CSC 591 Cryptography 10/08/2018

Lecture 9 - The RSA Trapdoor Function.

Lecturer: Alessandra Scafuro *Scribe: PRASHANTHI KANNIAPPAN MURTHY*

Topic Overview

In this lecture we'll see the ground breaking work of Ron Rivest, Adi Shamir, and Len Adleman. Till late 1970s, both sender and receiver shared the same key and the method of key transfer was a problem. Encryption and cryptography in general belonged to the domain of intelligence and military organizations, and only with the advent of public-key techniques did the use of cryptography spread to masses with us being able to use Internet services. The core idea lies with the trapdoor functions, which the receiver uses to decrypt the messages.

NUMBER THEORY BASICS

Groups and Cyclic groups are discussed in the previous lecture on DDH. Here we will continue with Multiplicative groups and their difference with Cyclic groups.

Multiplicative groups: A group defined with respect to multiplication modulo N is called a Multiplicative group. Cyclic groups are also multiplicative groups. They have cyclic nature in addition to being multiplicative.
Group G has the following properties

- Identity

  Since 1 *is always in* Z_N^* , *the set clearly contains an identity element.*

- Inverse

  $$\forall a \in G, \exists x,$$

  such that

  $$a.x = 1$$

- Closure

  $$\forall a, b \in G$$

  $$c = a.b \in G$$

To be in Multiplicative group, the element should be invertible to modulo N.
.Eg,

$$Z_N^* =^{def} \{b \in \{1, ..., N-1\} |\ gcd\ (b, N) = 1\}$$

$$Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$$

Modular Inverse

'a' has a modular inverse with mod n , IFF 'a' and 'n' are co-primes

$$gcd(a, n) = 1$$

Extended GCD Algorithm from Bezout's Equation: Used in calculating the modular inverse

$$d = gcd(a, b)$$

$$d = ax + by$$

Eg, $gcd(7, 15) = 1$ Plugging in values, $1 = 7x + 15y$ where x is the modular inverse of $7 \pmod{15}$
To find the inverse of $a \pmod{n}$, we can compute ext.gcd(a,n) = d,x,y where x is the modular inverse of a

$$|\mathbb{Z}_{15}^*| = 8$$

8 is also, the number of elements co prime with 15

Euler s Totient Function

$\Phi(n) = \{$number of elements that are coprime with n$\}$

Totient funtion is defined as
if n = p, where p is prime $\Phi(n) = p - 1$
n = p.q, where p,q are prime $\Phi(n) = (p - 1)(q - 1)$

Repeated square algorithm To calculate $X^a \pmod{n}$:
Can be done by re using the results of smaller powers for further calculations
Eg,
$2^{16} \pmod 3$, can be calculated as
$y_2 = 2^2 \ mod \ 3$
$y_4 = y_2 * y_2 \ mod \ 3$
$y_8 = y_4 * y_4 \ mod \ 3$
$y_{16} = y_8 * y_8 \ mod \ 3$
The calculation was reduced to just 3 steps

Definition

Let GenRSA be a PPT algorithm that, on input 1^n, outputs a modulus N that is the product of two n-bit primes, along with integers e, d satisfying ed = 1 mod Φ(N).
Such an algorithm can be easily constructed from any algorithm GenModulus outputs a composite modulus N along with its factorization

RSA key generation GenRSA
Input: Security parameter 1^n
Output: N, e, d as described in the text
1) (N, p, q) $\leftarrow$ ***GenModulus(***1^n***)***
2) Φ***(N) := (p-1)(q-1)***
3) choose e > 1 such that gcd(e, Φ ***(N)) = 1***
4) compute d := $[e^{-1}$***mod (N)] 5)return N, e, d***

Assumption

- Finding the e^{th} modular root of Ciphertext C ($\sqrt[e]{c}$) is assumed to be hard.(Stronger assumption than hardness in factoring)
- Finding e^{th}root's hardness is dependent on finding $\Phi(n)$which in turn relies on factoring of two primes
- The RSA problem (which lies on finding the modular root) cannot be more difficult than factoring; hardness of factoring can only potentially be a weaker assumption than hardness of the RSA problem, as depicted below
  - If Factoring of 2 prime is broken then RSA is broken
    But if RSA is broken, then factoring is not broken

Scheme

Gen(1^λ **):**

1. Pick 2 primes p,q and compute n=p*q
2. Calculate Φ(n) = (p-1)(q-1)
3. Compute 2 numbers e,d such that

$$e.d = 1 \ \ mod \ \ \Phi(n)$$

- How to compute this?

  (a) Pick e Co Prime with Φ(n)
  (b) Compute modular inverse of e using Extended gcd(e,Φ(n))

Public Key → {n,e} Private Key→ {d}

Encryption: Enc(n,e,m)

$$c \leftarrow m^e \ mod \ n$$

Decryption: Dec(d,c)

$$m \leftarrow c^d \ mod \ n$$

Security Proof:

Intuition: Adversary can see (n,e), To go from Ciphertext to Message , it can only be done by knowing the value of (d).Finding the value of d, lies in the hardness of finding the e^{th} root.

Is this easy version (textbook version) of RSA, CPA secure ?
No, as it is deterministic (same output, everytime you feed the same message)
Proof:
CPA Adversary A (n,e):

- Training Phase
- Challenge Phase:
  - sends 2 messages m_0 , m_1
  - obtains c from the oracle
- Decision Phase
  - check if c = m_0^e mod n then output 0
  - check if c = m_1^e mod n then output 1

To overcome this, randomness has to be added to RSA
(m $\|$ $(r \ bits)$)^e
This is done in PKCS v1.5 which makes it CPA-secure
Disadvantage of PKCS v1.5, the longer the r , its more secure. But in practise its an expensive operation
OAEP-RSA, overcomes this by hashing the message $\mathtt{H}(m)$ to get randomness, without adding random bits. Hashing will be covered in further classes.

Efficiency of RSA:

RSA Algorithm is efficient when worked out using Repeated Square Algorithm
Its still costlier than just shuffling done in previous encryptions.