

# IIC3253

Teoría de números y RSA

# RSA: claves públicas y privadas

Las claves pública y privada de un usuario  $A$  son construidas de la siguiente forma:

1. Genere números primos distintos  $P$  y  $Q$ . Defina

$$N := P \cdot Q$$

2. Defina  $\phi(N) := (P - 1) \cdot (Q - 1)$

3. Genere un número  $d$  tal que  $MCD(d, \phi(N)) = 1$

4. Construya un número  $e$  tal que

$$e \cdot d \equiv 1 \pmod{\phi(N)}$$

5. Defina  $S_A = (d, N)$  y  $P_A = (e, N)$

# RSA: funciones de cifrado y descifrado

Considere un usuario  $A$  con clave pública  $P_A = (e, N)$  y clave secreta  $S_A = (d, N)$


Se tiene que:

$$Enc_{P_A}(m) = m^e \bmod N$$

$$Dec_{S_A}(c) = c^d \bmod N$$

# ¿Por qué funciona RSA?

¿Cuáles son los espacios de mensajes, llaves y textos cifrados? 

¿Es cierto que  $Dec_{S_A}(Enc_{P_A}(m)) = m$ ? 

¿Qué algoritmos de tiempo polinomial se debe tener para que se pueda ejecutar el protocolo?

¿De qué depende la seguridad de RSA? ¿Qué problemas no pueden ser resueltos en tiempo polinomial?

# ¿Qué algoritmos eficientes necesitamos para ejecutar RSA?

Generación aleatoria de números primos

Construcción de claves:

1. Genere números primos distintos  $P$  y  $Q$ . Defina  
 $N := P \cdot Q$

2. Defina  $\phi(N) := (P - 1) \cdot (Q - 1)$

3. Genere un número  $d$  tal que  $MCD(d, \phi(N)) = 1$

4. Construya un número  $e$  tal que

$$e \cdot d \equiv 1 \pmod{\phi(N)}$$

5. Defina  $S_A = (d, N)$  y  $P_A = (e, N)$

# ¿Qué algoritmos eficientes necesitamos para ejecutar RSA?

Operaciones básicas (multiplicación, suma, resto, ...)

Construcción de claves:

1. Genere números primos distintos  $P$  y  $Q$ . Defina

$$N := P \cdot Q$$

2. Defina  $\phi(N) := (P - 1) \cdot (Q - 1)$

3. Genere un número  $d$  tal que  $MCD(d, \phi(N)) = 1$

4. Construya un número  $e$  tal que

$$e \cdot d \equiv 1 \pmod{\phi(N)}$$

5. Defina  $S_A = (d, N)$  y  $P_A = (e, N)$

# ¿Qué algoritmos eficientes necesitamos para ejecutar RSA?

Máximo común divisor

Construcción de claves:

1. Genere números primos distintos  $P$  y  $Q$ . Defina  
 $N := P \cdot Q$

2. Defina  $\phi(N) := (P - 1) \cdot (Q - 1)$

3. Genere un número  $d$  tal que  $MCD(d, \phi(N)) = 1$

4. Construya un número  $e$  tal que

$$e \cdot d \equiv 1 \pmod{\phi(N)}$$

5. Defina  $S_A = (d, N)$  y  $P_A = (e, N)$

# ¿Qué algoritmos eficientes necesitamos para ejecutar RSA?

Generación de un primo relativo

Construcción de claves:

1. Genere números primos distintos  $P$  y  $Q$ . Defina  
 $N := P \cdot Q$

2. Defina  $\phi(N) := (P - 1) \cdot (Q - 1)$

3. Genere un número  $d$  tal que  $MCD(d, \phi(N)) = 1$

4. Construya un número  $e$  tal que

$$e \cdot d \equiv 1 \pmod{\phi(N)}$$

5. Defina  $S_A = (d, N)$  y  $P_A = (e, N)$



# ¿Qué algoritmos eficientes necesitamos para ejecutar RSA?

Algoritmo extendido de Euclides

Construcción de claves:

1. Genere números primos distintos  $P$  y  $Q$ . Defina  $N := P \cdot Q$
2. Defina  $\phi(N) := (P - 1) \cdot (Q - 1)$
3. Genere un número  $d$  tal que  $MCD(d, \phi(N)) = 1$
4. Construya un número  $e$  tal que
$$e \cdot d \equiv 1 \pmod{\phi(N)}$$
5. Defina  $S_A = (d, N)$  y  $P_A = (e, N)$

# ¿Qué algoritmos eficientes necesitamos para ejecutar RSA?

Funciones de cifrado y descifrado:

$$Enc_{P_A}(m) = m^e \bmod N$$

$$Dec_{S_A}(c) = c^d \bmod N$$



Exponenciación rápida

# Algoritmos eficientes para ejecutar RSA

Tenemos que resolver dos problemas: cómo generar un número primo al azar, y cómo generar un primo relativo al azar

# ¿Cuál es la densidad de los números primos?

Sea  $\pi(n)$  la cantidad de números primos menores o iguales a  $n$

- Por ejemplo,  $\pi(10) = 4$  y  $\pi(19) = 8$

**Teorema:**  $\lim_{n \rightarrow \infty} \frac{\pi(n)}{\frac{n}{\ln(n)}} = 1$

Por lo tanto:  $\pi(n) \approx \frac{n}{\ln(n)}$

# ¿Cuál es la probabilidad de obtener un número primo al azar?

$$\Pr_{x \sim \mathbb{U}(1,n)}(x \text{ sea primo}) = \frac{\text{casos favorables}}{\text{casos totales}}$$

