Cristina Nita-Rotaru

CS355: Cryptography

Lecture 12: Public-Key Cryptography.RSA. Mental Poker Protocol.

Public Key Cryptography Overview

- Proposed in Diffie and Hellman (1976) "New Directions in Cryptography"
  - public-key encryption schemes
  - public key distribution systems
    - Diffie-Hellman key agreement protocol
  - digital signature
- Public-key encryption was proposed in 1970 by James Ellis
  - in a classified paper made public in 1997 by the British Governmental Communications Headquarters
- Diffie-Hellman key agreement and concept of digital signature are still due to Diffie & Hellman

Public Key Encryption

- Each party has a PAIR (K, K^{-1}) of keys: K is the **public** key and K^{-1} is the **private** key, such that

$$\mathbf{D}_{K^{-1}}[\mathbf{E}_K[M]] = M$$

- Knowing the public-key and the cipher, it is computationally infeasible to compute the private key
- Public-key crypto systems are thus known to be *asymmetric* crypto systems
- The public-key K may be made publicly available, e.g., in a publicly available directory
- Many can encrypt, only one can decrypt

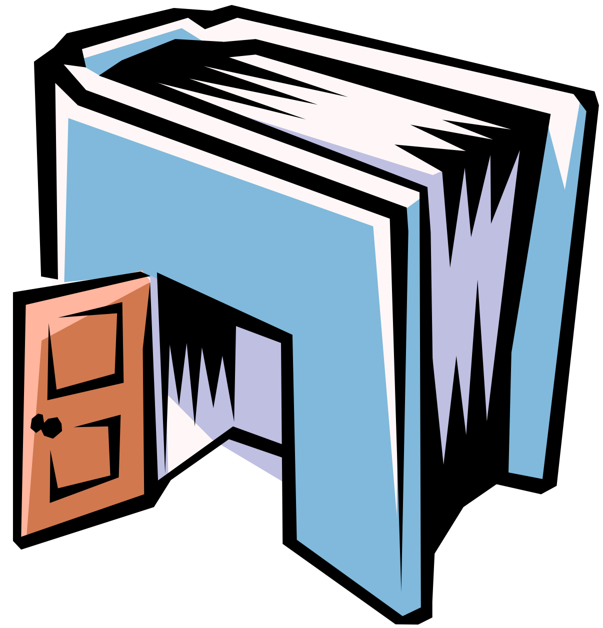
Public-Key Encryption Needs One-way Trapdoor Functions

- Given a public-key crypto system,
  - Alice has public key K
  - $\mathbf{E}_K$ must be a one-way function, knowing y= $\mathbf{E}_K[x]$, it should be difficult to find x
  - However, $\mathbf{E}_K$ must not be one-way from Alice's perspective. The function $\mathbf{E}_K$ must have a trapdoor such that knowledge of the trapdoor enables one to invert it

Trapdoor One-way Functions

Definition:

A function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ is a trapdoor one-way function iff f(x) is a one-way function; however, given some extra information it becomes feasible to compute f^{-1}: given y, find x s.t. $y = f(x)$



RSA Algorithm

- Invented in **1978** by Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman
  - 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
- Security relies on the difficulty of factoring large composite numbers
- Essentially the same algorithm was discovered in 1973 by Clifford Cocks, who works for the British intelligence

Z_{pq}*

- Let p and q be two large primes
- Denote their product n=pq.
- $Z_n^* = Z_{pq}^*$ contains all integers in the range [1,pq-1] that are relatively prime to both p and q
- The size of Z_n^* is

  $$\Phi(pq) = (p-1)(q-1) = n-(p+q)+1$$
- For every $x \in Z_{pq}^*$, $x^{(p-1)(q-1)} \equiv 1 \bmod n$

Exponentiation in Z_{pq}^*

- Motivation: We want to use exponentiation for encryption
- Let e be an integer, $1<e<(p-1)(q-1)$
- When is the function $f(x)=x^e$, a one-to-one correspondence function in Z_{pq}^*?
- If x^e is one-to-one correspondence, then it is a permutation in Z_{pq}^*.

Exponentiation in Z_{pq}^*

- Claim: If e is relatively prime to (p-1)(q-1) then $f(x)=x^e$ is a one-to-one correspondence function in Z_{pq}^*

- Proof by constructing the inverse function of f. As $gcd(e,(p-1)(q-1))=1$, then there exists d and k s.t. $ed=1+k(p-1)(q-1)$
- Let $y=x^e$, then $y^d=(x^e)^d=x^{1+k(p-1)(q-1)}=x \pmod{pq}$, i.e., $g(y)=y^d$ is the inverse of $f(x)=x^e$.

