Introduction to Information Security

Dr. Ashok Kumar Das

Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad

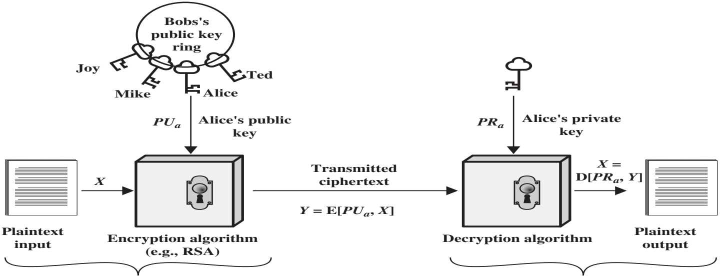
E-mail: *ashok.das@iiit.ac.in*
URL: http://www.iiit.ac.in/people/faculty/ashokkdas
Personal Home Page: http://sites.google.com/view/iitkgpakdas/

Public-Key Encryption

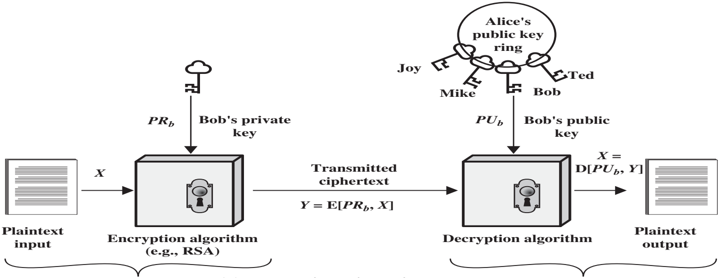
Model of public key encryption

- Consider an encryption scheme consisting of
  - the set of encryption transformations $\{E_e : e \in K\}$
  - the set of corresponding decryption transformations $\{D_d : d \in K\}$, where K is the key space.
- The encryption scheme is said to be public-key or asymmetric-key, if for each associated encryption/decryption key pair (e, d), called public/private key pair, it is computationally "infeasible" to determine private key d from public key e.

Public-Key Cryptography



(a) Encryption with public key



(b) Encryption with private key

The RSA Algorithm

Introduction

- In 1978, Rivest, Shamir and Adleman at MIT, USA discovered a public-key cryptosystem, known as RSA algorithm.
- They received Turing Award (equivalent to Nobel Prize in Computer Science field).
- Their approach is based on elementary number theory concepts.
- The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and $n-1$ for some n. A typical size for n is 1024 bits, or 309 decimal digits.

The RSA Algorithm

Key Generation

Table: Key generation of the RSA algorithm

Select p, q	p and q both prime, $p \neq q$ (p and q are large)
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(e, \phi(n)) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \mod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

The RSA Algorithm

Encryption

Table: Encryption of the RSA algorithm

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

The RSA Algorithm

Decryption

Table: Decryption of the RSA algorithm

Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

The RSA Algorithm

Correctness proof of the RSA algorithm

We have, $C = M^e \pmod{n}$.
So, $M = C^d \pmod{n}$
$= (M^e \pmod{n})^d \pmod{n}$
$= M^{ed} \pmod{n}$
$= M^{1+k\phi(n)} \pmod{n}$, as $d \equiv e^{-1} \mod \phi(n)$,
that is, $ed \equiv 1 \pmod{\phi(n)}$,
that is, $ed = 1 + k\phi(n)$
$= M$.

Computational aspects of the RSA algorithm

The Miller-Robin Primality Test Algorithm

- It is a randomised algorithm.
- It runs in poly-logarithm time.
- It is based on Fermat's Theorem: If n is prime and a is relatively prime to n that is $\gcd(a, n) = 1$, then $a^{n-1} \equiv 1 \pmod{n}$.

Computational aspects of the RSA algorithm

Boolean MillerRobinTest (integer n)

{n to be tested whether prime or not}
Find integers k, q with $k > 0$, q odd, so that $n - 1 = 2^k q$;
Select a random integer a, $1 < a < n - 1$;
if $(a^q \mod n = 1)$ **then**
  return "inconclusive";
end if
for $j = 0 \to k - 1$ **do**
  if $(a^{2^j \cdot q} \mod n = n - 1)$ **then**
    return "inconclusive";
  end if
end for
return "composite";

Computational aspects of the RSA algorithm

Probability of success in MillerRobinTest

- The MillerRobinTest returns inconclusive, but n is not prime, for at most $\frac{n-1}{4}$ integers a with $1 < a < n-1$.
- The probability that MillerRobinTest will return inconclusive (fail to detect n is not prime) is $\frac{(n-1)/4}{n} < 1/4$.
- If MillerRobinTest returns inconclusive t times in succession, then the probability that n is prime is $\geq 1 - (\frac{1}{4})^t$.
- For $t = 10$, the probability of success that n is prime is $\geq 1 - \frac{1}{4^{10}} = 0.9999999$.

