CIS 551 / TCOM 401

Computer and Network Security

Spring 2009

Lecture 16

Announcements

- Plan for Today:
  - Key exchange
  - Public Key Cryptography
- Project 3 is due 6 April 2009 at 11:59 pm
  - Handout for SDES available in class today
  - Please read the project description *BEFORE* looking at the code
- Stefan Savage "Spamalytics: Exploring the Technical and Economic Underpinnings of Bulk E-mail Scams"
  - TODAY: at 3:00 p.m. in Wu & Chen Auditorium

Problems with Shared Key Crypto

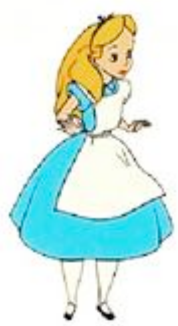
- Compromised key means interceptors can decrypt any ciphertext they've acquired.
  - Change keys frequently to limit damage
- Distribution of keys is problematic
  - Keys must be transmitted securely
  - Use couriers?
  - Distribute in pieces over separate channels?
- Number of keys is $O(n^2)$ where n is # of participants
- Potentially easier to break?

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, …, p-1} can be written as g^k mod p
  - Example: 2 is a primitive root of 5
  - $2^0 = 1 \quad 2^1 = 2 \quad 2^2 = 4 \quad 2^3 = 3 \pmod 5$
  - Intuitively means that it's hard to take logarithms base g because there are many candidates.

Diffie-Hellman

Alice



Bart

"Let's use (p, g)" →

← "OK"

$g^A \bmod p$ →

← $g^B \bmod p$

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} mod p because she knows A:
  - g^{AB} mod p = $(g^B \bmod p)^A$ mod p

- An eavesdropper gets g^A mod p and g^B mod p
  - They can easily calculate g^{A+B} mod p but that doesn't help.
  - The problem of computing discrete logarithms (to recover A from g^A mod 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.

One-way Functions

- A function is one-way if it's
  - Easy to compute
  - Hard to invert (in the average case)
- Examples
  - Exponentiation vs. Discrete Log
  - Multiplication vs. Factoring
  - Knapsack Packing
    - Given a set of numbers {1, 3, 6, 8, 12} find the sum of a subset
    - Given a target sum, find a subset that adds to it
- Trapdoor functions
  - Easy to invert given some extra information
  - E.g. factoring p*q given q

Public Key Cryptography

- Sender encrypts using a *public key*
- Receiver decrypts using a *private key*
- Only the private key must be kept secret
  - Public key can be distributed at will
- Also called *asymmetric* cryptography
- Can be used for digital signatures
- Examples: RSA, El Gamal, DSA, various algorithms based on elliptic curves

- Used in SSL, ssh, PGP, …

Public Key Notation

- Encryption algorithm

  E : keyPub x plain $\rightarrow$ cipher

  Notation: K{msg} = E(K, msg)
- Decryption algorithm

  D : keyPriv x cipher $\rightarrow$ plain

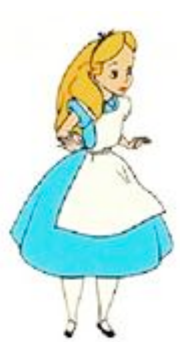
  Notation: k{msg} = D(k,msg)
- D inverts E

  D(k, E(K, msg)) = msg
- Use capital “K” for public keys
- Use lower case “k” for private keys
- Sometimes E is the same algorithm as D

Secure Channel: Private Key

Alice

Bart



$K_B\{\text{Hello!}\}$ $\rightarrow$

$\leftarrow$ $K_A\{\text{Hi!}\}$

K_A,K_B
k_A

K_A,K_B
k_B

Trade-offs for Public Key Crypto

- More computationally expensive than shared key crypto
  - Algorithms are harder to implement
  - Require more complex machinery
- More formal justification of difficulty
  - Hardness based on complexity-theoretic results
- A principal needs one private key and one public key
  - Number of keys for pair-wise communication is O(n)

RSA Algorithm

- Ron Rivest, Adi Shamir, Leonard Adleman
  - Proposed in 1979
  - They won the 2002 Turing award for this work
- Has withstood years of cryptanalysis
  - Not a guarantee of security!
  - But a strong vote of confidence.
- Hardware implementations: 1000 x slower than DES

RSA at a High Level

- Public and private key are derived from secret prime numbers
  - Keys are typically ≥ 1024 bits
- Plaintext message (a sequence of bits)
  - Treated as a (large!) binary number
- Encryption is modular exponentiation
- To break the encryption, conjectured that one must be able to factor large numbers
  - Not known to be in P (polynomial time algorithms)
  - Is known to be in BQP (bounded-error, quantum polynomial time – Shor's algorithm)

Number Theory: Modular Arithmetic

- Examples:
  - 10 mod 12 = 10
  - 13 mod 12 = 1
  - (10 + 13) mod 12 = 23 mod 12 = 11 mod 12
  - $23 \equiv 11 \pmod{12}$
  - "23 is congruent to 11 (mod 12)"

- $a \equiv b \pmod{n}$ iff $a = b + kn$ (for some integer k)
- The *residue* of a number modulo n is a number in the range 0…n-1

Number Theory: Prime Numbers

- A *prime number* is an integer > 1 whose only factors are 1 and itself.

