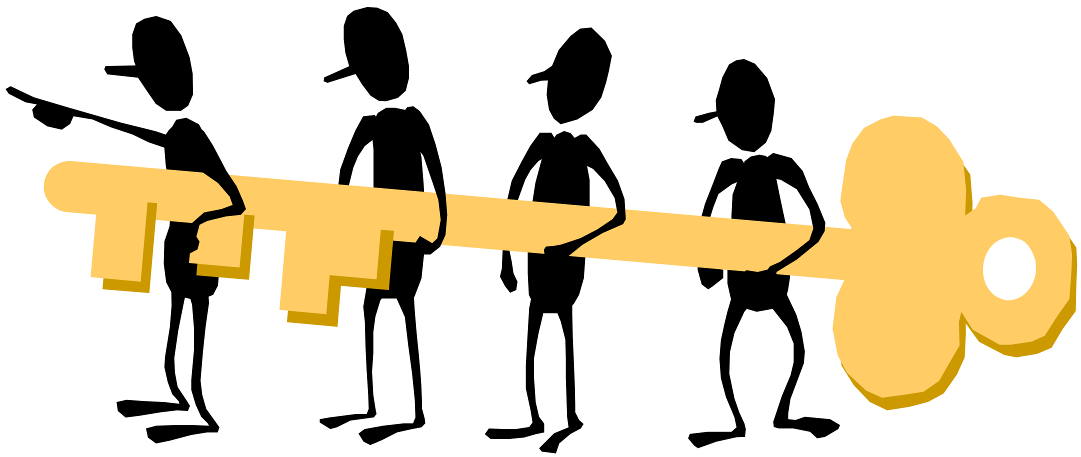
Public Key Algorithms



Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

Audio/Video recordings of this lecture are available at:

http://www.cse.wustl.edu/~jain/cse571-09/

1. Number Theory
2. RSA Public Key Encryption
3. Public-Key Cryptography Standards (PKCS)
4. Diffie-Hellman Key Agreement
5. Digital Signature Standard
6. Elliptic Curve Cryptography (ECC)
7. Zero-Knowledge Proof Systems

Public Key Encryption

- Invented in 1975 by Diffie and Hellman
- Encrypted_Message = Encrypt(Key1, Message)
- Message = Decrypt(Key2, Encrypted_Message)

$$\text{Text} \xrightarrow{\text{Key1}} \text{Ciphertext}$$

$$\text{Ciphertext} \xrightarrow{\text{Key2}} \text{Text}$$

Public Key Encryption Example

- Rivest, Shamir, and Adleman
- RSA: Encrypted_Message = m^3 mod 187
- Message = $\text{Encrypted\_Message}^{107}$ mod 187
- Key1 = <3,187>, Key2 = <107,187>
- Message = 5
- Encrypted Message = $5^3 = 125$
- Message = 125^{107} mod 187 = 5
  = $125^{(64+32+8+2+1)}$ mod 187
  = $\{(125^{64} \bmod 187)(125^{32} \bmod 187)\ldots$
  $(125^2 \bmod 187)(125 \bmod 187)\} \bmod 187$

Modular Arithmetic

- $xy \bmod m = (x \bmod m)\,(y \bmod m) \bmod m$
- $x^4 \bmod m = (x^2 \bmod m)(x^2 \bmod m) \bmod m$
- $x^{ij} \bmod m = (x^i \bmod m)^j \bmod m$
- $125 \bmod 187 = 125$
- $125^2 \bmod 187 = 15625 \bmod 187 = 104$
- $125^4 \bmod 187 = (125^2 \bmod 187)^2 \bmod 187$
  $= 104^2 \bmod 187 = 10816 \bmod 187 = 157$
- $128^8 \bmod 187 = 157^2 \bmod 187 = 152$
- $128^{16} \bmod 187 = 152^2 \bmod 187 = 103$
- $128^{32} \bmod 187 = 103^2 \bmod 187 = 137$
- $128^{64} \bmod 187 = 137^2 \bmod 187 = 69$
- $128^{64+32+8+2+1} \bmod 187 = 69\times137\times152\times104\times125 \bmod 187$
  $= 18679128000 \bmod 187 = 5$

