Chapter 4: Public Key Cryptography

You should not live one way in private, another in public.
Publilius Syrus

Three may keep a secret, if two of them are dead.
Ben Franklin

Public Key Cryptography

- Two keys, one to encrypt, another to decrypt
  - Alice uses Bob's **public key** to encrypt
  - Only Bob's **private key** decrypts the message
- Based on "trap door, one way function"
  - "One way" means easy to compute in one direction, but hard to compute in other direction
  - Example: Given p and q, product N = pq easy to compute, but hard to find p and q from N
  - "Trap door" is used when creating key pairs

Public Key Cryptography

- Encryption
  - Suppose we ***encrypt*** M with Bob's public key
  - 's private key can ***decrypt*** C to recover M
- Digital Signature
  - ***signs*** by "encrypting" with his private key
  - Anyone can ***verify*** signature by "decrypting" with 's public key
  - But only Bob could have signed
  - Like a handwritten signature, but much better…

Knapsack

Knapsack Problem

- *Given a set of* n *weights* $W_0, W_1, \ldots, W_{n-1}$ *and a sum* S*, find* $a_i \in \{0,1\}$ *so that*

$$S = a_0W_0 + a_1W_1 + \ldots + a_{n-1}W_{n-1}$$

(technically, this is the subset sum problem)

- ***Example***
  - *Weights* (62,93,26,52,166,48,91,141)
  - *Problem: Find a subset that sums to* $S = 302$
  - *Answer:* $62 + 26 + 166 + 48 = 302$ (~10101100)
- *The (general) knapsack is NP-complete*

Knapsack Problem

- ***General knapsack*** *(GK) is hard to solve*
- *But* ***superincreasing knapsack*** *(SIK) is easy*
- *SIK each weight greater than the sum of all previous weights*
- ***Example***
  - *Weights* (2,3,7,14,30,57,120,251)
  - *Problem: Find subset that sums to* S = 186
  - *Work from largest to smallest weight*
  - *Answer:* 120 + 57 + 7 + 2 = 186 (10100110)

Knapsack Cryptosystem

1. Generate superincreasing knapsack (SIK)
2. Convert SIK to "general" knapsack (GK)
3. **Public Key:** GK
4. **Private Key:** SIK and conversion factor

- Goal…
  - Easy to encrypt with GK
  - With private key, easy to decrypt (solve SIK)
  - Without private key, Trudy has no choice but to try to solve GK

Example

- *Start with* (2,3,7,14,30,57,120,251) *as the SIK*
- *Choose* m = 41 *and* n = 491 *(*m, n *relatively prime,* n *exceeds sum of elements in SIK)*
- *Compute "general" knapsack*

  2 ▾ 41 mod 491 = 82
  3 ▾ 41 mod 491 = 123
  7 ▾ 41 mod 491 = 287
  14 ▾ 41 mod 491 = 83
  30 ▾ 41 mod 491 = 248
  57 ▾ 41 mod 491 = 373
  120 ▾ 41 mod 491 = 10
  251 ▾ 41 mod 491 = 471

- *"General" knapsack:* (82,123,287,83,248,373,10,471)

Knapsack Example

- ***Private key:*** (2,3,7,14,30,57,120,251)

  $m^{\blacksquare}$ mod n = $41^{\blacksquare}$ mod 491 = 12

  (m^{-1} mod n= s when m.s = 1 (mod n) or m.s = k.n+1)
- ***Public key:*** (82,123,287,83,248,373,10,471), n=491
- *Example: Encrypt* 10010110

  82 + 83 + 373 + 10 = 548
- *To decrypt, use private key…*
  - 548 · 12 = 193 mod 491
  - *Solve (easy) SIK with* S = 193

Knapsack Weakness

- **Trapdoor:** Convert SIK into "general" knapsack using modular arithmetic
- **One-way:** General knapsack easy to encrypt, hard to solve; SIK easy to solve
- This knapsack cryptosystem is **insecure**
  - Broken in 1983 with Apple II computer
  - The attack uses **lattice reduction**
- "General knapsack" is not general enough!
  - This special case of knapsack is easy to break