RSA Public Key Crypto System

Key generation:

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

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

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

Compute d, $1 < d < \Phi(n)$ s.t. $ed \equiv 1 \bmod \Phi(n)$

Public key: (e, n)

Private key: d

Note: p and q must remain secret

RSA Description (cont.)

Encryption

Given a message M, $0 < M < n$ $\quad M \in Z_n - \{0\}$

use public key (e, n)

compute $C = M^e \bmod n$ $\quad C \in Z_n - \{0\}$

Decryption

Given a ciphertext C, use private key (d)

Compute $C^d \bmod n = (M^e \bmod n)^d \bmod n = M^{ed} \bmod n = M$

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 $C \equiv M^e \bmod n$

  $C \equiv 15^{37} \pmod{77} = 71$

- $M \equiv C^d \bmod n$

  $M \equiv 71^{13} \pmod{77} = 15$

Why does RSA work?

- Need to show that $(M^e)^d \pmod n = M$, $n = pq$
- We have shown that when $M \in Z_{pq}^*$, i.e., $\gcd(M, n) = 1$, then $M^{ed} \equiv M \pmod n$
- What if $M \in Z_{pq} - \{0\} - Z_{pq}^*$, e.g., $\gcd(M, n) = p$.

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

$M^{ed} \bmod p = (M \bmod p)^{ed} \bmod p = 0$

so $M^{ed} \equiv M \bmod p$

$M^{ed} \bmod q = (M^{k*\Phi(n)} \bmod q)(M \bmod q) = M \bmod q$

so $M^{ed} \equiv M \bmod q$

As p and q are distinct primes, it follows from the CRT that $M^{ed} \equiv M \bmod pq$

RSA Implementation

n, p, q

- The security of RSA depends on how large n is, which is often measured in the number of bits for n. Current recommendation is 1024 bits for n.
- p and q should have the same bit length, so for 1024 bits RSA, p and q should be about 512 bits.
- p-q should not be small

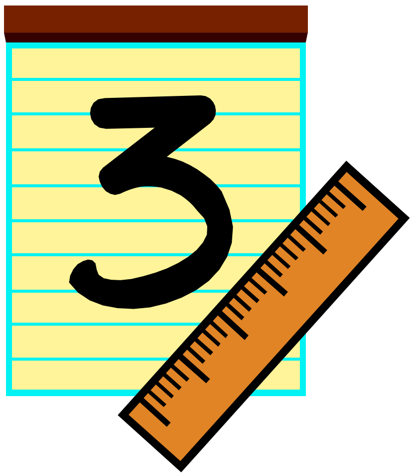
RSA Implementation

- 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

e

- e is usually chosen to be 3 or $2^{16} + 1 = 65537$
- In order to speed up the encryption
  - the smaller the number of 1 bits, the better
  - why?



Square and Multiply Algorithm for Exponentiation

- Computing $(x)^c \bmod n$
  - Example: suppose that c=53=110101
  - $x^{53}=(x^{13})^2 \cdot x=(((x^3)^2)^2 \cdot x)^2)^2 \cdot x =(((x^2 \cdot x)^2)^2 \cdot x)^2)^2 \cdot x \bmod n$

Alg: Square-and-multiply $(x, n, c = c_{k-1}\ c_{k-2} \ldots c_1\ c_0)$

```
z=1
for i ← k-1 downto 0 {
        z ← z^2 mod n
        if c_i = 1 then z ← (z * x) mod n
}
return z
```

RSA Implementation: Decryption

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

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

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

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

  Fermat theorem on the exponents

  $M1 \equiv C^{d \bmod (p-1)} \bmod p$

  $M2 \equiv C^{d \bmod (q-1)} \bmod q$

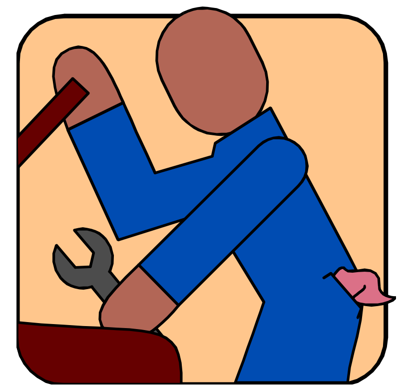
  - then the pair of equations

    $M \equiv M1 \bmod p,$

    $M \equiv M2 \bmod q$

  has a unique solution M.

