

---

# CSE331: Introduction to Networks and Security

Lecture 25

Fall 2006

# Announcements

---

- Midterm 2 is Friday
  - Covers all material since Midterm 1
  - Including this lecture!
- Project 3 due November 20th
- Project 2: Avg. 74/91
- HW 2: Avg. 30/40

# RSA Key Generation

---

- Choose large primes  $p$  and  $q$ .
  - Should be roughly equal length (in bits)
- Let  $n = p * q$
- Choose a random encryption exponent  $e$ 
  - With requirement:  $e$  and  $(p-1)*(q-1)$  are relatively prime.
- Derive the decryption exponent  $d$ 
  - $d = e^{-1} \text{ mod } ((p-1)*(q-1))$
  - $d$  is  $e$ 's inverse mod  $((p-1)*(q-1))$
- Public key:  $K = (e, n)$  pair of  $e$  and  $n$
- Private key:  $k = (d, n)$
- Discard primes  $p$  and  $q$  (they're not needed anymore)

# RSA Encryption and Decryption

- Message:  $m$
- Assume  $m < n$ 
  - If not, break up message into smaller chunks
  - Good choice: largest power of 2 smaller than  $n$
- Encryption:  $E((e,n), m) = m^e \bmod n$
- Decryption:  $D((d,n), c) = c^d \bmod n$

# Example RSA

- Choose  $p = 47$ ,  $q = 71$
- $n = p * q = 3337$
- $(p-1)*(q-1) = 3220$
- Choose  $e$  relatively prime with 3220:  $e = 79$ 
  - Public key is  $(79, 3337)$
- Find  $d = 79^{-1} \bmod 3220 = 1019$ 
  - Private key is  $(1019, 3337)$
- To encrypt  $m = 688232687966683$ 
  - Break into chunks  $< 3337$
  - 688 232 687 966 683
- Encrypt:  $E((79, 3337), 688) = 688^{79} \bmod 3337 = 1570$
- Decrypt:  $D((1019, 3337), 1570) = 1570^{1019} \bmod 3337 = 688$

# Euler's *totient* function: $\phi(n)$

---

- $\phi(n)$  is the number of positive integers less than  $n$  that are relatively prime to  $n$ 
  - $\phi(12) = 4$
  - Relative primes of 12 (less than 12):  $\{1, 5, 7, 11\}$
- For  $p$  a prime,  $\phi(p) = p-1$ . Why?
- For  $p, q$  two primes,  $\phi(p \cdot q) = (p-1) \cdot (q-1)$

# Fermat's Little Theorem

---

- Generalized by Euler.
- Theorem: If  $\gcd(a,n) = 1$  then  $a^{\phi(n)} \bmod n = 1$ .
- Easy to compute  $a^{-1} \bmod n$ 
  - $a^{-1} \bmod n = a^{\phi(n)-1} \bmod n$
  - Why?  $a * a^{\phi(n)-1} \bmod n$ 
    - =  $a^{\phi(n)-1+1} \bmod n$
    - =  $a^{\phi(n)} \bmod n$
    - = 1

# Example of Fermat's Little Theorem

- What is the inverse of 5, modulo 7?
- 7 is prime, so  $\phi(7) = 6$
- $5^{-1} \pmod{7}$ 
  - =  $5^{6-1} \pmod{7}$
  - =  $5^5 \pmod{7}$
  - =  $((25 \pmod{7} * 5^3 \pmod{7}) \pmod{7})$
  - =  $(4 \pmod{7} * 5^3 \pmod{7}) \pmod{7}$
  - =  $((4 \pmod{7} * 4 \pmod{7}) \pmod{7} * 5 \pmod{7}) \pmod{7}$
  - =  $(16 \pmod{7} * 5 \pmod{7}) \pmod{7}$
  - =  $(2 * 5) \pmod{7}$
  - =  $10 \pmod{7}$
  - = 3

# Chinese Remainder Theorem

- Suppose:
  - $p$  and  $q$  are relatively prime
  - $a \equiv b \pmod{p}$
  - $a \equiv b \pmod{q}$
- Then:  $a \equiv b \pmod{p^*q}$
- Proof:
  - $p$  divides  $(a-b)$  (because  $a \pmod{p} = b \pmod{p}$ )
  - $q$  divides  $(a-b)$
  - Since  $p, q$  are relatively prime,  $p^*q$  divides  $(a-b)$
  - But that is the same as:  $a \equiv b \pmod{p^*q}$

# Proof that D inverts E

$$\begin{aligned} & c^d \bmod n \\ &= (m^e)^d \bmod n && \text{(definition of } c\text{)} \\ &= m^{ed} \bmod n && \text{(arithmetic)} \\ &= m^{k*(p-1)*(q-1) + 1} \bmod n && \text{(d inverts e)} \\ &= m * m^{k*(p-1)*(q-1)} \bmod n && \text{(arithmetic)} \\ &= m * 1 \bmod n && \text{(C. R. theorem)} \\ &= m && \text{(} m < n \text{)} \end{aligned}$$

