

Modular Arithmetic

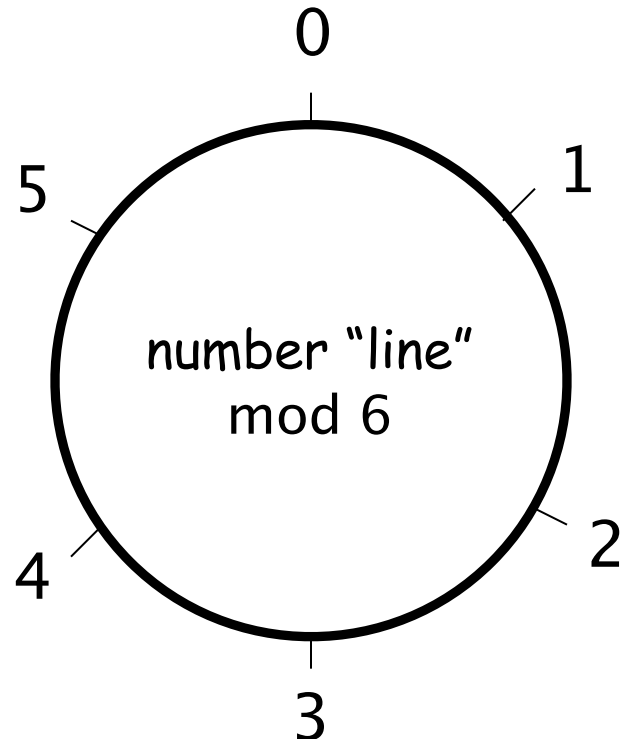
Clock Arithmetic

□ For integers x and n , " $x \bmod n$ " is the remainder when we compute $x \div n$

- We can also say " x modulo n "

□ Examples

- $33 \bmod 6 = 3$
- $33 \bmod 5 = 3$
- $7 \bmod 6 = 1$
- $51 \bmod 17 = 0$
- $17 \bmod 6 = 5$



Modular Addition

□ Notation and fun facts

- $7 \bmod 6 = 1$
- $7 = 13 = 1 \bmod 6$
- $((a \bmod n) + (b \bmod n)) \bmod n = (a + b) \bmod n$
- $((a \bmod n)(b \bmod n)) \bmod n = ab \bmod n$

□ Addition Examples

- $3 + 5 = 2 \bmod 6$
- $2 + 4 = 0 \bmod 6$
- $3 + 3 = 0 \bmod 6$
- $(7 + 12) \bmod 6 = 19 \bmod 6 = 1 \bmod 6$
- $(7 + 12) \bmod 6 = (1 + 0) \bmod 6 = 1 \bmod 6$

Modular Multiplication

□ Multiplication Examples

- $3 \cdot 4 = 0 \bmod 6$
- $2 \cdot 4 = 2 \bmod 6$
- $5 \cdot 5 = 1 \bmod 6$
- $(7 \cdot 4) \bmod 6 = 28 \bmod 6 = 4 \bmod 6$
- $(7 \cdot 4) \bmod 6 = (1 \cdot 4) \bmod 6 = 4 \bmod 6$

Modular Inverses

- Additive inverse of $x \bmod n$, denoted $-x \bmod n$, is the number that must be added to x to get $0 \bmod n$
 - $-2 \bmod 6 = 4$, since $2 + 4 = 0 \bmod 6$
- Multiplicative inverse of $x \bmod n$, denoted $x^{-1} \bmod n$, is the number that must be multiplied by x to get $1 \bmod n$
 - $3^{-1} \bmod 7 = 5$, since $3 \cdot 5 = 1 \bmod 7$

"Trivial" Arithmetic

Additive inverse of x :

$$x + (?) = 0$$

$$x + (-x) = 0$$

Multiplicative inverse of x :

$$x * (?) = 1$$

$$x * (1/x) = 1$$

Modular Arithmetic

Additive inverse of x is:

$$x + (?) \bmod n = 0$$

Multiplicative inverse of x :

$$x * (?) \bmod n = 1$$

Modular Arithmetic Quiz

- ❑ Q: What is $-3 \bmod 6$?
- ❑ A: 3
- ❑ Q: What is $-1 \bmod 6$?
- ❑ A: 5
- ❑ Q: What is $5^{-1} \bmod 6$?
- ❑ A: 5
- ❑ Q: What is $2^{-1} \bmod 6$?
- ❑ A: No number works!
- ❑ Multiplicative inverse might not exist

