- **Bob's private:** $B = 7$
- **Alice sends Bob:** $4(2,5) = (7,32)$
- **Bob sends Alice:** $7(2,5) = (18,35)$
- **Alice computes:** $4(18,35) = (22,1)$
- **Bob computes:** $7(7,32) = (22,1)$

Uses for Public Key Crypto

Uses for Public Key Crypto

- Confidentiality
  - Transmitting data over insecure channel
  - Secure storage on insecure media
- Authentication protocols (later)
- Digital signature
  - Provides integrity and **non-repudiation**
  - No non-repudiation with symmetric keys

Non-non-repudiation

- Alice orders 100 shares of stock from Bob
- Alice computes **MAC** using symmetric key
- Stock drops, Alice claims she did ***not*** order
- Can Bob prove that Alice placed the order?
- **No!** Bob also knows the symmetric key, so he could have forged the **MAC**
- **Problem:** Bob knows Alice placed the order, but he can't prove it

Non-repudiation

- Alice orders 100 shares of stock from Bob
- Alice **signs** order with her private key
- Stock drops, Alice claims she did not order
- Can Bob prove that Alice placed the order?
- **Yes!** Alice's private key used to sign the order — only Alice knows her private key
- This assumes Alice's private key has not been lost/stolen

Public Key Notation

- **Sign** message M with Alice's **private key**: $\{M\}K_{APriv}$
- **Encrypt** message M with Alice's **public key**: $\{M\}K_{APub}$
- Then

$$\{\{M\}K_{APriv}\}k_{APub} = M$$

$$\{\{M\}K_{APub}\}k_{APriv} = M$$

Sign and Encrypt
vs
Encrypt and Sign

Confidentiality and Non-repudiation?

- Suppose that we want confidentiality and integrity/non-repudiation
- Can public key crypto achieve both?
- Alice sends message to Bob
  - **Sign and encrypt:** $\{\{M\}K_{APriv}\}K_{BPub}$
  - **Encrypt and sign:** $\{\{M\}K_{BPub}\}K_{APriv}$
- Can the order possibly matter?

Sign and Encrypt

- M = "I love you"

Alice

$\{\{M\}K_{APriv}\}K_{BPub}$ →

Bob

$\{\{M\}K_{APriv}\}K_{CPub}$ →

Charlie

- **Q:** What's the problem?
- **A:** No problem — public key is public

Encrypt and Sign

- M = "My theory, which is mine…."

Alice

$\{\{M\}K_{BPub}\}K_{APriv}$ $\longrightarrow$

Charlie

$\{\{M\}K_{BPub}\}K_{CPriv}$ $\longrightarrow$

Bob

- **Note** that Charlie cannot decrypt M
- **Q:** What is the problem?
- **A:** No problem — public key is public

Public Key Infrastructure