The RSA Cryptosystem

Debdeep Mukhopadhyay

Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
INDIA -721302

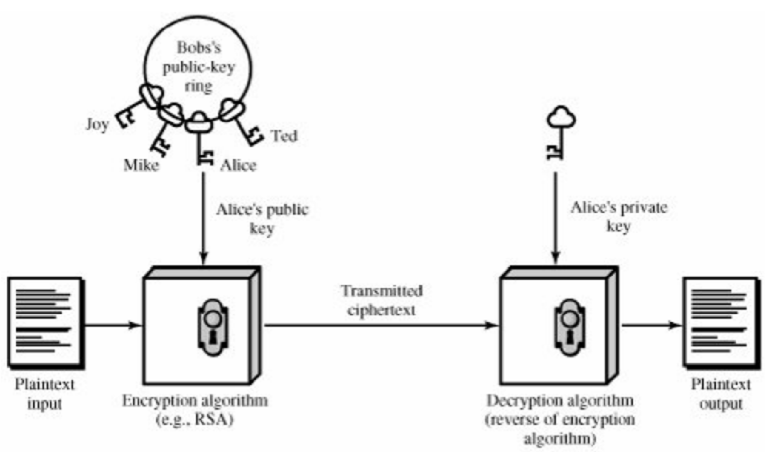
Public Key Cryptography

- **Two keys**
  - **Sender uses recipient's public key to encrypt**
  - **Receiver uses his private key to decrypt**
- **Based on trap door, one way function**
  - **Easy to compute in one direction**
  - **Hard to compute in other direction**
  - **"Trap door" used to create keys**
  - **Example: Given p and q, product N=pq is easy to compute, but given N, it is hard to find p and q**

Public Key Cryptography

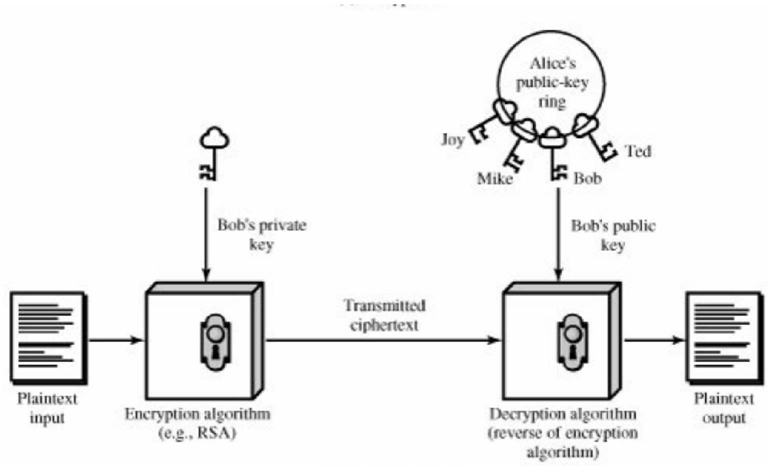
- Encryption
  - **Suppose we encrypt M with Bob's public key**
  - **Only Bob's private key can decrypt to find M**
- Digital Signature
  - **Sign by "encrypting" with private key**
  - **Anyone can verify signature by "decrypting" with public key**
  - **But only private key holder could have signed**
  - **Like a handwritten signature**

Encryption



(a) Encryption

Authentication



(b) Authentication

The RSA

RSA Cryptosystem

Let $n = pq$, where p and q are primes. Let $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$, and define

$$\mathcal{K} = \{(n, p, q, a, b) : ab \equiv 1 \pmod{\phi(n)}\}.$$

For $K = (n, p, q, a, b)$, define

$$e_K(x) = x^b \bmod n$$

and

$$d_K(y) = y^a \bmod n$$

($x, y \in \mathbb{Z}_n$). The values n and b comprise the public key, and the values p, q and a form the private key.

Proof of Correctness

$ab \equiv 1(\text{mod } \phi(\text{n})) \Rightarrow \text{ab}=1+\text{t}\phi(\text{n})$

for some integer t ≥ 1.

Suppose, $x \in Z_n^* \Rightarrow x^{ab} \equiv x^{1+t\phi(n)} \equiv x(x^{\phi(n)})^t \equiv x \ (\text{mod n})$

[follows from Euler's Theorem]

Now, consider x $\in Z_n \setminus Z_n^*$

$So, \gcd(x,n) \neq 1 \Rightarrow (x$ is a multiple of $p)$or$(x$ is a multiple of $q)$

Thus, gcd(x,p)=p or gcd(x,q)=q

If gcd(x,p)=p, then gcd(x,q)=1

[as otherwise x is a multiple of both p and q and still
x is less than n=pq]

Proof of Correctness

Thus, $x^{\phi(q)} \equiv 1(\text{mod } q) \Rightarrow x^{t\phi(q)} \equiv 1(\text{mod } q)$

$\Rightarrow x^{t\phi(q)\phi(p)} \equiv 1(\text{mod } q)$

$\Rightarrow x^{t\phi(n)} \equiv 1(\text{mod } q)$

Thus, $x^{t\phi(n)} = 1 + kq,$

where k is a positive integer

Multiplying both sides by x,

$x^{t\phi(n)+1} = x + kqx$

$\because \gcd(x, p) = p \Rightarrow x = cp,$ for some positive integer c

$x^{\phi(n)+1} = x + kcpq$

$\Rightarrow x^{t\phi(n)+1} \equiv x^{ab} \equiv x(\text{mod n})$

Similarly, we can prove when gcd(x,q)=q

Example

- **Bob chooses p=101 and q=113**
  - **Thus n=11413**
  - **$\Phi(n)=100\text{x}112=11200=2^6 5^2 7$**
  - **b can be used for encryption if and only if it is not a multiple of 2, 5 or 7. Let b=3533**
- **In practice Bob will not factor $\Phi(n)$, but will check whether gcd(b, $\Phi(n)$)=1 using EA and compute b^{-1} at the same time.**