# ¿Cuál es la probabilidad de obtener un número primo al azar?

$$\Pr_{x \sim \mathbb{U}(1,n)}(x \text{ sea primo}) = \frac{\text{casos favorables}}{n}$$

¿Cuál es la probabilidad de  
obtener un número primo  
al azar?

$$\Pr_{x \sim \mathbb{U}(1,n)}(x \text{ sea primo}) \approx \frac{\frac{n}{\ln(n)}}{n}$$

# ¿Cuál es la probabilidad de obtener un número primo al azar?

$$\Pr_{x \sim \mathbb{U}(1,n)}(x \text{ sea primo}) \approx \frac{1}{\ln(n)}$$

¿Es esta una probabilidad alta? ¿Podemos obtener un número primo de 400 dígitos en poco tiempo?



# ¿Cuál es la probabilidad de obtener un número primo al azar?

La probabilidad de obtener un número primo de 400 dígitos:

$$\Pr_{x \sim \mathbb{U}(10^{399}, 10^{400}-1)} \approx \frac{\frac{10^{400}-1}{\ln(10^{400}-1)} - \frac{10^{399}-1}{\ln(10^{399}-1)}}{10^{400} - 10^{399}}$$

Esta probabilidad es aproximadamente 0.001085

- Por lo tanto, necesitamos aproximadamente 922 intentos para obtener un número primo



**Conclusión: para generar números primos al azar nos basta con encontrar un algoritmo eficiente para verificar si un número es primo**

# Antes de estudiar un algoritmo de primalidad

Vamos a ver cómo se puede generar un primo relativo al azar

Y vamos a responder una de las preguntas pendientes:  
¿de qué depende la seguridad de RSA?

- ¿Qué problemas no pueden ser resueltos en tiempo polinomial para que este protocolo sea seguro?

# Generando un primo relativo al azar

Queremos generar un primo relativo  $a$  de un número dado  $n$

Sea  $\phi(n)$  la cardinalidad del conjunto

$$\{a \in \{0, \dots, n-1\} \mid \text{MCD}(a, n) = 1\}$$

# Generando un primo relativo al azar

Por ejemplo, si  $N = P \cdot Q$  con  $P$  y  $Q$  primos distintos, entonces  $\phi(N) = (P - 1) \cdot (Q - 1)$

- Por eso usamos la notación  $\phi(N)$  en RSA

$\phi(n)$  es llamada la función  $\phi$  de Euler

# Generando un primo relativo al azar

¿Cuán grande es el valor de  $\phi(n)$ ?

**Teorema:**  $\phi(n)$  es  $\Omega\left(\frac{n}{\log(\log(n))}\right)$

Podemos entonces generar un primo relativo a  $n$  usando el mismo argumento que para los primos

- Generamos números al azar  $a \in \{0, \dots, n-1\}$  y verificamos si  $MCD(a, n) = 1$

# ¿De qué depende la seguridad de RSA?

Obviamente la respuesta depende de la definición de seguridad

Pero un requerimiento básico es que no se pueda descubrir la clave privada a partir de la clave pública

Para esto es necesario que **no** exista un algoritmo eficiente para encontrar un divisor de un número

# RSA: tratando de factorizar

$$N = P \cdot Q$$

$N = P \cdot Q$  en RSA está diseñado para que sea difícil encontrar un divisor

$N$  está diseñado para que el siguiente tipo de ataques a RSA no puede funcionar:

- Genere números  $a \in \{2, \dots, N - 1\}$  hasta que se cumpla la condición  $MCD(a, N) > 1$  y, por lo tanto,  $MCD(a, N)$  sea un divisor de  $N$  mayor a 1 y menor que  $N$



# RSA: tratando de factorizar

$$N = P \cdot Q$$

$$\begin{aligned} \Pr_{x \sim \mathbb{U}(2, N-1)} (MCD(x, N) > 1) &= \frac{N - \phi(N) - 1}{N - 2} \\ &= \frac{P + Q - 2}{N - 2} \end{aligned}$$

# RSA: tratando de factorizar

$$N = P \cdot Q$$

$$\Pr_{x \sim \mathbb{U}(2, N-1)} (MCD(x, N) > 1) = \frac{N - \phi(N) - 1}{N - 2}$$

$$= \frac{P + Q - 2}{N - 2}$$

Este evento no va a suceder

$$\leq \frac{2 \cdot 10^{400}}{10^{798} - 2} \approx \frac{1}{10^{398}}$$

Si  $P$  y  $Q$  tienen 400 dígitos, entonces  $P < 10^{400}$ ,  $Q < 10^{400}$  y  $N \geq 10^{798}$

# La última pregunta a responder

¿Cómo se puede verificar de manera eficiente si un número es primo?

Vamos a estudiar las ideas detrás de un test aleatorizado de tiempo polinomial para verificar si un número es primo

- Hasta el día de hoy, todos los test de primalidad realmente eficientes son aleatorizados