RSA

RSA

- *Invented by Clifford Cocks (GCHQ) and Rivest, Shamir, and Adleman (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*, we 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 $x^{\phi(n)} = 1 \bmod n$
- *Facts:*
  1) $ed = 1 \bmod (p - 1)(q - 1)$
  2) *By definition of "mod",* $ed = k(p - 1)(q - 1) + 1$
  3) $\phi(N) = (p - 1)(q - 1)$
- *Then* $ed - 1 = k(p - 1)(q - 1) = k\phi(N)$
- *So,* $C^d = M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\phi(N)}$
  $= M \cdot (M^{\phi(N)})^k \bmod N = M \cdot 1^k \bmod N =$ **M mod 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 mod 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*

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

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

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 ▼ 2, 5 = 2 ▼ 2 + 1, 10 = 2 ▼ 5, 20 = 2 ▼ 10
  - $5^1 = 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 = 25 \bmod 35$
  - $5^5 = (5^2)^2$ ▼ $5^1 = 25^2$ ▼ $5 = 3125 = 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 = 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 = 25 \bmod 35$
- *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
  - 0 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

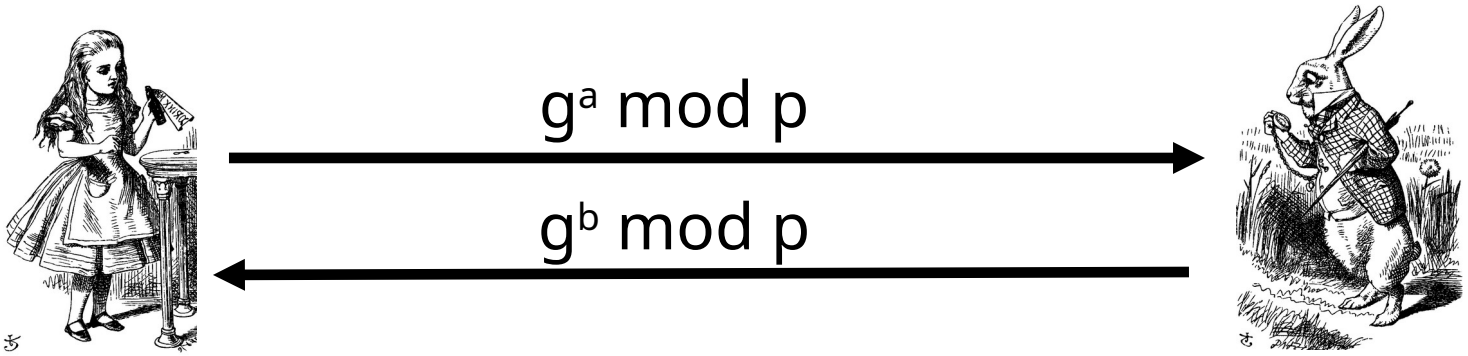
- *Invented by Williamson (GCHQ) and, independently, by D and H (Stanford)*
- *A "key exchange" algorithm*
  - *Used to establish a shared symmetric key*
  - ***Not** for encrypting or signing*
- *Based on **discrete log** problem*
  - ***Given:*** g, p, *and* $g^k \bmod p$
  - ***Find:*** *exponent*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

Diffie-Hellman

- Let p be prime, let g be a **generator**
  - For any x ∈ {1,2,...,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