Relative Primality

- ❑ x and y are **relatively prime** if they have no common factor other than 1
- ❑ $x^{-1} \bmod y$ exists only when x and y are relatively prime
- ❑ If it exists, $x^{-1} \bmod y$ is easy to compute using Euclidean Algorithm
 - We won't do the computation here
 - But, an efficient algorithm exists

Totient Function

- $\varphi(n)$ is “the number of numbers less than n that are relatively prime to n ”
 - Here, “numbers” are positive integers
- Examples
 - $\varphi(4) = 2$ since 4 is relatively prime to 3 and 1
 - $\varphi(5) = 4$ since 5 is relatively prime to 1,2,3,4
 - $\varphi(12) = 4$
 - $\varphi(p) = p-1$ if p is prime
 - $\varphi(pq) = (p-1)(q-1)$ if p and q prime

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
- Bob's private key can **decrypt** C to recover M

❑ Digital Signature

- Bob **signs** by "encrypting" with his private key
- Anyone can **verify** signature by "decrypting" with Bob's public key
- But only Bob could have signed
- Like a handwritten signature, but much better...

Public Key Cryptography

- ❑ 2 keys mathematically related
 - Cracking with the private key might be easy
 - Cracking with public key must be very tough!
- ❑ If I know only the public key... I should not be able to crack a cipher
 - At least, not in little time!

Knapsack



Knapsack Problem

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

$$S = a_0 W_0 + a_1 W_1 + \dots + a_{n-1} W_{n-1}$$

(technically, this is the subset sum problem)

- **Example**

- Weights (62, 93, 26, 52, 166, 48, 91, 141)
- Problem: Find a subset that sums to $S = 302$

$$S = (1*62) + (0*93) + (1*26) + (0*52) + (1*166) + (1*48) + (0*91) + (0*141)$$

$$S = 62 + 26 + 166 + 48 = 302$$

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: $120 + 57 + 7 + 2 = 186$

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
 - $2 \cdot 41 \bmod 491 = 82$
 - $3 \cdot 41 \bmod 491 = 123$
 - $7 \cdot 41 \bmod 491 = 287$
 - $14 \cdot 41 \bmod 491 = 83$
 - $30 \cdot 41 \bmod 491 = 248$
 - $57 \cdot 41 \bmod 491 = 373$
 - $120 \cdot 41 \bmod 491 = 10$
 - $251 \cdot 41 \bmod 491 = 471$
- ❑ “General” knapsack: (82,123,287,83,248,373,10,471)


Knapsack Example

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

$$m^{-1} \bmod n = 41^{-1} \bmod 491 = 12$$

□ **Public key:** (82,123,287,83,248,373,10,471), $n=491$

□ **Example: Encrypt** 10010110

82,123,287,83,248,373,10,471  mult
1, 0, 0, 1, 0, 1, 1, 0

$$82 + 83 + 373 + 10 = \mathbf{548}$$

Knapsack Example

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

$$m^{-1} \bmod n = 41^{-1} \bmod 491 = 12$$

❑ **Public key:** (82,123,287,83,248,373,10,471), $n=491$

❑ To decrypt, use private key...

- $548 \cdot 12 = 193 \bmod 491$

- Solve (easy) SIK with $S = 193$

2,3,7,14,30,57,120,251

- Obtain plaintext 10010110



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 $ed = 1 \pmod{(p-1)(q-1)}$
- ❑ 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, $M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{k\varphi(N)} =$
 $M \cdot (M^{\varphi(N)})^k \bmod N = M \cdot 1^k \bmod N = M \bmod N$

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 $ed = 1 \pmod{20}$
 - 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
$$\begin{aligned} M &= C^d \bmod N = 17^7 = 410,338,673 \\ &= 12,434,505 * 33 + 8 = 8 \bmod 33 \end{aligned}$$

More Efficient RSA (1)

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

$e=3$ might be a problem

- ❑ 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
 - o "better" than $e = 3$

