El cifrado RSA:

Descripción

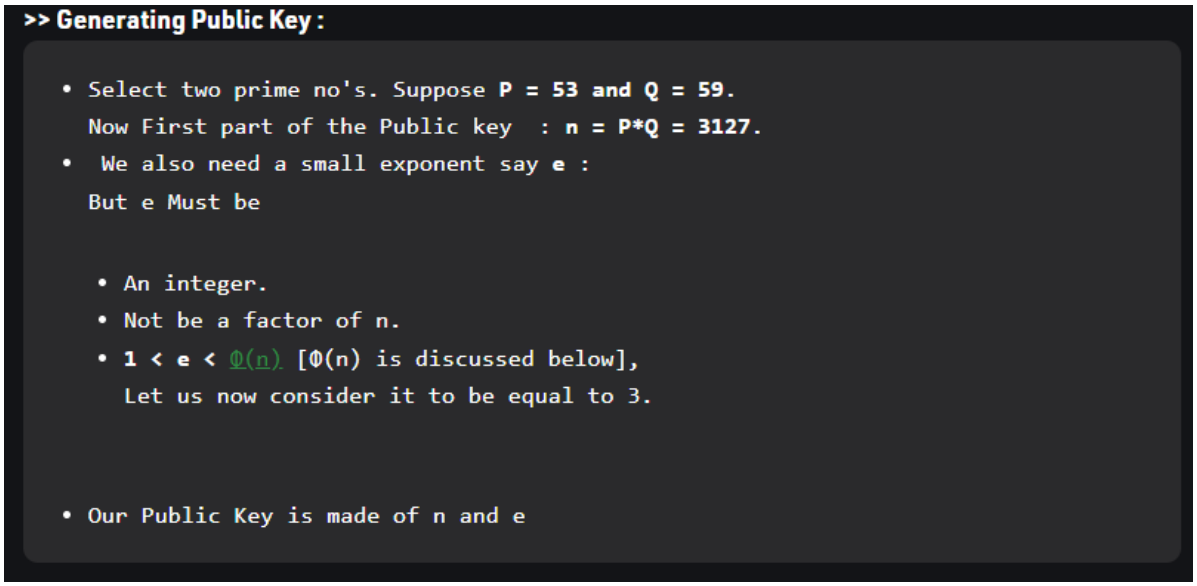
El cifrado RSA es un cifrado assymetrico en sentido en que tiene dos claves separados: una publica y una secreta. Como sugieren los nomres la clave publica esta accessible por todo mundo y la clave secreta esta guardada en privat.

La idea datras del algorithmo RSA esta que esta mut dificil de hacer un factorizacion del un numero entero grande. Esto es important por que una del claves publicas esta el producto del dos primos mayores. Entonces si alguien puede factorizar el clave publica puede calcular facil la clave secreta tambien.

Mencionando estos esta bastante claro que la principal factor del seguridad en RSA esta bazando en el tamano del numeros primos. Si mayoremos los numeros la fuerta del cifrado crece exponencial.

El algoritmo tiene 3 pasos principales:

1**.Generar la clave publica**



```
int KeyGeneration(){
    int p, q;
    int phi_n;

    do{
        do
            p = rand();
        while (p % 2 == 0);

    }
    while (!PrimarityTest(2, p));

    do{
        do
            q = rand();
        while (q % 2 == 0);
    }
    while (!PrimarityTest(2, q));

    n = p * q;
    phi_n = (p - 1) * (q - 1);

    do
        e = rand() % (phi_n - 2) + 2; // 1 < e < phi_n
    while (gcd(e, phi_n) != 1);

    d = inverse(phi_n, e);
}
```

2. **Generar la clave privada/ secreta**

>> Generating Private Key :

- We need to calculate Φ(n) :
  Such that **Φ(n) = (P-1)(Q-1)**
  so, Φ(n) = 3016

- Now calculate Private Key, **d** :
  d = (k*Φ(n) + 1) / e for some integer k
  For k = 2, value of d is 2011.

Junto con el punto 1.

```
int Encryption(int value, FILE *out){
    int cipher;
    cipher = FindT(value, e, n);          2 = 1;
    fprintf(out, "%d ", cipher);
}

int Decryption(int value, FILE *out){
    int decipher;
    decipher = FindT(value, d, n);
    fprintf(out, "%c", decipher);
}
        t1 = t2;
        t2 = t;
    }

    if (r1 == 1){
        inv = t1;
    }

    if (inv < 0){
        inv = inv + a;
    }

    return inv;
}
```

3. **Cifrar/ Descifrar**

- Convert letters to numbers : H = 8 and I = 9

- Thus **Encrypted Data c = 89^e mod n.**
  Thus our Encrypted Data comes out to be 1394