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

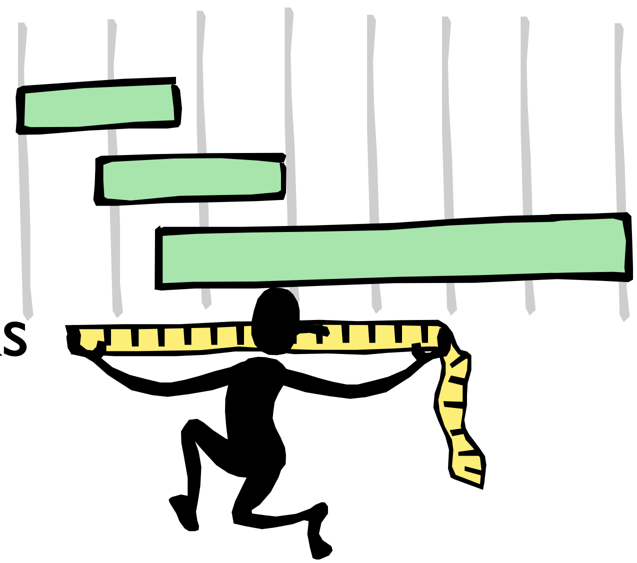


Efficiency of computation modulo n

- Suppose that n is a k-bit number, and $0 \le x, y \le n$
  - computing $(x+y) \bmod n$ takes time $O(k)$
  - computing $(x-y) \bmod n$ takes time $O(k)$
  - computing $(xy) \bmod n$ takes time $O(k^2)$
  - computing $(x^{-1}) \bmod n$ takes time $O(k^3)$
  - computing $(x)^c \bmod n$ takes time $O((\log c)\, k^2)$

RSA on Long Messages

- RSA requires that the message M is at most n-1 where n is the size of the modulus.
- What about longer messages?
  They are broken into blocks.
  Smaller messages are padded.
  CBC is used to prevent attacks regarding the blocks.



- NOTE: In practice RSA is used to encrypt symmetric keys, so the message is not very long.

Pohlig-Hellman Exponentiation Cipher

- A symmetric key exponentiation cipher
  - encryption key (e,p), where p is a prime
  - decryption key (d,p), where $ed \equiv 1 \pmod{(p-1)}$
  - to encrypt M, compute $M^e \bmod p$
  - to decrypt C, compute $C^d \bmod p$

Cristina Nita-Rotaru

Mental Poker Protocol

The Mental Poker Problem

- Alice and Bob want to play poker, deal 5 cards to each of Alice and Bob so that
  - Alice's hand of 5 cards does not overlap with Bob's hand
  - Neither Alice nor Bob can control which cards they each get
  - Neither Alice nor Bob knows the other party's hand
  - Both hands should be random provided one party follows the protocol
- First solution due to Shamir, Rivest, and Adelman in 1980 (SRA protocol)
  - uses commutative encryption schemes

Commutative Encryption

Definition:

An encryption scheme is commutative if
$E_{K1}[E_{K2}[M]] = E_{K2}[E_{K1}[M]]$

Given an encryption scheme that is commutative, then
$D_{K1}[D_{K2}[E_{K1}[E_{K2}[M]] = M$

Most symmetric encryption scheme (such as DES and AES) are not commutative

SRA encryption scheme

- Commutative encryption
- Alice and Bob share $n=pq$ and they both know p and q
- Alice: encryption key e_1
  decryption key d_1
  $e_1 d_1 \equiv 1 \pmod{(p-1)(q-1)}$
- Bob: encryption key e_2
  decryption key d_2
  $e_2 d_2 \equiv 1 \pmod{(p-1)(q-1)}$

The SRA Mental Poker Protocol

Setup: Alice and Bob share M_1, M_2, …, M_{52} denote the 52 cards, n=pq, p, and q. Alice has e_1,d_1 and Bob has e_2,d2

Protocol:

- Alice encrypts M_1, M_2, …, M_{52} using her key, i.e., computes $C_j=M_j^{e1} \bmod n$ for $1 \le j \le 52$, randomly permute them and send the ciphertexts to Bob
- Bob picks 5 cards as Alice's hand and sends them to Alice
- Alice decrypts them to get her hand
- Bob picks 5 other cards as his hand, encrypts them using his key, and sends them to Alice
- Alice decrypts the 5 ciphertexts and sends to Bob
- Bob decrypts what Alice sends and gets his hand
- Both Alice and Bob reveal their key pairs to the other party and verify that the other party was not cheating.