Diffie-Hellman

Diffie-Hellman Key Exchange

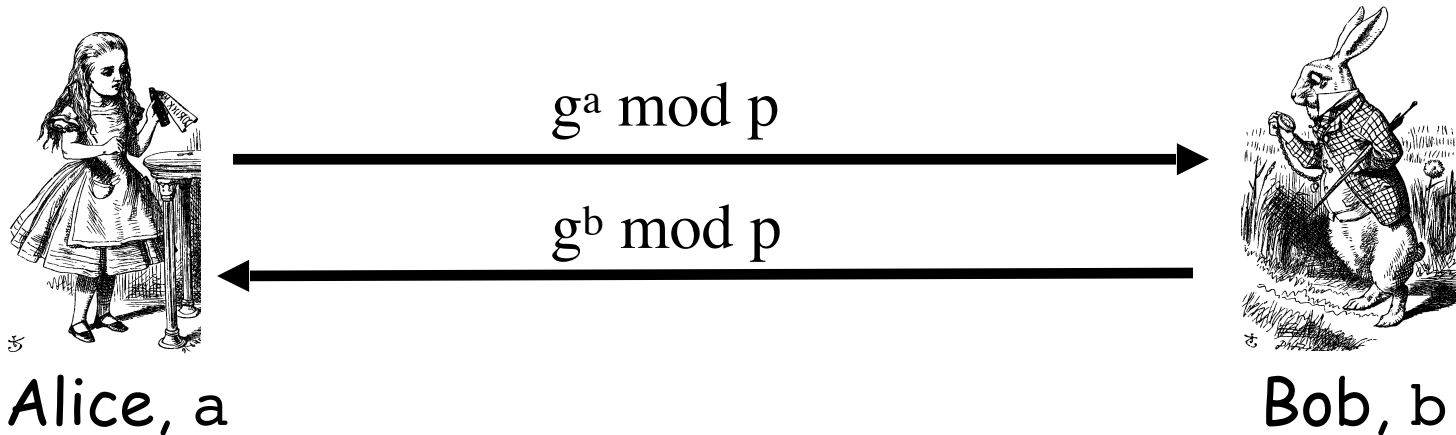
- ❑ Invented by Williamson (GCHQ) and, independently, by D and H (Stanford)
- ❑ A “key exchange” algorithm
 - Used to establish a shared symmetric key
 - Not for encrypting or signing
- ❑ Based on **discrete log** problem
 - **Given:** g , p , and $g^k \bmod p$
 - **Find:** exponent k

Diffie-Hellman

- ❑ Let p be prime, let g be a **generator**
 - For any $x \in \{1, 2, \dots, p-1\}$ 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 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

If we have these selected values:

- $g = 2$ (public)
- $p = 3$ (public)
- $a = 2$ (private, only Alice knows it)
- $b = 3$ (private, only Bob knows it)
- Alice's computed value: $g^a \bmod p = 2^2 \bmod 3 = 1$
- Bob's computed value: $g^b \bmod p = 2^3 \bmod 3 = 2$
- Shared key:
 $g^{ab} \bmod p = 2^{2*3} \bmod 3 =$
 $= 2^6 \bmod 3 = 64 \bmod 3 = 1$

Now, let's look from the perspective of the two parties.