# Finished Proof

---

- Note:  $m^{p-1} \equiv 1 \pmod{p}$  (if  $p$  doesn't divide  $m$ )
  - Why? Fermat's little theorem.
- Implies:  $m^{k*\phi(n)+1} \equiv m \pmod{p}$ 
  - Also holds for  $m = a*p$
- Same argument yields:  $m^{k*\phi(n)+1} \equiv m \pmod{q}$
- Chinese Remainder Theorem implies:  
 $m^{k*\phi(n)+1} \equiv m \pmod{n}$

# How to Generate Prime Numbers

- Many strategies, but *Rabin-Miller* primality test is most used in practice.
- Efficiently checkable test that, with probability  $\frac{3}{4}$ , verifies that a number  $p$  is prime.
  - Iterate the Rabin-Miller primality test  $t$  times.
  - Probability that a composite number will slip through the test is  $(\frac{1}{4})^t$
  - These are worst-case assumptions.
- In practice (takes several seconds to find a 512 bit prime):
  1. Generate a random  $n$ -bit number,  $p$
  2. Set the high and low bits to 1 (to ensure it is the right number of bits and odd)
  3. Check that  $p$  isn't divisible by any "small" primes  $3, 5, 7, \dots, < 2000$
  4. Perform the Rabin-Miller test at least 5 times.



# Diffie-Hellman Key Exchange

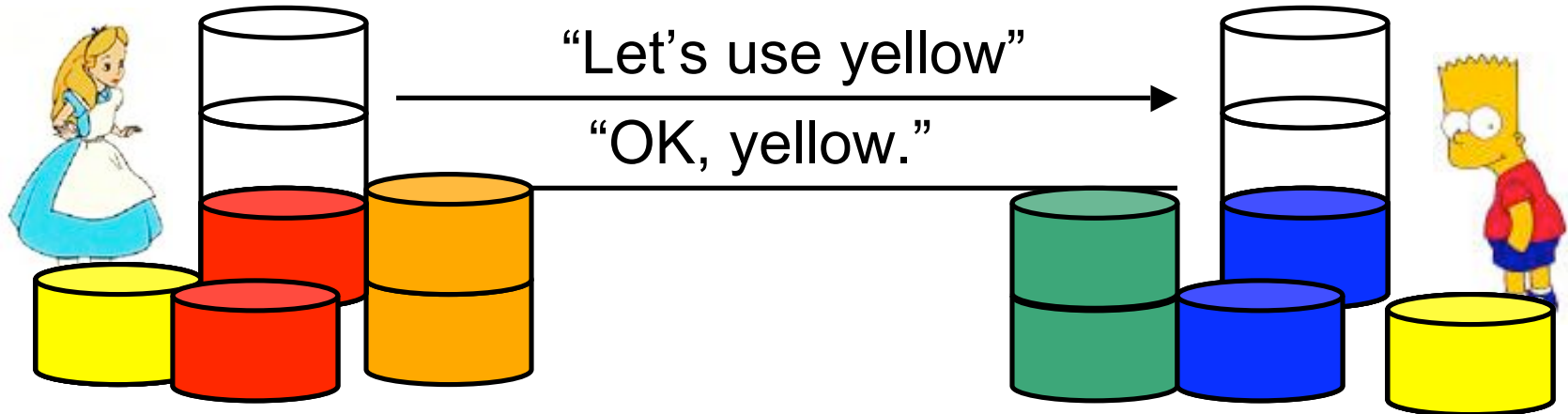
---

- Problem with shared-key systems:  
Distributing the shared key
- Suppose that Alice and Bart want to agree on a secret (i.e. a key)
  - Communication link is public
  - They don't already share any secrets

# Diffie-Hellman by Analogy: Paint

Alice

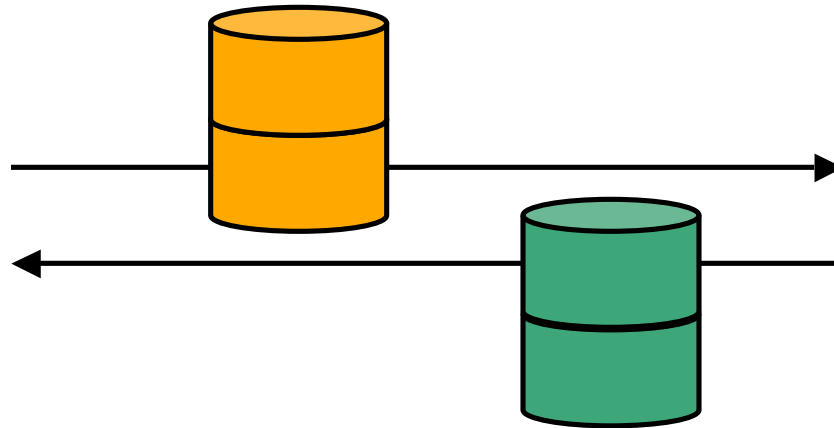
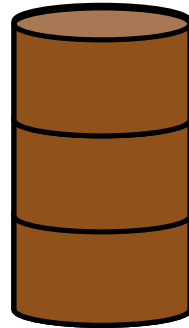
Bart