Definitions

- Co-Prime = Relatively Prime:
  x and y are relatively prime if gcd(x,y)=1
  - Example: 6, 11
- Inverse: x is inverse of y if xy=1 mod n
  - If $ux + vn = 1$, $x^{-1} = u \bmod n$
  - Example: what is inverse of 6 mod 11
    - $2 \times 6 - 1 \times 11 = 1 \Rightarrow$ Inverse of 6 mod 11 is 2.
- Smooth Number = Product of small primes

Euclid's Algorithm

- Goal: To find greatest common divisor

Example: gcd(10,25)=5 using long division

```
10) 25 (2
    20
    --
  5)10 (2
    10
     --
    00
```

Euclid's Algorithm: Tabular Method

		10	25
q_i	r_i	u_i	v_i
0	25	0	1
0	10	1	0
2	5	-2	1
2	0	5	-2

- $r_i = u_i x + v_i y$
- $u_i = u_{i-2} - q_i u_{i-1}$
- $v_i = v_{i-2} - q_i v_{i-1}$
- Finally, If $r_i = 0$, $\gcd(x,y) = r_{i-1}$
- If $r_i = 1$, $u_i x + v_i y = 1 \Rightarrow x^{-1} \bmod y = u_i$

Chinese Remainder Theorem

- The solution to the following equations:

  $x = a_1 \bmod n_1$

  $x = a_2 \bmod n_2$

  $x = a_k \bmod n_k$

  where $n_1, n_2, \ldots, n_k$ are relatively prime is found as follows:

  $N = n_1\, n_2 \ldots n_k$

  $N_i = N/n_i$

  Find s_i such that $r_i\, n_i + s_i\, N_i = 1$

  Let $e_i = s_i\, N_i$, then

  $x = \sum_i^k a_i\, e_i \bmod N$

Chinese Remainder Theorem (Cont)

- Example: Solve the equations:
- $x = 3 \bmod 6$
- $x = 6 \bmod 7$
- $x = 10 \bmod 11$
- $N = 6 \times 7 \times 11 = 462$
- gcd(6,77): $13 \times 6 - 1 \times 77 = 1 \Rightarrow e_1 = -77$
- gcd(7,66): $19 \times 7 - 2 \times 66 = 1 \Rightarrow e_2 = -132$
- gcd(11,42): $-19 \times 11 + 5 \times 42 \Rightarrow e_3 = 210$
- $x = 3 \times (-77) + 6 \times (-132) + 10 \times 210 = -231 - 792 + 2100 = 1077 \bmod 462 = 153$

		7	66
q_i	r_i	u_i	v_i
0	66	0	1
0	7	1	0
9	3	-9	1
2	1	19	-2

Euler's Totient Function

- Z_n = Set of all numbers mod n = {0, 1, 2, …, n-1}
- Z_n* = Set of all numbers relatively primes to n
- $\Phi(n)$ = "Phi(n)" = Number of elements in Z_n*
- If n is prime, $\Phi(n)=n-1$
- Example:

  $Z_{10}=\{0, 1, 2, 3, ..., 9\}$

  $Z_{10}^* = \{1, 3, 7, 9\}$

  $\Phi(10)=4$

Euler's Theorem

- For all $a \in Z_n^*$ $a^{\Phi(n)} = 1 \bmod n$
- For all $a \in Z_n^*$ $a^{(k\Phi(n)+1)} = a \bmod n$
- Examples:
  - $z_{10}^* = \{1, 3, 7, 9\}$
  - $\Phi(10)=4$
  - $1^4 \bmod 10 = 1$
  - $3^4 \bmod 10 = 1$
  - $7^4 \bmod 10 = 1$
  - $9^4 \bmod 10 = 1$

Fermat's Theorem

- If p is prime and $0<a<p$, $a^{p-1} \bmod p = 1$
- Example:
  - $2^6 \bmod 7 = 64 \bmod 7 = 1$
  - $3^4 \bmod 5 = 81 \bmod 5 = 1$
  - This is a necessary condition. Not sufficient.
  - $a^{p-1} \bmod p = 1$ for all a $\not\Rightarrow$ p is prime
  - **Carmichael Numbers** or pseudo-primes
  - Example: $561 = 3 \times 11 \times 17$

Miller and Rabin Method Prime Test

- Express n-1 as $2^b c$ where *c* is odd.
- Pick a random *a*.
- $a^{n-1} = a^{2^b c} = (a^c)^{2^b}$
- Compute a^c mod n
- Square it b times: a^{2c}, a^{4c}, a^{8c}, ..., $a^{\{2^b\}c}$
- If the final result is not one $\Rightarrow$ n is not a prime
- If any of the intermediate results is 1, check if the previous number is $\pm$ 1.
- If $\pm$ 1 then n is potentially prime.
- Pick another *a* and try again.

