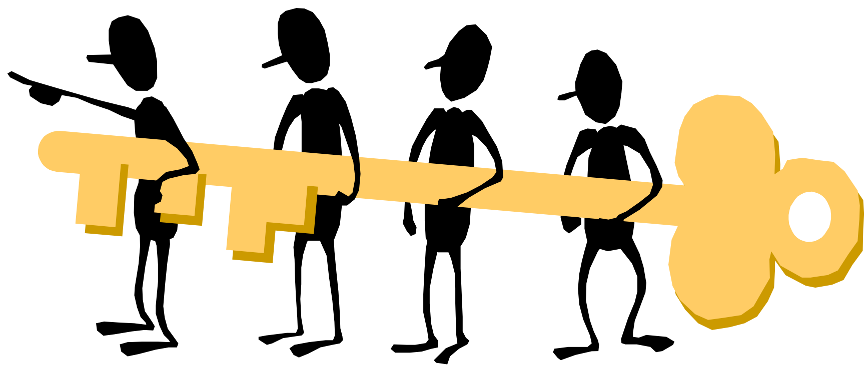
Public Key Cryptography and RSA



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/cse571-11/

1. Public Key Encryption
2. Symmetric vs. Public-Key
3. RSA Public Key Encryption
4. RSA Key Construction
5. Optimizing Private Key Operations
6. RSA Security

These slides are based partly on Lawrie Brown's slides supplied with William Stallings's book "Cryptography and Network Security: Principles and Practice," 5th Ed, 2011.

Public Key Encryption

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

$$\text{Text} \xrightarrow{\text{Key1}} \text{Ciphertext} \xrightarrow{\text{Key2}} \text{Text}$$

- Keys are **interchangeable**:

$$\text{Text} \xrightarrow{\text{Key2}} \text{Ciphertext} \xrightarrow{\text{Key1}} \text{Text}$$

- One key is made **public** while the other is kept **private**
- Sender knows only public key of the receiver ⇒ **Asymmetric**

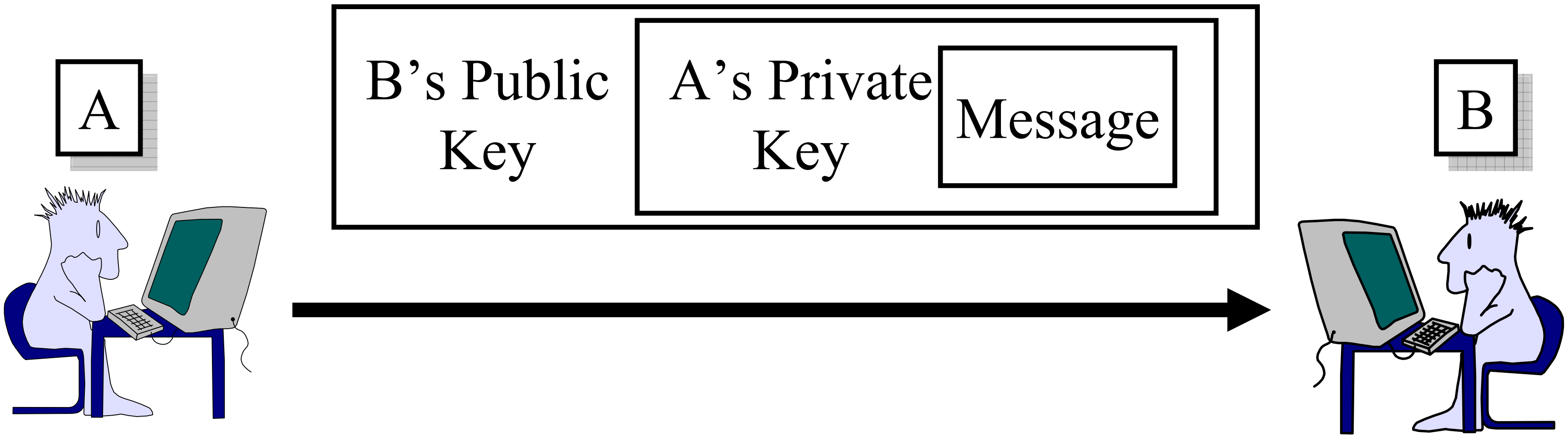
Public Key Encryption Example

- Rivest, Shamir, and Adleman at MIT
- RSA: Encrypted_Message = m^3 mod 187
- Message = 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
  = $\{(125^{64} \bmod 187)(125^{32} \bmod 187)\ldots$
  $(125^{2} \bmod 187)(125 \bmod 187)\} \bmod 187$

