

# Cryptography CS 555

## Lecture 12



Department of Computer Sciences  
Purdue University

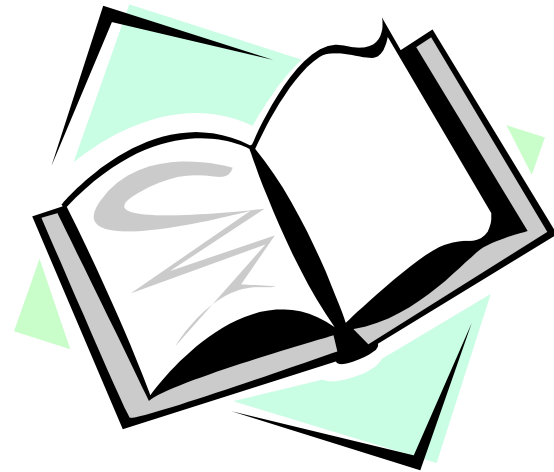
# Lecture Outline

- Public cryptography
- RSA
- Factorization



# Recommended Reading

- From Stinson, Chapter 5



# Project Proposal

- Due next Tuesday in class or by email (crisn) if you can not come to the class (about 2 pages)
- Exceptional case: you can also submit your Project proposal the Tuesday after the midterm if you are still looking for a project
- An updated list of projects send to the list last night.
- Outline of the proposal (suggestions)
  - Problem definition (what do you want to solve)
  - Plan of action (how do you approach the problem)
  - Evaluation methodology (how will you decide if you succeeded or not)

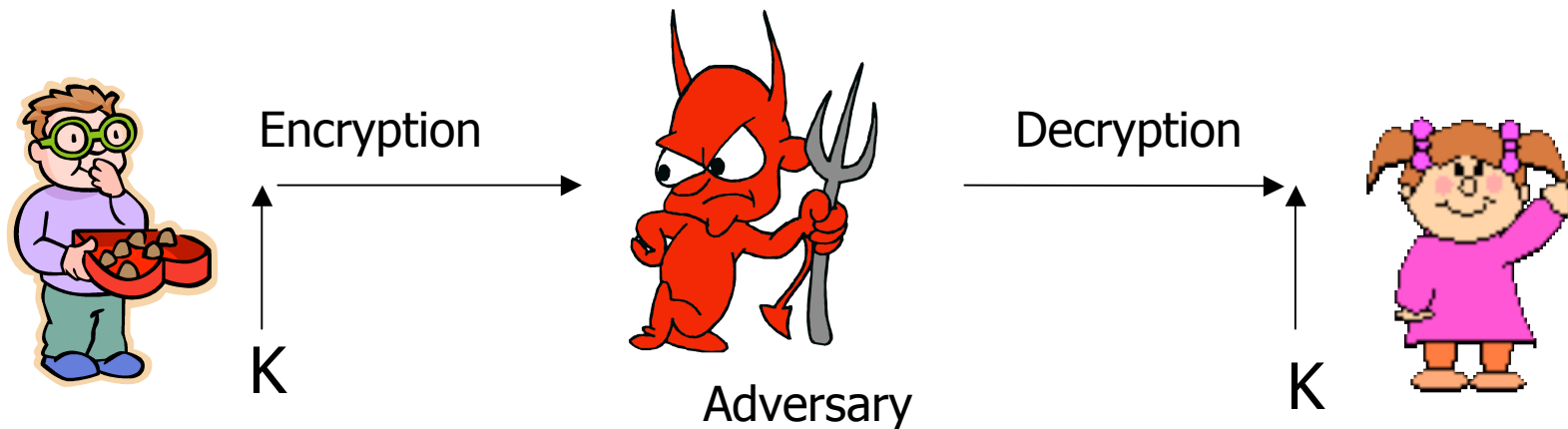
# Midterm

- Material includes today's lecture
- There will be 5 problems similar with the ones in homework
- Closed books, closed notes
- No number theory problems
- Some of the questions might ask you to make connections
- You don't need to remember schemes (DES, HASH)
- There will be 5 problems, 3 similar with the ones in homework and 2 that will test your understanding on the material.

