T-79.159
Cryptography and Data Security

Lecture 7: RSA
-Principles of Public Key Cryptosystems
-Setting up RSA
-Prime number generation
-RSA encryption and decryption
-Square and multiply
-Security of RSA

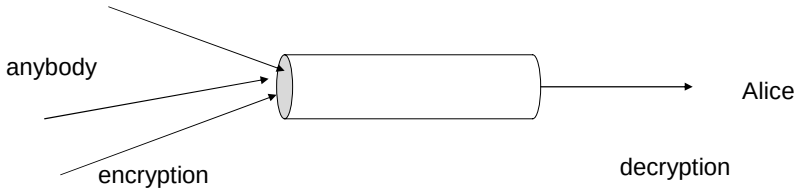
Kaufman et al: Ch 6.3
Stallings: Ch 9.1-2

1

The Principle of Public Key Cryptosystems

Encryption operation is public

Decryption is private



Alice's key for a public key cryptosystem is a pair: (K_{pub}, K_{priv}) where K_{pub} is public and K_{priv} is cannot be used by anybody else than Alice.

2

Setting up the RSA

- Generate two different odd primes p and q
- Compute n = pq and compute $\phi(n) = (p-1)(q-1)$
- Select a public exponent e such that $\gcd(e, \phi(n)) = 1$
- Using Extended Euclidean Algorithm compute the multiplicative inverse of e modulo $\phi(n)$. Denote $d = e^{-1} \bmod \phi(n)$.

Public key: $K_{pub} = (n,e)$

Private key: $K_{priv} = (n,d)$

(or $K_{priv} = (p,q,d)$. This is needed if private computations make use of the CRT)

n is called the RSA modulus; e is the public encryption exponent; d is the private decryption exponent.

3

Generating primes

Prime Number Theorem: Let

$$\pi(n) = \#\{a \mid 1 < a \le n, a \text{ is prime}\}$$

Then

$$\pi(n) \approx \frac{n}{\ln n}$$

Hence the probability that a randomly picked number is a prime is quite large (see excercise)

The primality of a random prime is tested using some primality test: Solovay-Strassen, Miller-Rabin,…, for a deterministic test (not yet practical) see: http://www.cse.iitk.ac.in/news/primality.html

4

Miller-Rabin Primality test

1. Let $n \geq 3$ be odd, consider the even number n -1, and write it as
$$n-1 = 2^k q, \text{ with } q \text{ odd}$$
2. Select a random integer a, $1 < a < n$ -1.
3. If $a^q \bmod n = 1$ then return: n maybe a prime.
4. For j = 0 to k -1 do
5. if $a^{2^j q} \bmod n = n-1$ then return: n may be a prime
6. Return: n is composite

5

RSA encryption and decryption

Let M be a message, $0 \leq M < n$. Then

Encryption of M is $C = M^e \bmod n$

Decryption of C is $M = C^d \bmod n$

This works, because $(M^e)^d \bmod n = M$.

Proof (For M in Z_n^*): By Euler's theorem, $M^{\phi(n)} \equiv 1 (\bmod n)$. On the other hand, $ed \equiv 1 (\bmod \phi(n))$

It follows:

$$(M^e)^d = M^{ed} = M^{1+k\phi(n)} = M \cdot (M^{\phi(n)})^k = M (\bmod n)$$

6

Square and multiply

Fast exponentiation algorithms exist. The simlest one is the Square and Multiply Algorithm:

To compute $a^d \pmod n$

use the binary representation of the exponent d :

$$d = d_0 + d_1 \cdot 2 + d_2 \cdot 2^2 + \cdots + d_{k-1} 2^{k-1}$$

where $d_0, d_1, d_2, \ldots, d_{k-1}$ are bits, i.e. equal to 0 or 1. Now

$$a^d = a^{d_0 + d_1 \cdot 2 + d_2 \cdot 2^2 + \cdots + d_{k-1} 2^{k-1}} = a^{d_0} (a^2)^{d_1} (a^{2^2})^{d_2} \cdots (a^{2^{k-1}})^{d_{k-1}} \pmod n$$

Compute the k powers: $a^0, a^{2^1}, a^{2^2}, a^{2^3}, \cdots, a^{2^{k-1}} \pmod n$

and from of product (modulo n) of those powers $a^{2^i} \pmod n$ for which the corresponding $d_i = 1$.

7

Security of RSA

If factoring of n is easy, then $\phi(n)$ is easy to compute. Given $\phi(n)$ and e, it is easy to compute the private exponent d .

But even if factoring is hard (as it is believed to be) there may be some other ways to break RSA, without factoring the modulus. But no such break is known. All known breaks can be handled by proper selection of parameters, and message formatting.

8