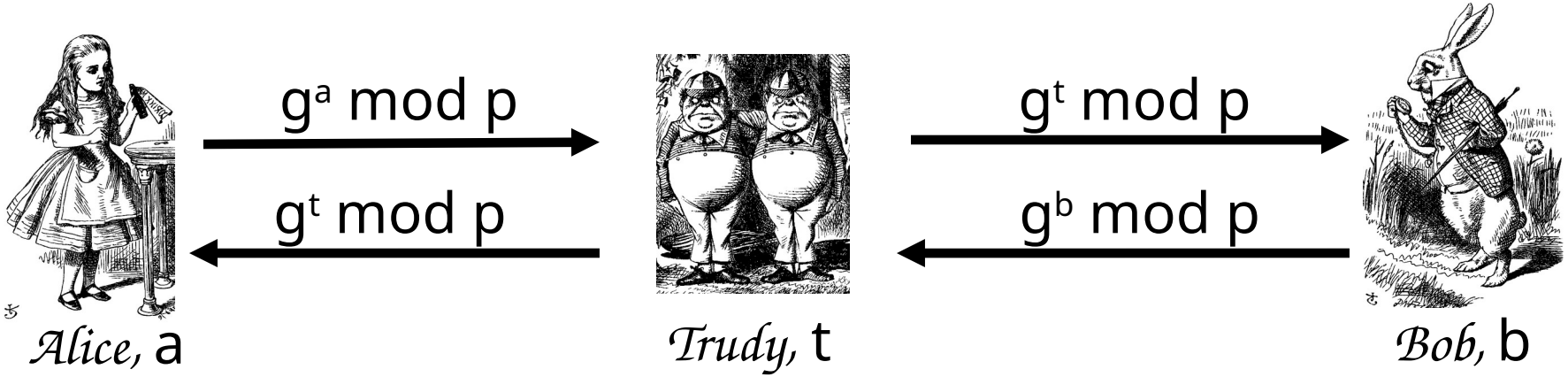
- *Alice computes* $(g^b)^a = g^{ba} = g^{ab} \bmod p$
- *Bob computes* $(g^a)^b = g^{ab} \bmod p$
- *They can use* $K = g^{ab} \bmod p$ *as symmetric key*

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…* $g^a g^b \bmod p = g^{a+b} \bmod p \quad g^{ab} \bmod p$
- *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* $g^{at} \bmod p$ *with Alice*
- *Trudy shares secret* $g^{bt} \bmod p$ *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, yet it is a big "win" overall

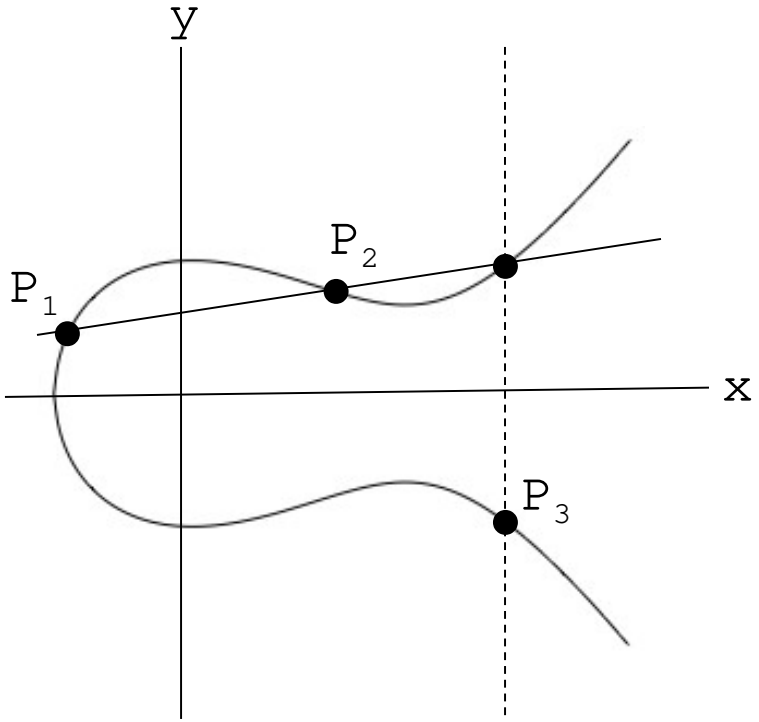
What is an Elliptic Curve?

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

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

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

Elliptic Curve Picture



- *Consider elliptic curve*

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

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

$$P_3 = P_1 + P_2$$

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,* $P_3 =$ 👍 *and*

  👍 $+ P = P$ *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* $(2,5)$ $\blacktriangle$ $b = 3$
- ***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:* $7(7,32) = (22,1)$

Larger ECC Example

- Example from Certicom ECCp-109
  - Challenge problem, solved in 2002
- Curve E: $y^2 = x^3 + ax + b \pmod p$
- Where

$$p = 564538252084441556247016902735257$$
$$a = 321094768129147601892514872825668$$
$$b = 430782315140218274262276694323197$$

- Now what?