- Two integers are *relatively prime* if their only common factor is 1
  - gcd = greatest common divisor
  - gcd(a,b) = 1
  - gcd(15,12) = 3, so they're not relatively prime
  - gcd(15,8) = 1, so they are relatively prime

- Easy to compute GCD using Euclid's Algorithm

Finite Fields (Galois Fields)

- For a prime p, the set of integers mod p forms a *finite field*
- Addition + Additive unit 0
- Multiplication * Multiplicative unit 1
- Inverses: $n * n^{-1} = 1$ for $n \neq 0$
  - Suppose p = 5, then the finite field is {0,1,2,3,4}
  - $2^{-1} = 3$ because $2 * 3 \equiv 1 \bmod 5$
  - $4^{-1} = 4$ because $4 * 4 \equiv 1 \bmod 5$

- Usual laws of arithmetic hold for modular arithmetic:
  - Commutativity, associativity, distributivity of * over +

RSA Key Generation

- Choose large, distinct 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
  - $d = e^{-1} \bmod ((p-1)*(q-1))$
  - 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} mod 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} mod 3337 = 1570
- Decrypt: D((1019, 3337), 1570) = 1570^{1019} mod 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 distinct primes, $\phi(p*q) = (p-1)*(q-1)$
  - There's p*q-1 numbers less than p*q
  - Factors of p*q =
    - {1*p, 2*p, …, q*p} for a total of q of them
    - {1*q, 2*q, …, p*q} for another p of them
    - No other numbers
    - $\phi(p*q) = (p*q) - (p + q - 1) = pq - p - q + 1 = (p-1)*(q-1)$

q many multiples of p

All #s ≤ p*q

don't double count p*q

P many multiples of q

Fermat's Little Theorem

- Generalized by Euler.
- Theorem: If p is a prime then $a^p \equiv a \bmod p$.
- Corollary: If $\gcd(a,n) = 1$ then $a^{\phi(n)} \equiv 1 \bmod n$.
- 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 $\phi(7) = 6$
- $5^{-1} \bmod 7 = 5^{6-1} \bmod 7$

$= 5^5 \bmod 7$

$= (5^2 * 5^2 * 5) \bmod 7$

$= ((5^2 \bmod 7) * (5^2 \bmod 7) * (5 \bmod 7)) \bmod 7$

$= ((4 \bmod 7) * (4 \bmod 7) * (5 \bmod 7)) \bmod 7$

$= ((16 \bmod 7) * (5 \bmod 7)) \bmod 7$

$= ((2 \bmod 7) * (5 \bmod 7)) \bmod 7$

$= (10 \bmod 7) \bmod 7$

$= 3 \bmod 7$

$= 3$

Chinese Remainder Theorem

- (Or, enough of it for our purposes…)

- 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 mod p = b mod 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

$c^d \bmod n$

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

$e*d \equiv 1 \bmod (p-1)*(q-1)$

Finished Proof

- Note: $m^{p-1} \equiv 1 \bmod p$ (if p doesn't divide m)
  - Why? Fermat's little theorem.
- Same argument yields: $m^{q-1} \equiv 1 \bmod q$

- Implies: $m^{k*\phi(n)+1} \equiv m \bmod p$
- And $m^{k*\phi(n)+1} \equiv m \bmod q$

- Chinese Remainder Theorem implies:
  $m^{k*\phi(n)+1} \equiv m \bmod n$

How to Generate Prime Numbers

- Many strategies, but *Rabin-Miller* primality test is often used in practice.
  - $a^{p-1} \equiv 1 \bmod p$
- Efficiently checkable test that, with probability ¾, 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 $(1/4)^t$
  - 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,…,<2000
4. Perform the Rabin-Miller test at least 5 times.

Rabin-Miller Primality Test

- Is n prime?
- Write n as $n = (2^r)*s + 1$
- Pick random number a, with $1 \leq a \leq n - 1$
- If
  - $a^s \equiv 1 \bmod n$ and
  - for all j in {0 … r-1}, $a^{2^j s} \equiv -1 \bmod n$
- Then return composite
- Else return probably prime