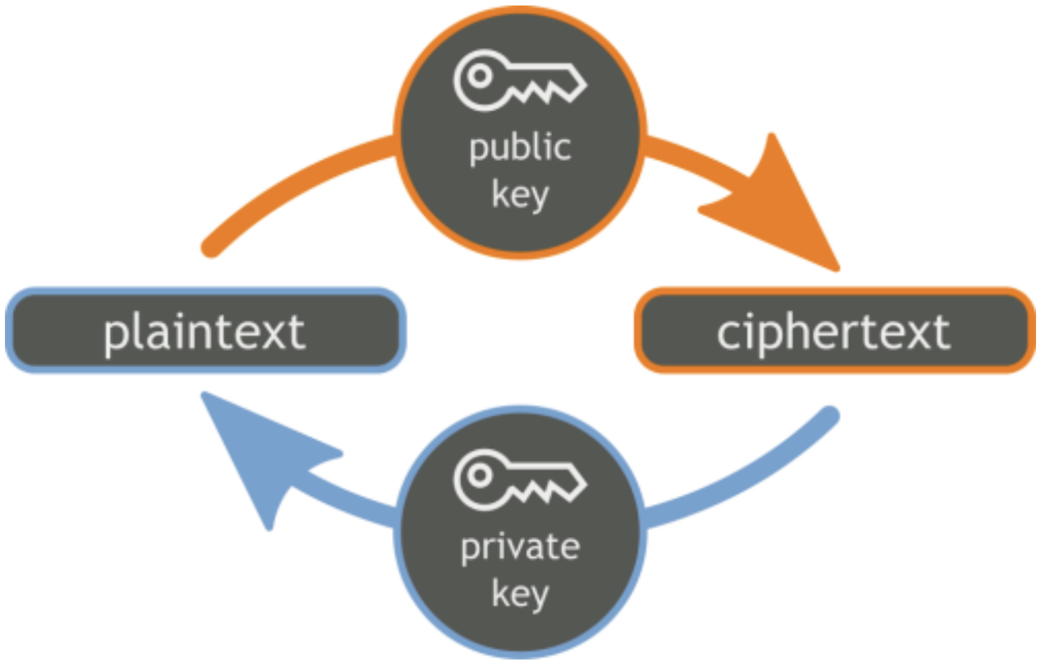
Public Key Crypto



CS 458: Information Security

Kevin Jin

Reading Material

- Text Chapters 2.3-4 and 21.3-4
- Handbook of Applied Cryptography, Chapter 8
  - http://www.cacr.math.uwaterloo.ca/hac/

Some materials borrowed from Mark Stamp at San Jose State University

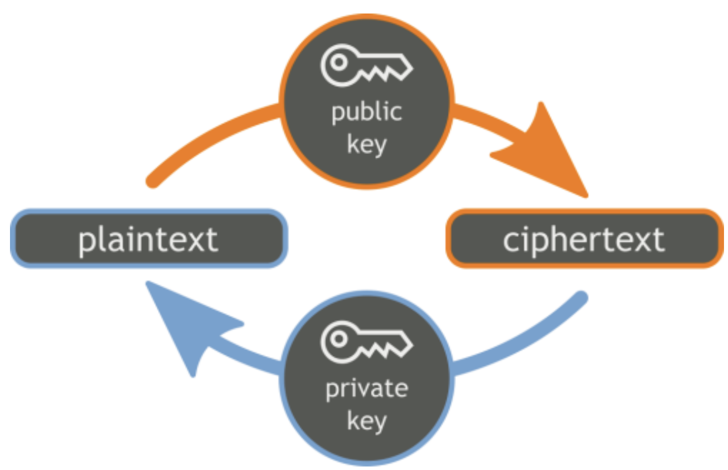
Symmetric Key's Problems

- Every pair of people must share a secret key
  - E.g., Alice, Bob, Carol, David

  K_{AB}, K_{AC}, K_{AD},K_{BC}, K_{BD}, K_{CD}
- How do you keep track of them all?
  - $O(N^2)$ keys for N people
- How do you exchange them?
  - Must use a secure, out-of-band channel

