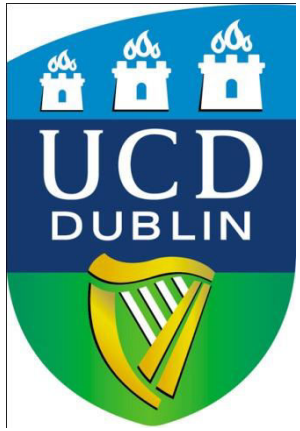
COM307000 - Cryptography
Public Key Crypto_part 3:

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,…)

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

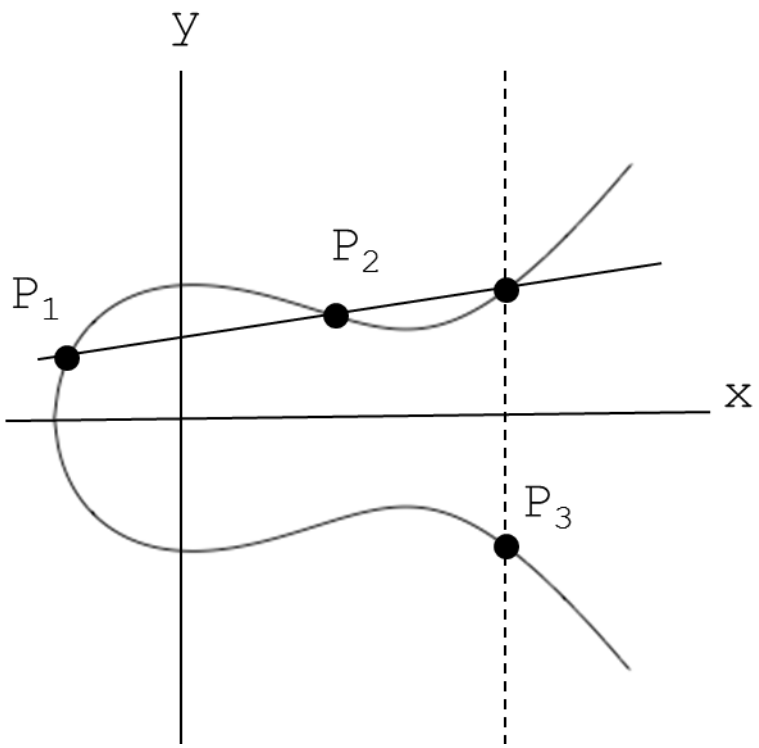
What is an Elliptic Curve?

- 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

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

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

  $x = 0 \Rightarrow y^2 = 3 \Rightarrow$ 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