ECC Example

- *The following point* P *is on the curve* E

(x,y) = (9733901098705906652315613390893 5, 14967037284616928576068237197889 8)

- *Let* k = 28118384031160194966820795453068 4
- *The* kP *is given by*

(x,y) = (44646769697405861057630861884284, 52296809889578588047540374779097)

- *And this point is also on the curve* E

Really Big Numbers!

- Numbers are big, but not big enough
  - ECCp-109 bit (32 digit) solved in 2002
- Today, ECC DH needs bigger numbers
- But RSA needs way bigger numbers
  - Minimum RSA modulus today is 1024 bits
  - That is, more than 300 decimal digits
  - That's about 10x the size in ECC example
  - And 2048 bit RSA modulus is common…

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:***
  $[M]_{Alice}$
- *Encrypt* message M *with Alice's* ***public key:***
  $\{M\}_{Alice}$
- *Then*

  $\{[M]_{Alice}\}_{Alice} = M$

  $[\{M\}_{Alice}]_{Alice} = M$

Sign and Encrypt

vs

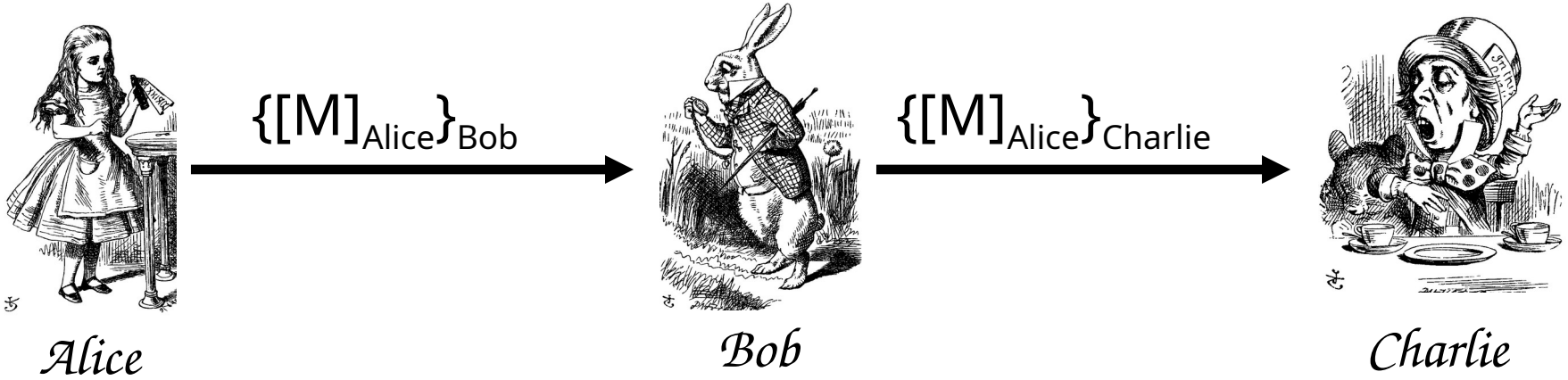
Encrypt and Sign

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:** $\{[M]_{Alice}\}_{Bob}$
  - **Encrypt and sign:** $[\{M\}_{Bob}]_{Alice}$
- Can the order possibly matter?