# Midterm

- Difference between perfect secrecy (Shannon's definition) and how ciphers and hash are evaluated with respect to security
- The general mechanisms of iterative hash functions
- Difference between stream ciphers and block ciphers
- Current considered "secure" sizes for the protocols discussed so far
- DES properties

# Secure Communication



Symmetric encryption requires the parties to share a common secret key. **How to get the key in the first place?**

# Public Key Cryptography

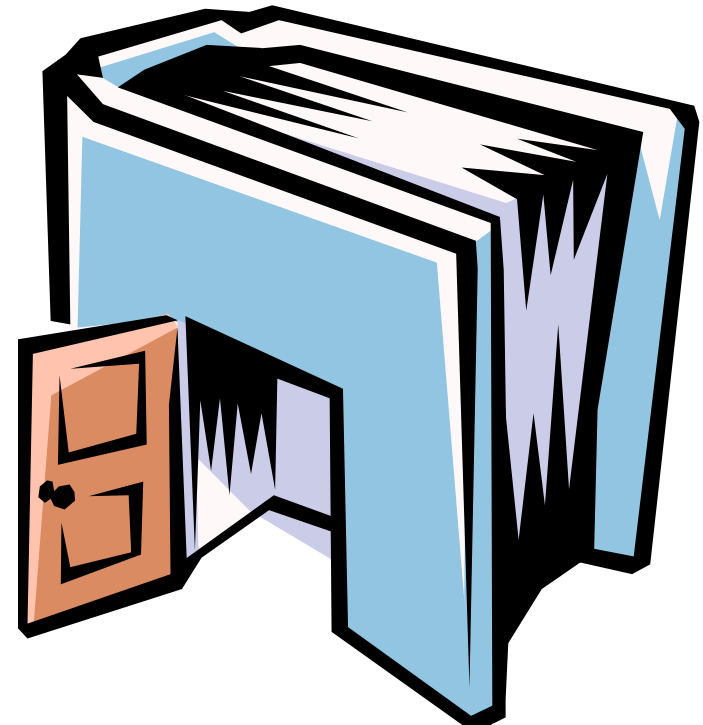
- Each party has a PAIR (P, S) of keys: P is the **public** key and S is the **secret** key.
- Knowing the public-key and the cipher, it is still computationally infeasible to compute the private key (an NP-class problem).
- The public-key may be distributed to anyone wishing to communicate securely with its owner
- $\text{Dec}_S(\text{Enc}_P) = M$



# Trapdoor Functions

## Definition:

A function  $f: \{0,1\}^* \rightarrow \{0,1\}^*$  is a trapdoor function iff  $F(x)$  is a one-way function such that given some extra information it becomes feasible to find for any  $y$  in  $\text{Img}(f)$  and  $x$  in  $X$ , s.t.  $y = f(x)$



Public key cryptography  
relies on trapdoor functions

# Trapdoor Functions

Factoring integers:  $f(p,q) = n = pq$

- Easy:  $pq$
- Hard: factoring  $pq$  into  $p$  and  $q$

Root-extraction:  $f(p,q,e,y) = y^e \bmod pq$

- Easy:  $y^e \bmod pq$
- Hard: given  $pq$ ,  $e$ , and  $y^e \bmod pq$ , compute a  $y'$  such that  $y'^e = y^e \bmod pq$

Discrete log problem:  $f(g,p,x) = g^x \bmod p$

- Easy:  $g^x \bmod p$
- Hard: determine  $x$  from  $p$ ,  $g$ , and  $g^x \bmod p$

# RSA Algorithm

- Invented in **1978** by Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman
- Security relies on the difficulty of factoring large composite numbers
- Published as R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978

# RSA Description

## Key generation:

Select 2 large prime numbers of about the same size,  $p$  and  $q$

Compute  $n = pq$ , and  $\phi = (q-1)(p-1)$

Select a random integer  $e$ ,  $1 < e < \phi(n)$ , s.t.  
 $\gcd(e, \phi(n)) = 1$

Compute  $d$ ,  $1 < d < \phi$  s.t.  $ed \equiv 1 \pmod{\phi(n)}$

**Public key:**  $(e, n)$

**Secret key:**  $d$

Note:  $\phi(n)$  is Euler's Totient function

How do you know that exists  $ed$  such that  $ed \equiv 1 \pmod{\phi(n)}$ ?

# RSA Description (cont.)

## Encryption

For the message  $M$ ,  $0 < M < n$   
use public key  $(e, n)$   
compute  $C = M^e \bmod n$

## Decryption

For a message  $C$   
use private key  $(d)$   
Compute  $C^d \bmod n = (M^e \bmod n)^d \bmod n = M$

# RSA Example

- $p = 11, q = 7, n = 77, \phi(n) = 60$
- $d = 13, e = 37$  ( $ed = 481; ed \bmod 60 = 1$ )
- Let  $M = 15$ . Then  $C \equiv M^e \pmod{n}$ 
  - $C \equiv 15^{37} \pmod{77} = 71$
- $M \equiv C^d \pmod{n}$ 
  - $M \equiv 71^{13} \pmod{77} = 15$

# Why Does RSA Work?

- IT'S MAGIC !