RSA Public Key Encryption

- Ron Rivest, Adi Shamir, and Len Adleman at MIT 1978
- Both plain text M and cipher text C
  are integers between 0 and n-1.
- Key 1 = {e, n},
  Key 2 = {d, n}
- $C = M^e \bmod n$
  $M = C^d \bmod n$
- How to construct keys:
  - Select two large primes: p, q, $p \neq q$
  - $n = p \times q$
  - Calculate Euler's Totient Fn $\Phi(n) = (p-1)(q-1)$
  - Select e relatively prime to $\Phi \Rightarrow \gcd(\Phi, e) = 1$; $0 < e < \Phi$
  - Calculate d = inverse of e mod $\Phi \Rightarrow de \bmod \Phi = 1$
  - Euler's Theorem: $x^{ed} = x^{k\Phi(n)+1} = x \bmod n$

RSA Key Construction: Example

- Select two large primes: p, q, $p \neq q$
  $p = 17, q = 11$
- $n = p \times q = 17 \times 11 = 187$
- Calculate $\Phi = (p-1)(q-1) = 16x10 = 160$
- Select e, such that $lcd(\Phi, e) = 1; 0 < e < \Phi$
  say, $e = 7$
- Calculate d such that de mod $\Phi = 1$
  - $160k+1 = 161, 321, 481, 641$
  - Check which of these is divisible by 7
  - 161 is divisible by 7 giving $d = 161/7 = 23$
  - Euclid's algorithm is a better way to find this
- Key 1 = {7, 187}, Key 2 = {23, 187}

RSA Issues

- RSA is computationally intense.
- Commonly used key lengths are 512 bits
- The plain text should be smaller than the key length
- The encrypted text is same size as the key length
- Generally used to encrypt secret keys.
- Basis: Factoring a big number is hard

Finding d and e

- $de = 1 \mod \Phi(n)$
- Select e first, e.g., $e=2^1+1$ or $2^{16}+1$
  $\Rightarrow$ Exponentiation is easy.
- Find inverse of e using Euclid's algorithm
- The public key can be small.
- The private key should be large. $\Rightarrow$ Don't select d=3.
- Both d and n are 512 bit (150 digits) numbers.

Optimizing Private Key Operations

1. $c^d \bmod n = c^d \bmod pq$
   - Compute $c^d \bmod p$ and $c^d \bmod q$
   - Use Chinese remainder theorem to compute $c^d \bmod pq$
2. Chinese remainder theorem requires $p^{-1} \bmod q$ and $q^{-1} \bmod p$. Compute them once and store.
3. Since d is much bigger than p, $c^d \bmod p = c^r \bmod p$ where $r= d \bmod (p-1)$
   - $d = k(p-1)+r$
   - Mod p: $a^d = a^{k(p-1)+r} = a^{k(\Phi(p)} a^r = a^r$ [Euler's Theorem]

Attacks on RSA

- Smooth Number Attack:
  - If you sign m_1 and m_2
  - $S_1 = m_1^d \bmod n$
  - $S_2 = m_2^d \bmod n$
  - Attacker can sign m_1m_2, m_1/m_2, m_1^2, $m_1^jm_2^k$
  - Easy to do if m_i's are small (smooth) numbers.
- Cube Root Problem of RSA
  - If public exponent e=3:
  - $h^{de} \bmod n = h$
  - $h^d \bmod n = h^{1/3}$
  - Simply compute $h^{1/3} \bmod n$

Public-Key Cryptography Standards

- RSA Inc developed standards on how to use public key cryptography
- Specify encoding of keys, signatures, etc.
- PKCS #1: Formatting a message for RSA encryption

