Lecture 7: RSA ~ Balanced Power Mod Operation

LEARNING OUTCOME
By the end of the lesson the student will be able to:

a) to understand Pubic Key RSA Algorithm
b) to understand the generation process of public and private keys
c) to relate encryption and decryption process using RSA to PKI.
d) to sign and verify a digital signature

RSA is popular

1. It is the first popular PKI
2. It is written in a simple formula.
3. It follows few thousand years concept of prime numbers.
4. It is being written and taught in crypto textbook.
5. It is part of the early standard to reckon with.

Key Generation Process:

1. Generate primes P and Q of size n=512 bits.
2. Compute the modulus $N = P \cdot Q$
3. Set the public exponent $E = 2^{16}+1$.
4. Compute private exponent $D = E^{-1} \bmod (P-1)\cdot(Q-1)$
5. Set Public key (N, E) and Private key(N, D).

Encryption Process

1. Alice takes a plaintext message M and public key (N, E) of the receiver Bob.
2. Alice computes the ciphertext $C = M^E \pmod N$ and send to Bob

Decryption Process

1. Bob takes the ciphertext C and his Private key (N, D)
2. Bob computes the message $M = C^D \pmod N$.

Next Algorithm we need to learn is power mod operation.
Let the exponent $b = b_0 \cdot 2^0 + b_1 \cdot 2^1 + \ldots b_{n-1} \cdot 2^{n-1}$
For example: $b = 2^{16}+1$ = 10000000000000001 in big endian
Take smaller example $b = 11_{10} = 1101_2$ in big endian
Normally, we write $b = 11_{10} = 1011_2$ in little endian.
Take M = 3. To compute M^b
Note: A current answer is always on the left.

b_i	b	1	1	0	1
Left	0	1	3	3	11
Right	1	2	4	8	16

How to get to 11?
Start from Left = 0 and Right = 1. Then upon seeing the first bit 1,
We add on the left, L=L+R. We double on the right. R=2R

b_i	M^b	1	1	0	1
Left	1	3	27	27	177147
Right	3	9	81	6561	43046721

In this mode, we always square on the right. We only multiply on the left when we see bit 1.

Algorithm 1: Power(M, b, N)
This is a textbook powermod operation
Let $b = b_0 \cdot 2^0 + b_1 \cdot 2^1 + \ldots b_{n-1} \cdot 2^{n-1}$ in big endian

---

```
Left =1. Right = M.
for i = 0; i<n; i++
        if b_i = 1, then
                Left = Left*Right mod N;
        end(*if*)
                Right = Right*Right mod N;
end(*for*)
```

Take smaller example $b = 11_{10} = 1011_2$ in little endian
Take M = 3. To compute M^b, Right = Left +1

b_i	b	1	0	1	1
Left	0	1	2	5	11
Right	1	2	3	6	12

b_i	M^b	1	0	1	1
Left	1	3	9	243	177147
Right	3	9	27	729	531441

Algorithm 2: Power(M, b, N)

Let $b = b_{n-1} \cdot 2^{n-1} + b_1 \cdot 2^1 + \ldots + b_0 \cdot 2^0$ in little endian

$= [b_{n-1} \ldots b_1\ b_0]$

Left =1. Right = M.
for i = 0; i<n; i++;
    if $b_i = 0$, then
        Right = Left*Right mod N;
        Left = Left*Left mod N;
    end(*if*)

    if $b_i = 1$, then Left = Left*Right mod N;
        Right = Right*Right mod N;
    end(*if*)
end(*for*)

Algorithm 2 use little endian structure to avoid side channel attacks.

Given a target K, how to reach K?
Traditionally, we will have a binary sequence of K.
Take $K = 200_{10} = 11001000_2$ which is typically written in little endian.
Let $K = 200 = a_7\ a_6 \ldots a_2\ a_1\ a_0$ in little endian.

In the textbook, 200 = 8 + 64 + 128 in big endian.
$= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 + 1 \cdot 2^6 + 1 \cdot 2^7$
Let $K = 200 = a_0\ a_1\ a_2 \ldots a_6\ a_7$ in big endian.

Algorithm 1: Moving toward a target from Right to Left
Input A = a_{n-1} a_6 … a_2 a_1 a_0

```
Left=1, Right =0.
for i from 0 to n−1.
        Left = 2 · Left
        if a_i = 1 then
                Right = Left + Right
        end*if*
end*for*
```

Table 3. Getting a target from right to left.

i	a_i	Left	Right
-1		0	0
0	0	1	0
1	0	2	0
2	0	4	0
3	1	8	8
4	0	16	8
5	0	32	8
6	1	64	72
7	1	128	200

Algorithm 2: Moving toward a target from Left to Right
Input A = $a_{n-1}\ a_6 \ldots a_2\ a_1\ a_0$ in little endian

---

```
Left=0, Right =0.
for i from n−1 down to 0.

    if ai = 0 then
        Right = Left + Right
        Left = 2·Left
    end*if*

    if ai = 1 then
        Left = Left + Right
        Right = 2·Right
    end*if*

end*for*
```

Table 3. Getting a target from right to left.

i	a_i	Left	Right
		0	1
7	1	1	2
6	1	3	4
5	0	6	7
4	0	12	13
3	1	25	26
2	0	50	51
1	0	100	101
0	0	200	201

Note: 1. A target answer is on the left
2. Right is always Left plus 1.

The objective here is to have a balance computation regardless of a bit sequence a_i.

Algorithm 3: ECC point multiplication
Moving toward a target from Left to Right
Input A = $a_{n-1}\ a_6 \ldots a_2\ a_1\ a_0$

```
Left=P0(x0, y0), Right = P1(x1, y1)

for i from n−1 down to 0,
        if ai = 0 then
                Right = Left + Right   //Add Point
                Left = 2·Left          //Double Point
        end*if*

        if ai = 1 then
                Left = Left + Right    //Add Point
                Right = 2·Right        //Double Point
        end*if*
end*for*
```

Algorithm 3: Power Mod Operation
Moving toward a target from Left to Right
Inputs M, N
Input exponent A = $a_{n-1}\ a_6 \ldots a_2\ a_1\ a_0$
Output: C = M^A mod N

```
L=1, R = M mod N.
for i from n−1 down to 0,
        if ai = 0 then
                R = L · R mod N     //Multiply Mod
                L = L^2 mod N       //Square Mod
        end*if*
        if ai = 1 then
                L = L · R mod N     //Multiply Mod
                R = R^2 mod N       //Square Mod
        end*if*
end*for*
return L.
```

Now, please select your partner to do next Tutorial 7 in pairs.