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

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

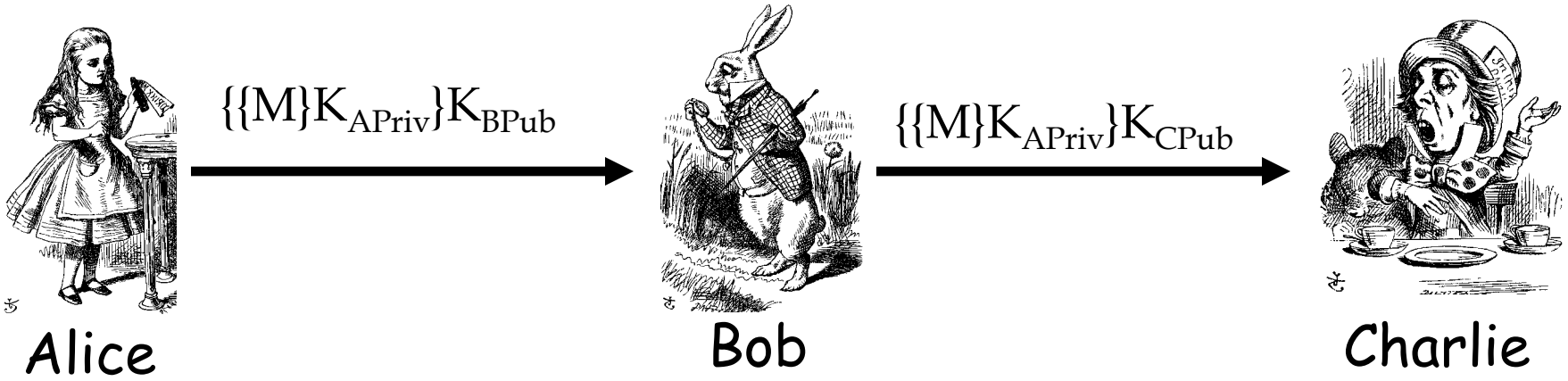
Sign and Encrypt
vs
Encrypt and Sign

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

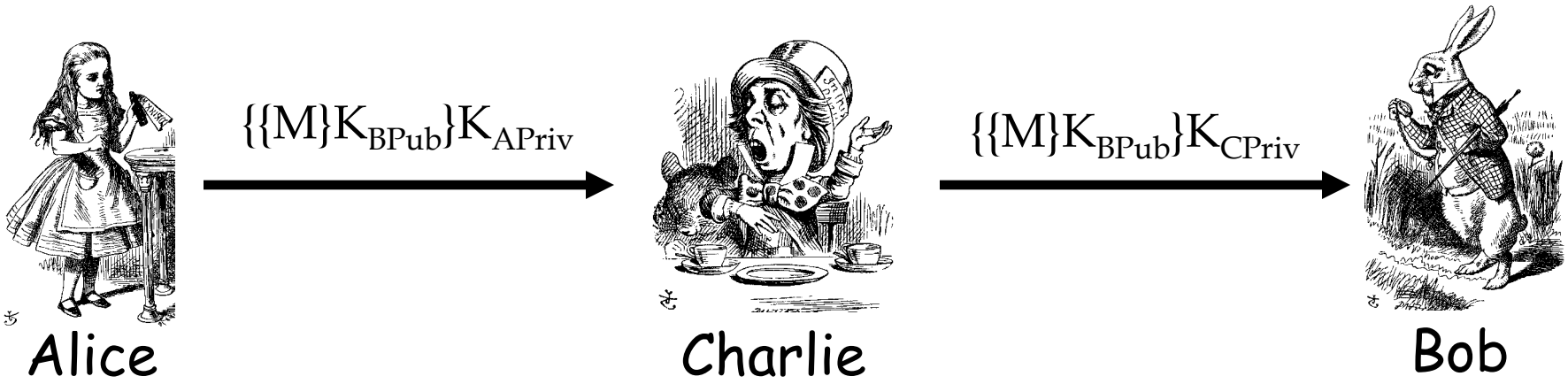
- M = "I love you"



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

Encrypt and Sign

- M = "My theory, which is mine…."



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

Public Key Infrastructure

Public Key Certificate

- Digital **certificate** contains name of user and user's public key (possibly other info too)
- It is ***signed*** by the issuer, a ***Certificate Authority*** (CA), such as VeriSign

  M = (Alice, Alice's public key), S = $\{M\}K_{CAPriv}$

  Alice's Certificate = (M, S)
- Signature on certificate is verified using CA's public key

  Must verify that M = $\{S\}K_{CAPub}$

Certificate Authority

- Certificate authority (CA) is a trusted 3rd party (TTP) — creates and signs certificates
- Verify signature to verify **integrity** & identity of **owner of corresponding private key**
  - Does **not** verify the identity of the **sender** of certificate — certificates are public!
- Big problem if CA makes a mistake
  - CA once issued Microsoft cert. to someone else
- A common format for certificates is X.509

PKI

- **Public Key Infrastructure (PKI)**: the stuff needed to securely use public key crypto
  - Key generation and management
  - Certificate authority (CA) or authorities
  - Certificate revocation lists (CRLs), etc.
- No general standard for PKI
- We mention 3 generic "trust models"
  - We only discuss the CA (or CAs)

PKI Trust Models (1)

- **Monopoly model**
  - One universally trusted organization is the CA for the known universe
  - Big problems if CA is ever compromised
  - Who will act as CA ???
    - System is useless if you don't trust the CA!