1. Alice & Bart decide on a public color, and mix one liter of that color.
2. They each choose a random secret color, and mix two liters of their secret color.
3. They keep one liter of their secret color, and mix the other with the public color.

# Diffie-Hellman by Analogy: Paint

Alice



Bart



4. They exchange the mixtures over the public channel.
5. When they get the other person's mixture, they combine it with their retained secret color.
6. The secret is the resulting color: Public + Alice's + Bart's

# Diffie-Hellman Key Exchange

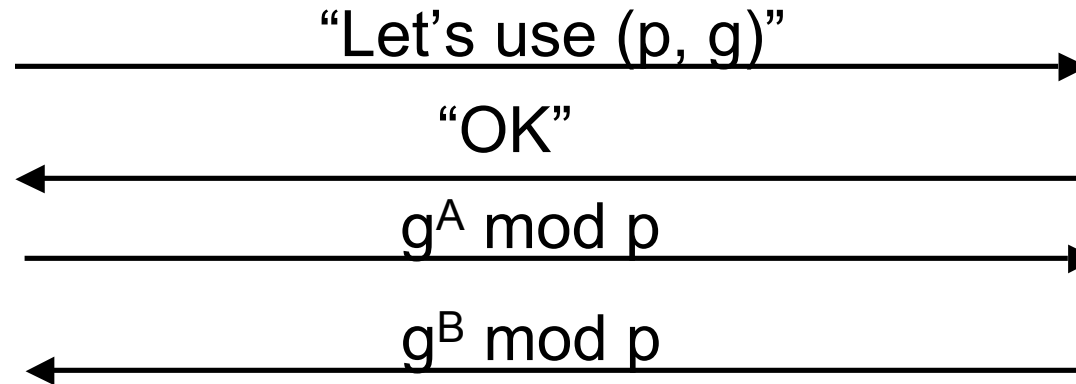
- Choose a prime  $p$  (publicly known)
  - Should be about 512 bits or more
- Pick  $g < p$  (also public)
  - $g$  must be a *primitive root* of  $p$ .
  - A primitive root generates the finite field  $p$ .
  - Every  $n$  in  $\{1, 2, \dots, p-1\}$  can be written as  $g^k \pmod p$
  - Example: 2 is a primitive root of 5
  - $2^0 = 1$      $2^1 = 2$      $2^2 = 4$      $2^3 = 3 \pmod 5$
  - Intuitively means that it's hard to take logarithms base  $g$  because there are many candidates.

# Diffie-Hellman

Alice



Bart



1. Alice & Bart decide on a public prime  $p$  and primitive root  $g$ .
2. Alice chooses secret number  $A$ . Bart chooses secret number  $B$
3. Alice sends Bart  $g^A \bmod p$ .
4. The shared secret is  $g^{AB} \bmod p$ .

# Details of Diffie-Hellman

- Alice computes  $g^{AB} \bmod p$  because she knows  $A$ :
  - $g^{AB} \bmod p = (g^B \bmod p)^A \bmod p$
- An eavesdropper gets  $g^A \bmod p$  and  $g^B \bmod p$ 
  - They can easily calculate  $g^{A+B} \bmod p$  but that doesn't help.
  - The problem of computing discrete logarithms (to recover  $A$  from  $g^A \bmod p$ ) is hard.

# Example

---

- Alice and Bart agree that  $q=71$  and  $g=7$ .
- Alice selects a private key  $A=5$  and calculates a public key  $g^A \equiv 7^5 \equiv 51 \pmod{71}$ . She sends this to Bart.
- Bart selects a private key  $B=12$  and calculates a public key  $g^B \equiv 7^{12} \equiv 4 \pmod{71}$ . He sends this to Alice.
- Alice calculates the shared secret:  
 $S \equiv (g^B)^A \equiv 4^5 \equiv 30 \pmod{71}$
- Bart calculates the shared secret  
 $S \equiv (g^A)^B \equiv 51^{12} \equiv 30 \pmod{71}$

# Why Does it Work?

---

- Security is provided by the difficulty of calculating discrete logarithms.
- Feasibility is provided by
  - The ability to find large primes.
  - The ability to find primitive roots for large primes.
  - The ability to do efficient modular arithmetic.
- Correctness is an immediate consequence of basic facts about modular arithmetic.