Symmetric vs. Public-Key

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Public-Key Authentication and Secrecy



- A encrypts the message with its private key and then with B's public key
- B can decrypt it with its private key and A's public key
- No one else can decrypt ⇒ Secrecy
- No one else can send such a message
  ⇒ B is assured that the message was sent by A
  ⇒ Authentication

Public-Key Applications

- 3 Categories:
  - **Encryption/decryption** (provide secrecy)
  - **Digital signatures** (provide authentication)
  - **Key exchange** (of session keys)
- Some algorithms are suitable for all uses, others are specific to one

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Public-Key Requirements

- Need a trapdoor one-way function
- One-way function has
  - $Y = f(X)$ easy
  - $X = f^{-1}(Y)$ infeasible
- A trap-door one-way function has
  - $Y = f_k(X)$ easy, if k and X are known
  - $X = f_k^{-1}(Y)$ easy, if k and Y are known
  - $X = f_k^{-1}(Y)$ infeasible, if Y known but k not known
- A practical public-key scheme depends on a suitable trap-door one-way function

Security of Public Key Schemes

- Like private key schemes brute force **exhaustive search** attack is always theoretically possible
- But keys used are too large (>512bits)
- Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- More generally the **hard** problem is known, but is made hard enough to be impractical to break
- Requires the use of **very large numbers**
- Hence is **slow** compared to private key schemes

RSA Public Key Encryption

- Ron Rivest, Adi Shamir, and Len Adleman at MIT 1978
- Exponentiation in a Galois field over integers modulo a prime
  - Exponentiation takes $O((\log n)^3)$ operations (easy)
- Security due to cost of factoring large numbers
  - Factorization takes $O(e^{\log n \log \log n})$ operations (hard)
- Plain text M and ciphertext 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: $x^{ed} = x^{k\Phi(n)+1} = x \bmod n$

Finding d and e

- $de = 1 \bmod \Phi(n)$
- Select e first, e.g., $e=2^1+1$, 2^4+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.
  - Can be attacked using Chinese remainder theorem & 3 messages with different moduli
- Both d and n are 512 bit (150 digits) numbers.

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 $\Phi = (p-1)(q-1) = 16 x 10 = 160$
- Select e, such that $lcd(\Phi, e) = 1; 0 < e < \Phi$
  say, $e = 7$
- Calculate d such that $de \bmod \Phi = 1$
  - Use Euclid's algorithm to find $d = e^{-1} \bmod \Phi$
  - $160k+1 = 161, 321, 481, 641$
  - Check which of these is divisible by 7
  - 161 is divisible by 7 giving $d = 161/7 = 23$
- Key 1 = {7, 187}, Key 2 = {23, 187}

Exponentiation

- Can use the Square and Multiply Algorithm
- `E.g., 3`^{129} `= 3`^{128}`.3`^{1} `= 5.3 = 4 mod 11`
- Takes log (b) operations for a^b
- To compute a^b mod n:

Expand b as a binary number: $b_k b_{k-1} \ldots b_2 b_1 b_0$

k= Number of bits in b

c = 0; f = 1
for i = k **downto** 0
do c = 2 × c
f = (f × f) mod n
if b_i == 1 **then**
c = c + 1
f = (f × a) mod n
return f

Excel

a= 125
b= 107
n= 187

j	i=2^j	a^i	a^i mod n	bi	c	a^c mod n
0	1	125	125	1	1	125
1	2	15625	104	1	3	97
2	4	10816	157	0	3	97
3	8	24649	152	1	11	158
4	16	23104	103	0	11	158
5	32	10609	137	1	43	141
6	64	18769	69	1	107	5
7	128	4761	86	0	107	5
8	256	7396	103	0	107	5
9	512	10609	137	0	107	5
10	1024	18769	69	0	107	5

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]