Computational aspects of the RSA algorithm

The AKS Algorithm

- Prior to 2000, there was no known method of efficiently proving the primality of very large numbers.
- All of the algorithms, including the most popular MillerRabin Test produced a probabilistic result.
- In 2002, Agrawal, Kayal and Saxena (CSE Dept., IIT Kanpur) developed a relatively simple deterministic algorithm that efficiently (in polynomial time) determines whether a given large number is a prime.
- Due to this pioneer work, Agrawal, Kayal and Saxena received so many prizes and awards internationally and nationally.
- Reference: Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin (2004). "PRIMES is in P". Annals of Mathematics 160 (2): 781–793. doi:10.4007/annals.2004.160.781.

Computational aspects of the RSA algorithm

Fast Exponentiation Algorithm (Repeated-Square-and-Multiply Algorithm)

- For both encryption and decryption of RSA, need to compute modular exponentiation $x^e \mod n$.
- If e is a power of 2, i.e., $e = 2^k$, then can be exponentiated by successive squarings:
  $x^e = ((((x^2)^2)^2)^2 \cdots)^2$.
  For example, $x^8 = ((x^2)^2)^2)$.
- if e is not a power of 2, we take its binary representation. Assume $2^{k-1} \leq e < 2^k$.
- Express $e = (b_{k-1} b_{k-2} \ldots b_1 b_0)_2$
  $= b_{k-1}2^{k-1} + b_{k-2}2^{k-2} + \ldots b_1 2^1 + b_0 2^0$, with $b_{k-1} = 1$
  $= (b_{k-1}2^{k-2} + b_{k-2}2^{k-3} + \ldots + b_1).2 + b_0$
  $= ((b_{k-1}2^{k-3} + b_{k-2}2^{k-4} + \ldots + b_2).2 + b_1).2 + b_0$

Computational aspects of the RSA algorithm

Fast Exponentiation Algorithm (Repeated-Square-and-Multiply Algorithm) (Continued...)

- $\vdots$
  $= ((b_{k-1}2^1 + b_{k-2}).2 + \ldots + b_1).2 + b_0$
  $= ((2 + b_{k-2}).2 + \ldots + b_1).2 + b_0$
- $x^e = (((x^2.x^{b_{k-2}})^2.x^{b_{k-3}})^2 \ldots x^{b_1})^2.x^{b_0}$
- Use the property of modular arithmetic, $(a \times b) \pmod n$
  $= ((a \bmod n) \times (b \bmod n)) \pmod n$,
  $y = x^e \bmod n$
  $= ((((x^2.x^{b_{k-2}} \bmod n)^2.x^{b_{k-3}} \bmod n)^2 \ldots x^{b_1} \bmod n)^2.x^{b_0} \bmod n) \pmod n$.

Computational aspects of the RSA algorithm

Algorithm: Repeated-Squre-and-Multiply (x, e, n)

$\{$To compute $y = x^e \mod n\}$
$y \leftarrow x$;
$k \leftarrow$ BitLength (e)
for $i = k - 2 \rightarrow 0$ **do**
  $y \leftarrow y^2 \pmod{n}$;
  if $b_i = 1$ **then**
    $y \leftarrow y.x \pmod{n}$;
  end if
end for
return y;

Computational aspects of the RSA algorithm

Time complexity of Repeated-Squre-and-Multiply Algorithm

- Let $l = \lfloor log_2 n \rfloor$.
- Computation involves l modular squarings, l modular multiplications and l modular divisions.
- Time complexity is then polynomial in l.

The RSA Algorithm

Security of the RSA algorithm

- **Brute force:** This involves trying all possible private keys.
- **Mathematical attacks:** There are several approaches, all equivalent in effect to factoring the the product of two primes (Integer Factorization Problem (IFP)).
- **Timing attacks:** These depend on the running time of the decryption algorithm.

The RSA Algorithm

The Factoring Problem

We can identify three approaches to attacking RSA mathematically:

- Factor n into its two prime factors. This enables calculation of $\phi(n) = (p-1)(q-1)$, which, in turn, enables determination of $d = e^{-1} \pmod{\phi(n)}$.
- Determine $\phi(n)$ directly, without first determining p and q. Again, this enables determination of $d = e^{-1} \pmod{\phi(n)}$.
- Determine d directly, without first determining $\phi(n)$.

The RSA Algorithm

Problem:

The ciphertext message produced by the RSA algorithm with the public key $(e, n) = (223, 1643)$ is:
1451 0103 1263 0560 0127 0897.
Determine the original plaintext message.
Use the standard encoding procedure:
A = 01, B = 02, . . ., Z = 26,
, = 27, . = 28, ? = 29,
0 = 30, 1 = 31, . . ., 9 = 39, ! = 40.