  Now we will decrypt **1394** :

- **Decrypted Data = c^d mod n.**
  Thus our Encrypted Data comes out to be 89
  8 = H and I = 9 i.e. "HI".

```c
int Encryption(int value, FILE *out){
    int cipher;
    cipher = FindT(value, e, n);
    fprintf(out, "%d ", cipher);
}

int Decryption(int value, FILE *out){
    int decipher;
    decipher = FindT(value, d, n);
    fprintf(out, "%c", decipher);
}
```

```c
int FastExponention(int bit, int n, int *y, int *a){
    if (bit == 1){
        *y = (*y * (*a)) % n;
    }

    *a = (*a) * (*a) % n;
}

int FindT(int a, int m, int n)
{
    int r;
    int y = 1;

    while (m > 0)
    {
        r = m % 2;
        FastExponention(r, n, &y, &a);
        m = m / 2;
    }
    return y;
}
```

Principales ataques

Desde cuando temenos el cifrado RSA muchas gentes han intentado encontrar una metoda de ataquar. Son 4 categorias principals:

1. **Ataques elementarios bazados en usa incorecta del Sistema**
    a. **Factorizacion:**
        - Si alguien resuelva la factorizacion del clave publica, el ataquador puede calcular facil “phi(N)” y “d” tambien.
        - Pero si RSA esta usando manera apropiada esto tipo de ataque no hace un riesgo grande.
        - El algoritmo mas rapido ahora esta General Number Field Sieve
        - N(o) = $\left((c + o(1))n^{1/3}\, log^{2/3}n\right)$
    b. **Modulo comun**
        - En esto caso la ataque esta interna. Si un usuario tiene las claves (N, ei) y (N, di) puede factorizar mas facil N hacendo usa del su exponent.
        - Esto observacion del Simmons nos muestra que un modulo esta mas seguro si esta usando del solo una entitate.
    c. **Blinding**
        - En esto caso un usuario tiene de ciphrar un mensaje $M \epsilon Z^{*}_{N}$ que lo sabe el ataquador. Si hace esto el usuario, el ataquador puede obtener

la clave secreta del usuario:

$$S^e = (S')^e / Y^e = (M')^{ed} / Y^e \equiv M' / Y^e = M \pmod N$$

2. **Exponent menor privada**
   - Esto es el mas serio ataque por el sistema RSA. Si tenemes un “d” menor esto puede ropar el seguridad del Sistema totalmente.
   - Let $N = pq$ with $q < p < 2q$. Let $d < 1/3\ N^{1/4}$

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \frac{1}{d\varphi(N)}$$

   - Since $\varphi(N) = N-p-q+1$ and $p+q-1 < 3\sqrt{N}$ an attacker can use N to approximate $\varphi(N)$.
   - Para evitar esto en caso en que N tiene 1024 bits „d” necesita la menor 256 bits.
3. **Exponent menor publica**
   - Para reducir el tiempo del cifrar y descifrar una exponenta publica menor esta usando. La recomendacion por „e” esta $e = 2^{16} + 1$. En esto caso necesita solo 17 multiplicaciones para verificar la signatura en lugar del 1000+ si e esta aleatorio.
   - Pero en esto caso las ataques aplicados cuando „e” esta menor son bastante complejos y no ropen la sistema como hacen las ataques con exponent menor privada.
   - Teoremas para ataquar en esto manera son:
     1. Coppersmith teorem
     2. Hastad’s broadcast attack
     3. Franklin-Reiter Realted Message Attack
     4. Partial Key Exposure attack
4. **Ataque del implementacion**
   - Son totalmente dependiente del implementacion del sistema.
   - En casos en que RSA esta implementado correctamente son dos metodas principals del ataques del implementacion:
     1. Timing attacks
        - Let $d = d_n\ d_{n-1} \ldots d_0$ (binary of d)
        - Set $z = M$ and $C = 1$. For $i=0 \ldots n$ do:
          - (1) if $d_i = 1$ set $C = C \cdot z \bmod N$
          - (2) $z = z * z \bmod N$
        - C at the end has the value $M^d \bmod N$
     2. Random Faults

$$C_p = M^{dp} \bmod p \qquad \text{and} \qquad C_q = M^{dq} \bmod q$$

Where $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$.

then $C = T_1 C_p + T_2 C_q \pmod N$

where

$$T_1 = \{ \begin{matrix} 1 \bmod p \\ 0 \bmod q \end{matrix} \} \qquad \text{and} \qquad T_2 = \{ \begin{matrix} 0 \bmod p \\ 1 \bmod q \end{matrix} \}$$

Referencias:

https://www.utc.edu/sites/default/files/2021-04/course-paper-5600-rsa.pdf

https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

https://www.educba.com/rsa-algorithm/