Public Key Cryptography

- Cryptographers to the rescue!
- Two keys:
  - Private key known only to owner
  - Public key available to anyone
  - One key pair per person
    - O(N) keys

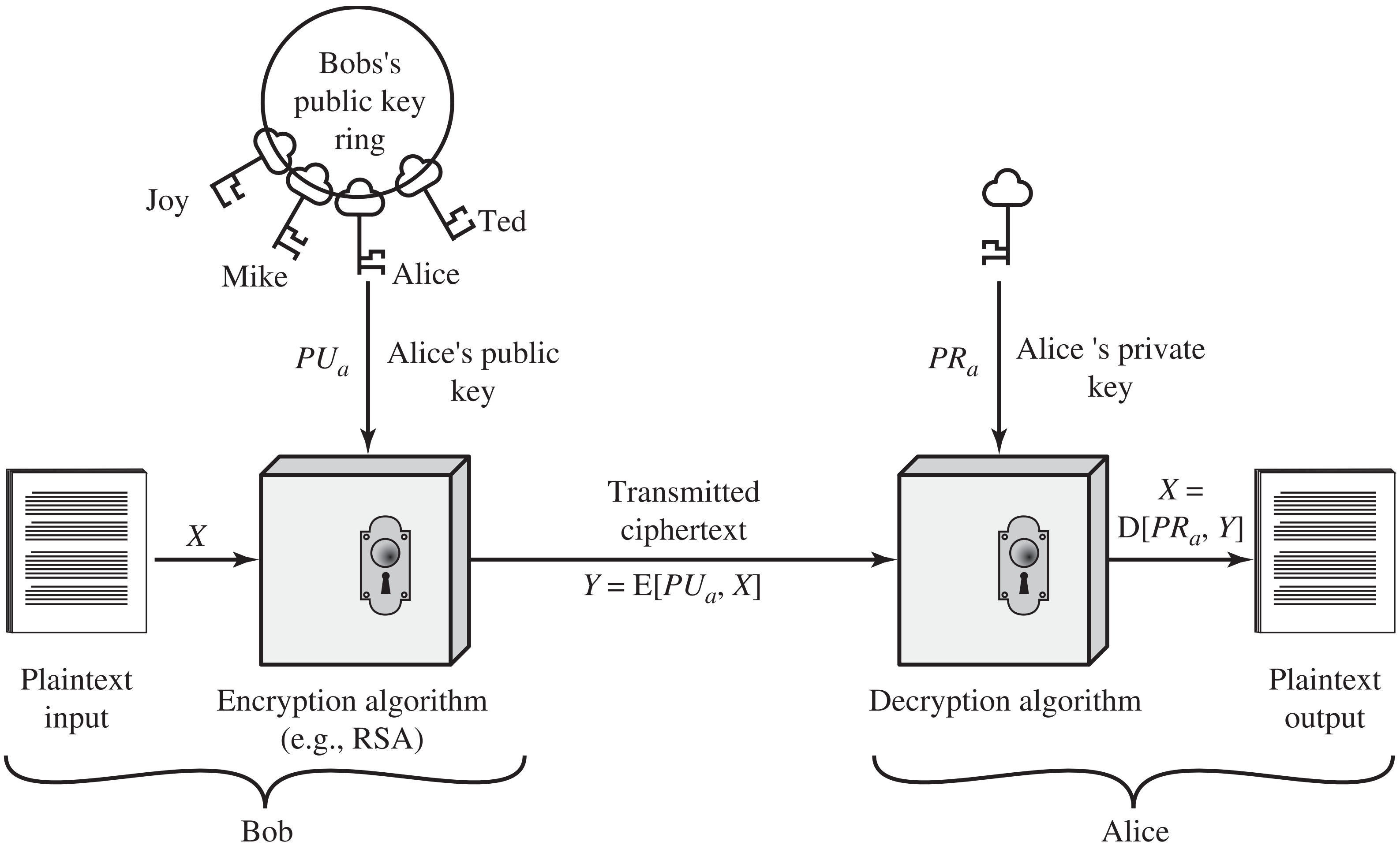


Public Key Cryptography

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 given N, it is hard to find p and q
- "Trap door" used to create key pairs

Public Key Cryptography



What is the Usage of Public Key Cryptography?