0	2	At least 8 Random nonzero octets	0	Data

- First octet = 0 ⇒ m < n
- Second Octet = Format Type. 2 ⇒ Encryption
- Random non-zero padding ⇒ cipher is different
- Zero ends the padding
- PKCS Signing

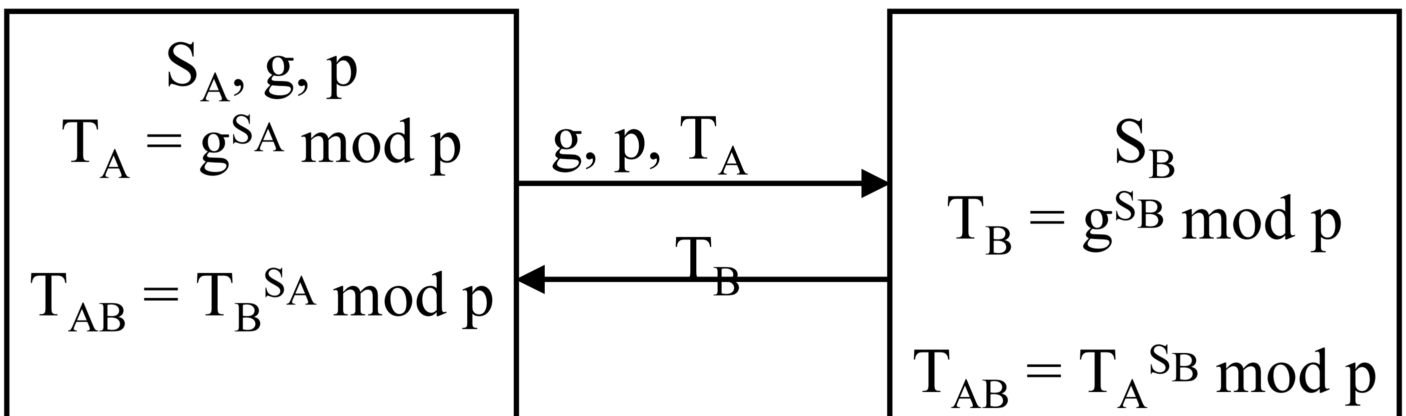
0	1	At least 8 octets of FF_{16}	0	Data

Million Message Attack on RSA

- In SSL if padding is incorrect, some Servers send "Incorrect Padding" error
- Attacker sends random variations until server accepts (decrypted message begins with 02)
- Would need 2^{16} tries to get the correct 16 bits
- PKCS#1 Rev 2 fixes this problem.
- Not sending error message is easier fix.

Diffie-Hellman Key Agreement

- Allows two party to agree on a secret key using a public channel
- A selects p=large prime, and g=a number less than p
- A selects a random # S_A, B selects another random # S_B



$T_{AB} = g^{S_A S_B} \bmod p$

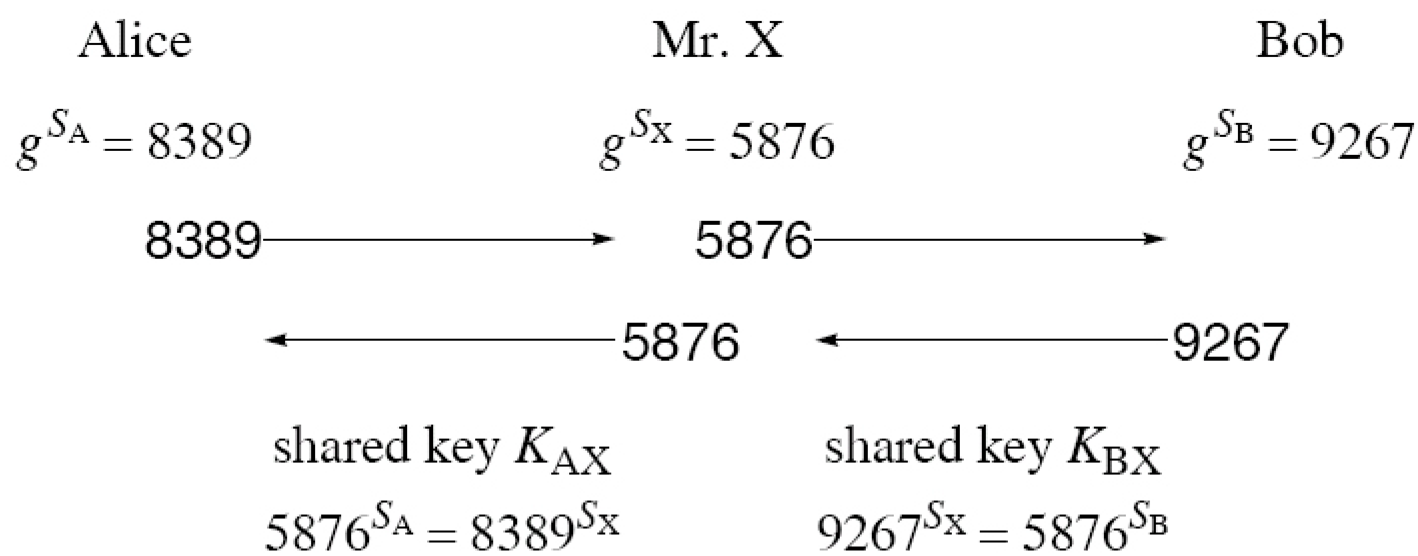
- Eavesdropper can see T_A, g, p but cannot compute S_A
- Computing S_A requires discrete logarithm - a difficult problem