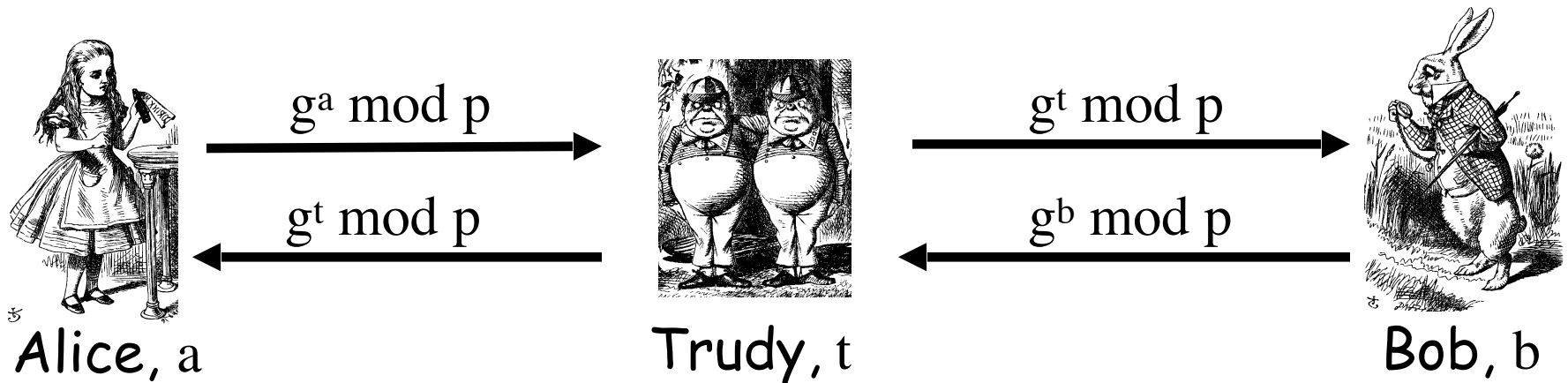
1. Alice sends $g^a \bmod p = 1$ to Bob
2. Bob reads 1 and computes the shared key, that is:
 - o $\left[(g^a) \bmod p \right]^b = 1^3 \bmod 3 = 1 \bmod 3 = 1$
3. Bob sends $g^b \bmod p = 2$ to Alice
4. Alice reads 2 and computes the shared key, that is:
 - o $\left[(g^b) \bmod p \right]^a = 2^2 \bmod 3 = 4 \bmod 3 = 1$
5. They both compute the same key and they can start to use it to encrypt their following messages.

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 $g^{at} \bmod p$ 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