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

Public-Key Cryptography

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone
- Idea
  - *Confidentiality*: encipher using public key, decipher using private key
  - *Integrity/authentication*: encipher using private key, decipher using public one
  - *Symmetric Key distribution*

General Facts about Public Key Systems

- Public Key Systems are **much slower** than Symmetric Key Systems
  - RSA 100 to 1000 times slower than DES. 10,000 times slower than AES?
  - Generally used in conjunction with a symmetric system for bulk encryption
- Public Key Systems are based on “hard” problems
  - Factoring large composites of primes, discrete logarithms, elliptic curves

Major Public Key Algorithms

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of secret keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Only a handful of public key systems perform both encryption and signatures

Diffie-Hellman

Merkle, Hellman, Diffie

Diffie-Hellman

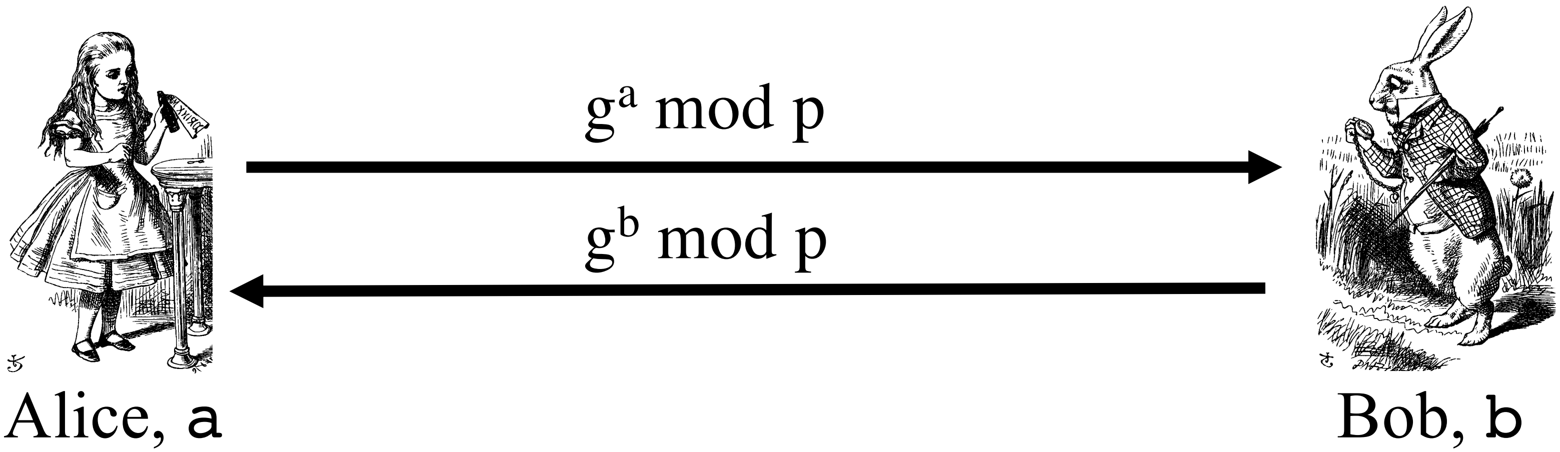
- The first public key cryptosystem proposed
  - Invented by Williamson (GCHQ) and, independently, by D and H (Stanford)
- Usually used for exchanging keys securely
- Compute a common, shared key
  - Called a *symmetric key exchange protocol*
- Based on discrete logarithm problem
  - Given integers n and g and prime number p, compute k such that $n = g^k \bmod p$
  - Solutions known for small p
  - Solutions computationally infeasible as p grows large

Diffie-Hellman

- Let p be prime, let g be a **generator**
  - For any $n \in \{1, 2, \ldots, p\text{-}1\}$ there is k s.t. $n = g^k \bmod p$
- 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