Diffie-Hellman (Cont)

- Example: g=5, p=19
  - A selects 6 and sends 5^6 mod 19 = 7
  - B selects 7 and sends 5^7 mod 19 = 16
  - A computes K = 16^6 mod 19 = 7
  - B computes K = 7^7 mod 19 = 7
- Preferably (p-1)/2 should also be a prime.
- Such primes are called safe prime.

Man-in-Middle Attack on Diffie-Hellman

- Diffie-Hellman does not provide authentication



- You can use RSA authentication and other alternatives

ElGamal Signatures

- Similar to Diffie-Hellman
- Public key: (g, p, T), $T=g^S$ mod p; Private key: S

Signature:

- Choose a random S_m, $0<S_m<p-1$ and $\gcd(S_m,p-1)=1$
- Compute $T_m=g^{Sm}$ mod p
- Compute $X = (H(m|T_m)S_m+T_m) \bmod (p-1)$
- If X=0 start over again
- The pair T_m,X is the signature.

Verification: Compute $H(m|T_m)$ and g^X and verify:

- $g^X = T_m T^{H(m|Tm)}$ (since $T=g^S$)
- Note: Each message needs a different per message key S_m. Two keys: S is the long term key, S_m is the per message key.
- If the same key is used on many messages, S can be obtained.

Digital Signature Standard

- FIPS 186 in 1991, 186-1 in 1993, 186-2 in 2000.
- A variation of ElGamal signature
- Choose a hash. Default = SHA-1
- Select a key size L: multiple of 64 between 512 to 1024.
- 186-2 requires 1024.
- 186-3 recommends 2048 or 3072 for lifetimes beyond 2010.

1. Algorithm Parameters:

- Choose a prime q with the same number of bits as hash
- Select a L-bit prime p such that p-1 is a multiple of q
- Select a generator g such that $g^q = 1 \bmod p$
- This can be done by $g=h^{(p-1)/q} \bmod p$ for some arbitrary h $1<h<p-1$.
- Algorithm parameters (p, q, g) may be shared among users.

DSS (Cont)

2. User Keys: public and private key for a user

- Choose S randomly $0 < S < p$
- $T = g^S \bmod p$
- Public key is (p, q, g, T). Private key is S.

3. Signing: Generate per message key S_m, $0<S_m<q$

- $T_m = (g^{S_m} \bmod p) \bmod q$
- Compute $S_m^{-1} \bmod q$
- Calculate message digest d_m
- Signature $X = S_m^{-1} (d_m + ST_m) \bmod q$
- Transmit message m, per message public number T_m, and signature X

DSS (Cont)

4. Verification:

- Calculate inverse of signature X^{-1} mod q
- Calculate message digest d_m
- Calculate $x = d_m X^{-1} \bmod q$
- $y = T_m x^{-1} \bmod q$
- $z = (g^x T^y \bmod p) \bmod q$
- If $z = T_m$ then signature is verified.