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**: $[M]_{\text{Alice}}$

□ **Encrypt** message M with Alice's **public key**: $\{M\}_{\text{Alice}}$

□ **Then**

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

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

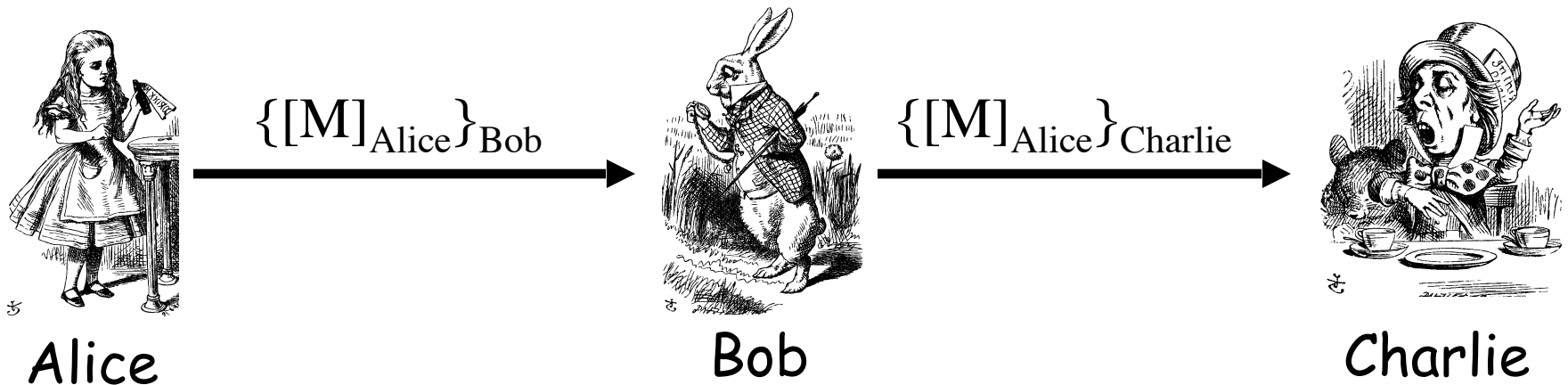
Sign and Encrypt vs Encrypt and Sign

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

Sign and Encrypt

□ $M = \text{"I love you"}$

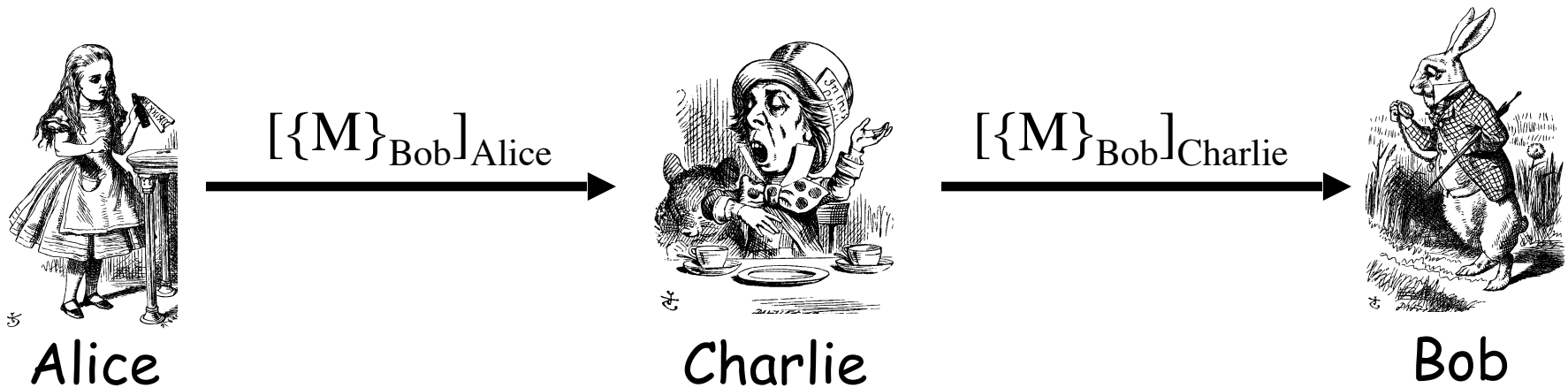


□ **Q:** What's the problem?

□ **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

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

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**

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

- Encrypt M with Alice's **public key**

$$\{M\}_{\text{Alice}}$$

□ Symmetric key notation

- Encrypt P with **symmetric key** K

$$C = E(P, K)$$

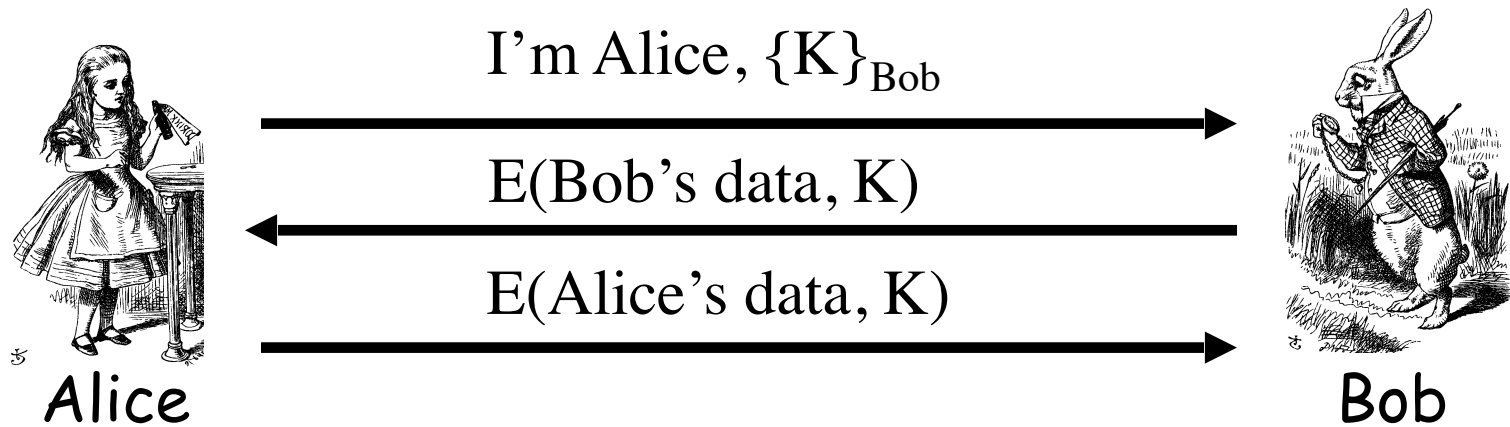
- Decrypt C with **symmetric key** K

$$P = D(C, K)$$

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?