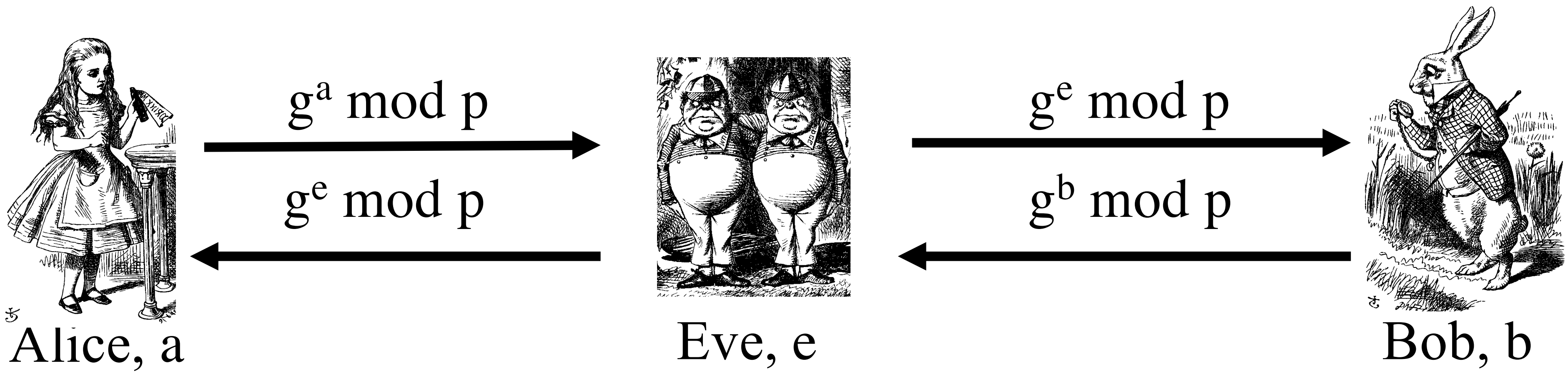
- Alice computes $g^{ab} \bmod p = (g^b)^a \bmod p$
  $= (g^b \bmod p)^a \bmod p$
- Bob computes $g^{ab} \bmod p = (g^a)^b \bmod p$
  $= (g^a \bmod p)^b \bmod p$
- Use $K = g^{ab} \bmod p$ as symmetric key

Diffie-Hellman

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

Diffie-Hellman

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



- Eve shares secret g^{ae} mod p with Alice
- Eve shares secret g^{be} mod p with Bob
- Alice and Bob don’t know Eve exists!

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
- In any case, you MUST be aware of MiM attack on Diffie-Hellman

Real public DH values

- For IPSec and SSL, there are a small set of g's and p's published that all standard implementations support.
  - Group 1 and 2
    - http://tools.ietf.org/html/rfc2409
  - Group 5 and newer proposed values
    - http://tools.ietf.org/html/draft-ietf-ipsec-ike-modp-groups-00

RSA

RSA

- By Clifford Cocks (GCHQ), independently, **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 $ed = 1 \pmod{(p-1)(q-1)}$
- **Public key** is (N, e)
- **Private key** is d

Modulo Operations

- The RSA algorithm is based on modulo operations
- `a mod n` is the remainder after division of a by the modulus n
- Second number is called modulus
- For example, (10 mod 3) equals to 1 and (15 mod 5) equals to 0
- Modulo operations are distributive:

$$(a+b) \mod n = [(a \mod n) + (b \mod n)] \mod n$$

$$a * b \mod n = [(a \mod n) * (b \mod n)] \mod n$$

$$a^x \mod n = (a \mod n)^x \mod n$$

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 Eve can factor $N = pq$, she can use e to easily find d, since $ed = 1 \pmod{(p-1)(q-1)}$
- **Factoring the modulus breaks RSA**
  - Is factoring the only way to break RSA?

Simple RSA Example

- Example of 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 $M = 8$
- Ciphertext C is computed as

$$C = M^e \bmod N = 8^3 \bmod 33 = 512 \bmod 33 = 17$$

- Decrypt C to recover the message M by

$$M = C^d \bmod N = 17^7 \bmod 33 = 410{,}338{,}673 \bmod 33 = 8$$

$$(410{,}338{,}673 = 12{,}434{,}505 * 33 + 8)$$

More Efficient RSA

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

Does RSA Really Work?