PKI Trust Models (2)

- **Oligarchy**
  - Multiple (as in, "a few") trusted CAs
  - This approach is used in browsers today
  - Browser may have 80 or more CA certificates, just to verify certificates!
  - User can decide which CA or CAs to trust

PKI Trust Models (3)

- **Anarchy model**
  - Everyone is a CA…
  - Users must decide who to trust
  - This approach used in PGP: "Web of trust"
- Why is it anarchy?
  - Suppose certificate is signed by Frank and you don't know Frank, but you do trust Bob and Bob says Alice is trustworthy and Alice vouches for Frank. Should you accept the certificate?
- **Many** other trust models/PKI issues

Confidentiality in the Real World

PKC Summary

- Two parties DO NOT need to trust each other.
- Two separate keys: a public and a private key.

- Keys (n users) = 2n

- **Services:** Data confidentiality, integrity, entity authentication, digital signatures, non repudiation, etc

- Many Ciphers Available
  - Examples: RSA, ElGamal, ECC Diffie Hellman, Digital Signature Standard. etc

- **Performance:** typically slower than Conventional Crypt

- **Key Management:** Great Flexibility

Symmetric Key vs Public Key

- Symmetric Key
  - **Speed**
  - No public key infrastructure (PKI) needed (but have to generate/distribute keys)
- Public Key
  - **Signatures** (non-repudiation)
  - No ***shared*** secret (but, do have to get private keys to the right user…)

Notation Reminder

- **Public key notation**
  - Sign M with Alice's **private key**

    $\{M\}K_{APriv}$
  - Encrypt M with Alice's **public key**

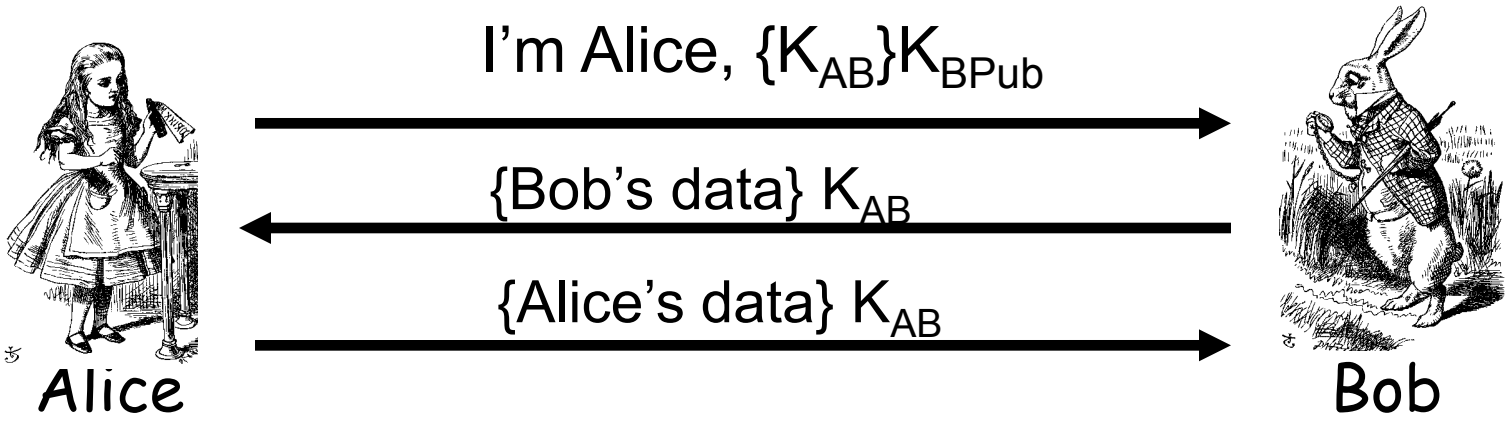
    $\{M\}K_{APub}$
- **Symmetric key notation**
  - Encrypt P with **symmetric key** K (shared by Alice and Bob)

    $C = E(P,K) = \{P\}K_{AB}$
  - Decrypt C with **symmetric key** K

    $P = D(C,K) = \{C\}K_{AB}$

Real World Confidentiality

- **Hybrid cryptosystem**
  - Public key crypto to establish a key
  - Symmetric key crypto to encrypt data…



- Can Bob be sure he's talking to Alice?

Next...Hash Functions++