DSS Insecurity

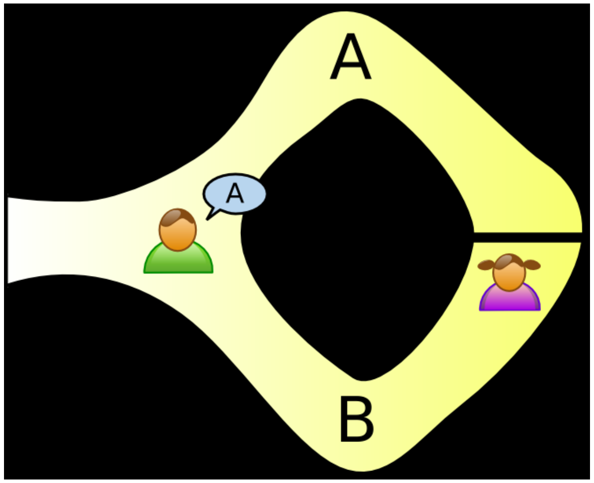
- <p, q, g> is shared.
- Anyone who breaks <p, q, g> can break all the users sharing it.
- Slower than RSA with e=3
- Both RSA and Diffie-Hellman require sub-exponential super-polynomial effort in key size
- Sub-exponential
  $\Rightarrow$ Need very large keys for public key cryptography
  $\Rightarrow$ 1024 for RSA, 80 bit for DES
  $\Rightarrow$ Use secret keys. Use public for key exchange.

Elliptic Curve Cryptography (ECC)

- Based on algebraic structure of elliptic curves over finite fields
- Elliptic curve is a plane curve
- $y^2 + axy + by = x^3 + cx^2 + d\,x + e$
- The set of points on the curve along with the point at infinity form a set over which operations similar to modular arithmetic can be defined.
- Multiplying two points results in a third point.
  The point at infinity is the identity element.
- ECC is still exponential difficulty and so key lengths can be shorter.

Zero-Knowledge Proof Systems

- The verifier can verify that you possess the secret but gets no knowledge of the secret.

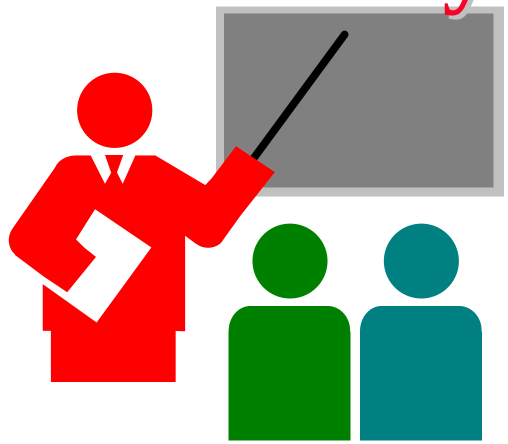


[Source: Wikipedia]

Any NP-complete problem can be used.

- Signature systems are zero-knowledge proof systems

Summary



- Public key cryptography uses two keys: public key and private key
- Modular Arithmetic, Euclid's algorithm, Euler's theorm, Fermat's theorem, Chinese remainder theorem
- RSA is based on difficulty of factorization
- Diffie-Hellman is based on difficulty of discrete logarithms.
- Digital signature standard is similar to Diffie-Hellman

References

- Chapter 6 and 7 of the text book
- Wikepedia entries:
  - http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
  - http://en.wikipedia.org/wiki/Chinese_remainder_theorem
  - http://en.wikipedia.org/wiki/Carmichael_number
  - http://en.wikipedia.org/wiki/PKCS
  - http://en.wikipedia.org/wiki/Diffie-Hellman
  - http://en.wikipedia.org/wiki/ElGamal_signature_scheme
  - http://en.wikipedia.org/wiki/Digital_Signature_Standard
  - http://en.wikipedia.org/wiki/Elliptic_curve_cryptography
  - http://en.wikipedia.org/wiki/Zero_Knowledge

Homework 8

- Read chapter 6 and 7
- 8a. In an RSA system, the public key of a given user is e=31, n=3599. What is the private key of this user.
- 8b. If x = 3 mod 7 = 5 mod 13 = 8 mod 11. What is x? Show all steps.