- Given $C = M^e \bmod N$
  we must show $M = C^d \bmod N = M^{ed} \bmod N$
- We'll use **Euler's Theorem:**
  If x is relatively prime to N, then $x^{\varphi(N)} = 1 \pmod N$

  $\varphi(N)$ is the number of integers in $\{1, 2, ..., N\}$ relatively prime to N.
- Facts:
  1) $ed = 1 \pmod{(p-1)(q-1)}$
  2) By definition of "mod", $ed = k(p-1)(q-1) + 1$
  3) $\varphi(N) = (p-1)(q-1)$
- Then $ed - 1 = k(p-1)(q-1) = k\,\varphi(N)$
- Finally, $\mathbf{M^{ed}} = M^{(ed-1)+1} = M^*M^{ed-1} = M^*M^{k\,\varphi(N)}$
  $= M^*(M^{\varphi(N)})^k \bmod N = M^*1^k \bmod N = \mathbf{M \bmod N}$

Where is the security?

- What problem must you solve to discover d?

Difficulty of Factorization

- Thought to be in NP, but not proven to be
- Decades of experience suggests difficulty
- Algorithm on quantum computer developed that has polynomial complexity

Best Known factorization algorithm (General number field sieve) has complexity

$$O(e^{(\frac{64}{9})b^{(1/3)}(\log b)^{(2/3)}})$$

where b is the number of bits in the number being factored.

236 digit number factored in 2 years, using distributed computing

Example: Confidentiality

- Take $p = 7$, $q = 11$, so $n = 77$ and $\Phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $07^{17} \bmod 77 = 28$
  - $04^{17} \bmod 77 = 16$
  - $11^{17} \bmod 77 = 44$
  - $11^{17} \bmod 77 = 44$
  - $14^{17} \bmod 77 = 42$
- Bob sends 28 16 44 44 42

Example

- Alice receives 28 16 44 44 42
- Alice uses private key, $d = 53$, to decrypt message; n = 77
  - $28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - $44^{53} \bmod 77 = 11$
  - $42^{53} \bmod 77 = 14$
- Alice translates message to letters to read HELLO
  - No one else could read it, as only Alice knows her private key and that is needed for decryption

Example: Data Integrity/Authentication

- Take $p = 7$, $q = 11$, so $n = 77$ and $\Phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49

Example

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:
  - $35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$
- Bob translates message to letters to read HELLO
  - Alice sent it as only she knows her private key, so no one else could have enciphered it
  - If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

Example: Both

- Alice wants to send Bob message HELLO both enciphered and authenticated (integrity-checked)
  - Alice's keys: public (17, 77); private: 53
  - Bob's keys: public: (37, 77); private: 13
- Alice enciphers HELLO (07 04 11 11 14):
  - $(07^{53} \bmod 77)^{37} \bmod 77 = 07$
  - $(04^{53} \bmod 77)^{37} \bmod 77 = 37$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(14^{53} \bmod 77)^{37} \bmod 77 = 14$
- Alice sends 07 37 44 44 14

Warnings

- Encipher message in blocks **considerably larger** than the examples here
  - If 1 character per block, RSA can be broken using statistical attacks
    (just like classical cryptosystems)
  - Attacker cannot alter letters, but can rearrange them and alter message meaning
    - Example: reverse enciphered message of text ON to get NO

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!** Since Bob also knows the symmetric key, he could have forged message
- **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!** Only someone with Alice's private key could have signed the order
- This assumes Alice's private key is not stolen (revocation problem)

Sign and Encrypt
vs
Encrypt and Sign

Public Key Notation

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

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

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

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"

Alice

$\{[M]_{Alice}\}_{Bob}$ →

Bob

$\{[M]_{Alice}\}_{Charlie}$ →

Charlie

- **Q:** What's the problem?
- **A:** Charlie misunderstand crypto!

Encrypt and Sign

- M = "My theory, which is mine…."

Alice

$[\{M\}_{Bob}]_{Alice}$ →

Charlie

$[\{M\}_{Bob}]_{Charlie}$ →

Bob

- **Note** that Charlie cannot decrypt M
- **Q:** What is the problem?
- **A:** Bob misunderstand crypto!

  No problem: public key is public

Key Points

- Public Key systems enable multiple operations
  - Confidentiality (key encryption)
  - Integrity and nonrepudiation
  - Symmetric key exchange
- Slower than symmetric crypto, but still practical
  - Especially in hybrid modes