Examples

- **Bob publishes n=11413 and b=3533.**
- **Suppose Alice wants to encrypt x=9726 and send to Bob.**
- **Hence, she computes x^b(mod n) $=9726^{3533}$mod 11413=5761 and sends it to Bob.**
- **Bob computes b^{-1}mod $\Phi(n)$=6597 and decrypts using 5761^{6597} mod 11413=9726**

Efficient Exponentiation

- **Compute x^c efficiently mod n.**
- **Express c as follows:** $c = \sum_{i=0}^{\ell-1} c_i 2^i$

SQUARE-AND-MULTIPLY(x, c, n)

$z \leftarrow 1$
for $i \leftarrow \ell - 1$ **downto** 0
  do $\begin{cases} z \leftarrow z^2 \bmod n \\ \textbf{if } c_i = 1 \\ \quad \textbf{then } z \leftarrow (z \times x) \bmod n \end{cases}$
return (z)

Prime Number Theorem

- **Number of primes that are less than or equal to N is given by:**

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

Hence,…

- If N is a 512 bit number, then there are around $2^{512}/\ln 2^{512} \approx 2^{512}/355$.
- So, a random 512 bit integer will be prime with probability of 1/355.
- Thus, if you choose 355 integers then there is one number which is prime
- If you choose only odd numbers the probability doubles.

Choosing the parameters of RSA

RSA PARAMETER GENERATION

1. Generate two large primes, p and q, such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$
3. Choose a random b $(1 < b < \phi(n))$ such that $\gcd(b, \phi(n)) = 1$
4. $a \leftarrow b^{-1} \bmod \phi(n)$
5. The public key is (n, b) and the private key is (p, q, a).

- n is known, but its factors are not known
- b is also known, so to compute a one needs the value of $\Phi(n)$, for which we need p and q
- It has been conjectured that breaking RSA is polynomially equivalent to factoring n. But there is no proof!
- Typically, value of n is 1024 bit long and the factors are also large of around 512 bits.

Computing $\Phi(n)$

- **For if n and $\Phi(n)$ are known, and n is the product of two primes p, q:**
  - **then n can be factored by solving:**

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

Combining we obtain:

$$p^2-(n-\Phi(n)+1)p+n=0$$

The roots are p and q.

Decryption Exponent

- **If the decryption exponent is known, then n can be factored:**
  - **there is a deterministic algorithm published in Crypto 2004 by Alexander May if:**
    - **a and b are of the same bit size**
    - **ab < n^2**
    - **Run Time: $O(\log^2 n)$**

Importance of this result

- **This result is important from practical point of view:**
  - **if b (the RSA secret key) is leaked, then changing it does not suffice**
  - **one needs to change the modulus n.**

Parity(y) and Half(y)

- **Parity(y) denotes the low order bit of x, that is parity(y)=0, if x is even and parity(y)=1 if x is odd.**

- **Half(y)=0, if $0 \le x < n/2$ and half(y)=1 if $n/2 < x \le n-1$.**

Reductions

- **Existence of a polynomial time algorithm that computes half(y)=>Existence of a polynomial time algorithm for RSA decryption.**
- **RSA Hard => Computing half(y) is hard.**

The Proof Idea

- **Let there be an oracle HALF, which computes half(y).**
  - **if half(y)=0, then x ε [0, n/2)**
  - **Now, $y=x^b$mod n. Compute,**

    $y=2^b y$ mod n

    $=(2x)^b$ mod n
  - **if half(y)=0, then 2x ε [0, n/2)**

    =>x ε [0, n/4) U [n/2, 3n/4)

Continuing in this fashion we obtain distinct boundaries of x. Then the actual x value, can be found out using binary search technique.

Algorithm

Algorithm: Oracle RSA Decryption(n,b,y)
external HALF
k $\leftarrow$ *floor*$(\log_2 n)$
for i=0 to k,
{
$h_i = HALF(n,b,y)$
$y = y \times 2^b \pmod{n}$
}

Binary Search Routine →

```
for i=0 to k
{
    mid=(hi+lo)/2
    if(h_i == 1)
        lo=mid
    else
        hi=mid
}
return floor(hi)
```

Parity ?

- **Computing parity(y) is polynomially equivalent to computing half(y):**
  - **half(y)=parity((y x $e_K(2)$) mod n)**
  - **parity(y)=half((y x $e_k(2^{-1})$) mod n)**

Proof Sketch

- Parity((y x $e_k(2)$) mod n)
  = Parity($e_k(2x)$ mod n)
  = 0, if 2x is even
  1, if 2x is odd

Now, n=2t+1, where t is an integer
and $0 \le x < n/2 \Rightarrow 0 \le x \le t \Rightarrow 0 \le 2x \le 2t = n-1$
All these number are even
If, $n/2 \le x \le n-1 \Rightarrow t+1 \le x \le 2t \Rightarrow 2t+2 \le 2x \le 4t$
or, $n+1 \le 2x \le 2n-2$
Taking, modulo n we have: $1 \le 2x \le n-2$
All these numbers are odd. This proves first eqn.

References

- D. Stinson, Cryptography: Theory and Practice, Chapman & Hall/CRC