# Why Does RSA Work?

Want to show that  $(M^e)^d \pmod{n} = M$ ,  $n = pq$

$ed \equiv 1 \pmod{\phi(n)}$ , so  $ed = k\phi(n) + 1$ , for some integer  $k$ .

Two cases:

$$\gcd(M, p) = 1$$

Since  $p$  prime, then we have  $M^{p-1} \equiv 1 \pmod{p}$

$$M^{(p-1)k(q-1)} \equiv 1 \pmod{p}$$

$$M * M^{(p-1)k(q-1)} \equiv M \pmod{p}$$

$$M^{ed} \equiv M \pmod{p}$$

$$\gcd(m, p) = p$$

$$M^{ed} \pmod{p} = M \pmod{p} = 0 \text{ so } M^{ed} \equiv M \pmod{p}$$

Similar show that  $M^{ed} \equiv M \pmod{q}$

Since  $p$  and  $q$  are distinct primes we obtain  $M^{ed} \equiv M \pmod{pq}$

# RSA Implementation

- Select  $p$  and  $q$  prime numbers
- In general, select numbers, then test for primality
- Many implementations use the Rabin-Miller test, (probabilistic test)



# RSA Implementation

- Significant improvement in decryption speed for RSA can be obtained by using the Chinese Remainder theorem to work modulo  $p$  and  $q$  respectively.

## Theorem

Let  $n_1, n_2, \dots, n_k$  be integers s.t.  $\gcd(n_i, n_j) = 1$ , where  $i \neq j$ . Then the system

$$x \equiv a_1 \pmod{n_1}$$

$$x \equiv a_2 \pmod{n_2}$$

...

$$x \equiv a_k \pmod{n_k}$$

There exists an integer  $x$  solving the system of simultaneous congruences and all solutions  $x$  are congruent modulo  $n = n_1 n_2 \dots n_k$

# RSA Implementation

- CRT is used in RSA by creating two equations for decryption:

The goal is to compute  $M$ , from  $M = C^d \pmod n$

$$M1 = M \pmod p = C^d \pmod p$$

$$M2 = M \pmod q = C^d \pmod q$$

Fermat theorem on the exponents

$$M1 = C^{d \pmod{(p-1)}} \pmod p$$

$$M2 = C^{d \pmod{(q-1)}} \pmod q$$

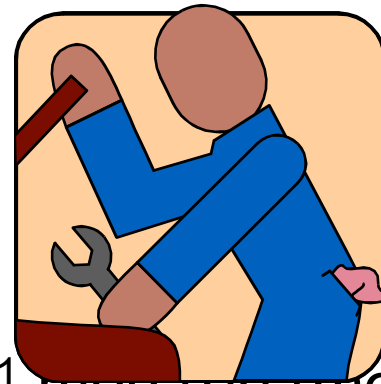
- then the pair of equations

$$M = M1 \pmod p,$$

$$M = M2 \pmod q$$

has a unique solution  $M$ .

$$M = M1(q^{-1} \pmod p)q + M2(p^{-1} \pmod q)p \pmod n$$



# RSA Implementation

E

- e is usually chosen to be 3 or  $2^{16} + 1 = 65537$
- Speed up the encryption, the algorithm is called square-and-multiply (computed exponentiation)
- The smallest the number of 1 bits, the better



# RSA Implementation

N, P, Q

- RSA is usually described as the number of bits for  $n$ . Current recommendation is 1024 bits for  $n$ .
- $P$  and  $q$  must have the same bit length, so for 1024 bits RSA,  $p$  and  $q$  must be about 512 bits.
- $P-q$  should not be small, otherwise  $p \approx \sqrt{n}$
- Some believe that  $p$  and  $q$  should be strong primes ( $p-1$  has a large prime  $r$ ,  $p+1$  has a large prime,  $r-1$  has a large prime)

# RSA and Factoring

- One possible attack against RSA is to factor  $n$
- Factor  $n$ : find  $p$  and  $q$
- Compute  $\phi$
- Compute  $d$
- Breaking RSA is equivalent to factoring  $n$

# Factoring in Practice

- Three methods
  - Quadratic sieve
  - Elliptic curve
  - Number field sieve

- RSA challenges

<http://www.rsasecurity.com/rsalabs/challenges/factoring/numbers.html>

RSA-155 (512 bits) was factored

# Summary

- RSA security relies on the difficulty of factoring big composite numbers
- CRT used to speed up decryption
- $e$  with small number of 1's the speed up exponentiation



# Next ...

- More about attacks on RSA
- Semantic security of RSA
- Rabin Cryptosystem
- Stinson, Chapter 5