Sign and Encrypt

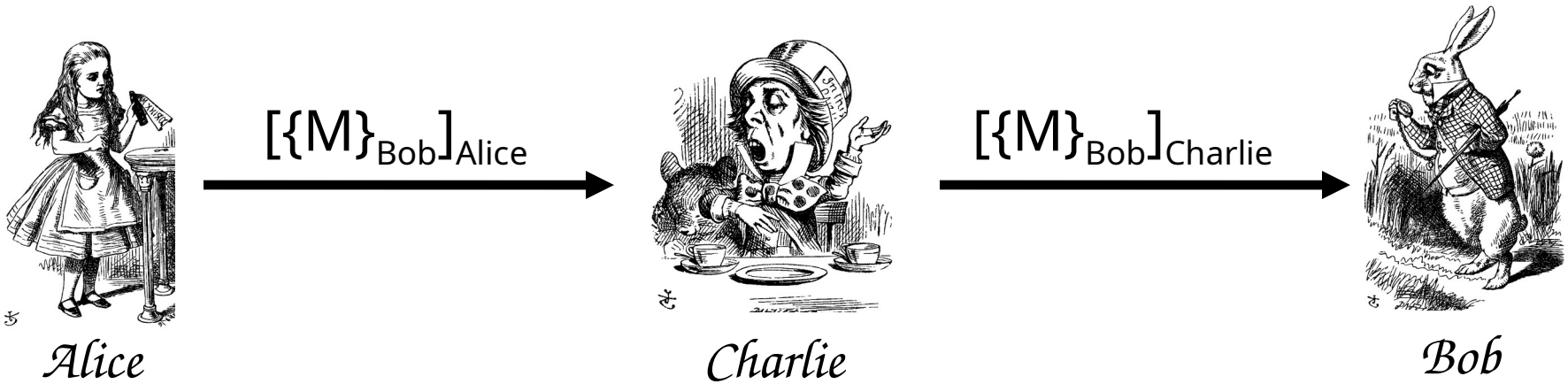
- M = *"I love you"*



- ***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

$$M = (\text{Alice, Alice's public key}),\ S = [M]_{CA}$$

Alice's Certificate = (M, S)

- Signature on certificate is verified using CA's public key

Must verify that $M = \{S\}_{CA}$

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

- 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

- 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

- 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

Symmetric Key vs Public Key

- Symmetric key + 's
  - **Speed**
  - No public key infrastructure (PKI) needed (but have to generate/distribute keys)
- Public Key + 's
  - **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***

    $[M]_{Alice}$
  - *Encrypt* M *with Alice's* ***public key***

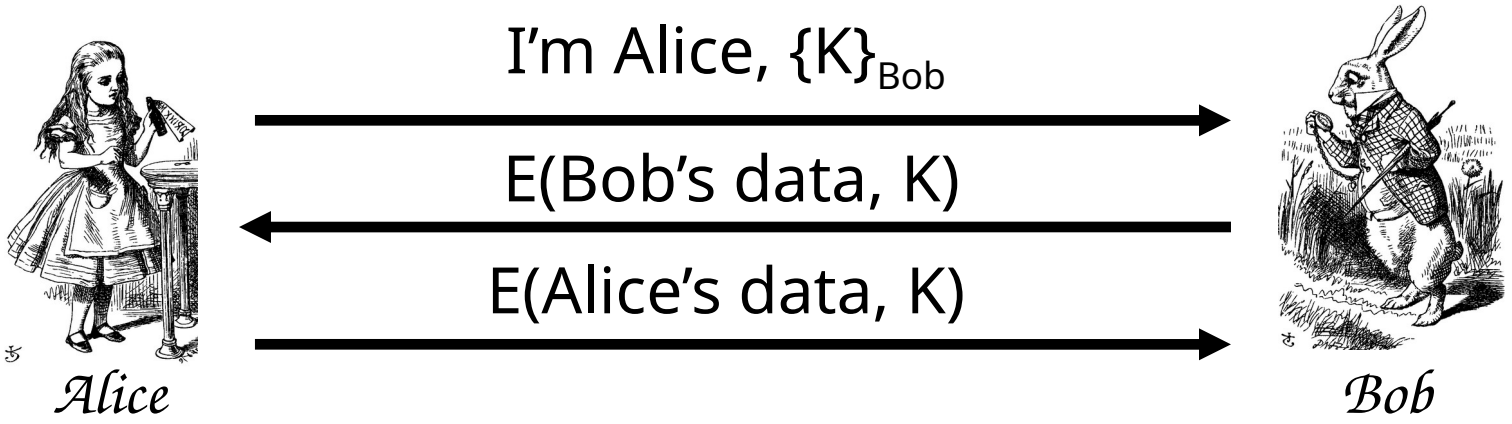
    $\{M\}_{Alice}$
- *Symmetric key notation*
  - *Encrypt* P *with* ***symmetric key*** K

    C = E(P,K)
  - *Decrypt* C *with* ***symmetric key*** K

    P = D(C,K)

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?