- Only owner of the private key knows p and q and can optimize

RSA Issues

- RSA is computationally intense.
- Commonly used key lengths are 1024 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.
- Potential Attacks:
  1. Brute force key search - infeasible given size of numbers
  2. Timing attacks - on running of decryption
     Can Infer operand size based on time taken
     $\Rightarrow$ Use constant time
  3. Mathematical attacks - based on difficulty of computing ø(n), by factoring modulus n
  4. Chosen ciphertext attacks

Progress in Factoring

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	1000	generalized number field sieve
140	465	February 1999	2000	generalized number field sieve
155	512	August 1999	8000	generalized number field sieve
160	530	April 2003	—	Lattice sieve
174	576	December 2003	—	Lattice sieve
200	663	May 2005	—	Lattice sieve

Ref: The RSA Factoring Challenge FAQ, http://www.rsa.com/rsalabs/node.asp?id=2094

Optimal Asymmetric Encryption Padding (OASP)

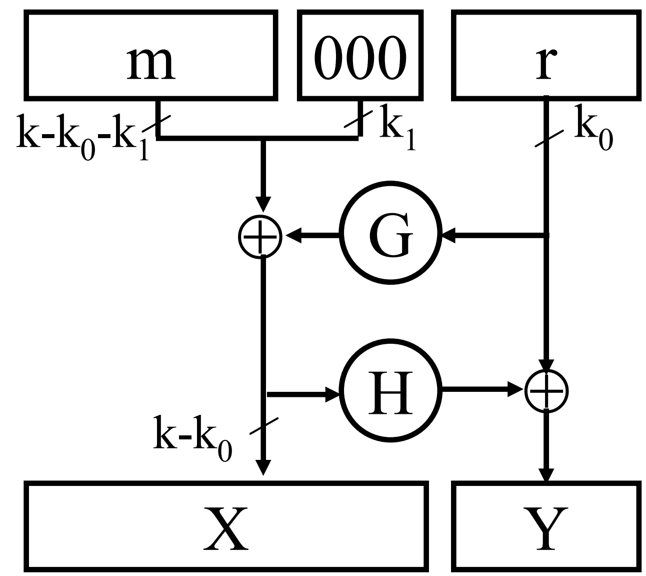
- RSA is susceptible to "Chosen Plaintext Attack"

$$E(PU, M) = M^e \bmod n$$

$$E(PU, M_1) \times E(PU, M_2) = E(PU, M_1 \times M_2)$$

$$E(PU, 2M) = 2^e E(PU, M)$$

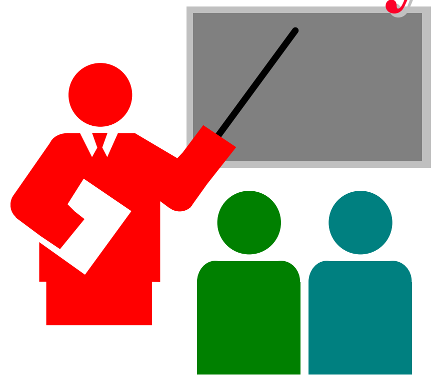
- Submit $2^e \times$ Ciphertext and get back 2M $\Rightarrow$ know Plaintext M
- OASP: Let k =# bits in RSA modulus
  - Plaintext m is $k\text{-}k_0\text{-}k_1$ bit string
  - G and H are Cryptographic fn
    G expands k_0 bits to $k\text{-}k_0$ bits
    H reduces $k\text{-}k_0$ bits to k_0 bits
  - *r* is a random k_0 bit seed



- Need to recover entire X and Y

Ref: http://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

Summary



1. Public key encryption uses two keys: one to encrypt and the other to decrypt. The keys are interchangeable. One key is public. Other is private.
2. RSA uses exponentiation in GF(n) for a large n.
   n is a product of two large primes.
3. RSA keys are <e, n> and <d, n> where ed mod $\Phi(n)=1$
4. Given the keys, both encryption and decryption are easy. But given one key finding the other key is hard.
5. The message size should be less than the key size. Use large keys 512 bits and larger.

Optional Exercises

- 9.2, 9.3, 9.4, 9.8, 9.16, 9.18
- Try on your own. Do not submit.