Espacios de mensajes, llaves y textos cifrados

Suponiendo que se genera el número N:

- Espacios de mensajes y textos cifrados: $\{0, \ldots, N-1\}$
- Espacios de llaves: $d, e \in \mathbb{N}$
  - Aunque en general se asume que $d, e \in \{0, \ldots, \phi(N)-1\}$

$$Dec(S_A, Enc(P_A, m)) = m$$

Para demostrar esto necesitamos un resultado fundamental

Pequeño teorema de Fermat: Si p es un número primo, entonces para cada $a \in \{1, ..., p-1\}$, se tiene que $a^{p-1} \bmod p = 1$

Un ejemplo del pequeño teorema de Fermat

Para $p = 7$:

$$1^6 \bmod 7 = 1 \bmod 7 = 1$$
$$2^6 \bmod 7 = 64 \bmod 7 = 1$$
$$3^6 \bmod 7 = 729 \bmod 7 = 1$$
$$4^6 \bmod 7 = 4096 \bmod 7 = 1$$
$$5^6 \bmod 7 = 15625 \bmod 7 = 1$$
$$6^6 \bmod 7 = 46656 \bmod 7 = 1$$

La correctitud de RSA

Sean $P_A = (e, N)$ y $S_A = (d, N)$ generados según el protocolo de RSA

Teorema: para cada $m \in \{0, ..., N-1\}$, se tiene que $Dec(S_A, Enc(P_A, m)) = m$

Demostración de la correctitud de RSA

Tenemos que:

- $N = P \cdot Q$, donde P y Q son números primos
- $e \cdot d = \alpha \cdot \phi(N) + 1$, dado que $e \cdot d \equiv 1 \mod \phi(N)$

Sea $m \in \{0, ..., N-1\}$

Demostración de la correctitud de RSA

Tenemos que:

$$\begin{aligned} Dec(S_A, Enc(P_A, m)) &= (m^e \bmod N)^d \bmod N \\ &= m^{e \cdot d} \bmod N \end{aligned}$$

Demostración de la correctitud de RSA

Entonces tenemos que demostrar que $m^{e \cdot d} \bmod N = m$.
Dado que $m \in \{0, \ldots, N-1\}$, esto es equivalente a:

$$m^{e \cdot d} \equiv m \mod N$$

Vamos a demostrar esto suponiendo primero que $MCD(m, N) = 1$

Demostración de la correctitud de RSA

Como $MCD(m,N)=1$, se tiene que $MCD(m,P)=1$

Por el pequeño teorema de Fermat:

$$m^{P-1} \equiv 1 \mod P$$

Por lo tanto:

$$\begin{aligned}
(m^{P-1})^{Q-1} &\equiv 1 \mod P \\
m^{\phi(N)} &\equiv 1 \mod P \\
m^{\alpha\cdot\phi(N)} &\equiv 1 \mod P \\
m^{\alpha\cdot\phi(N)+1} &\equiv m \mod P
\end{aligned}$$

Demostración de la correctitud de RSA

Por lo tanto:

$$m^{e \cdot d} \equiv m \mod P$$

De la misma forma se concluye que:

$$m^{e \cdot d} \equiv m \mod Q$$

Tenemos entonces que:

$$m^{e \cdot d} - m = \beta \cdot P$$

$$m^{e \cdot d} - m = \gamma \cdot Q$$

Demostración de la correctitud de RSA

Como $\beta \cdot P = \gamma \cdot Q$, se tiene que P divide a $\gamma \cdot Q$

- Como P es primo, se tiene que P divide a γ o P divide a Q

Concluimos que P divide a γ dado que P y Q son números primos distintos

Por lo tanto: $\gamma = \delta \cdot P$

Demostración de la correctitud de RSA

Dado que $\gamma = \delta \cdot P$ y $m^{e \cdot d} - m = \gamma \cdot Q$, concluimos que $m^{e \cdot d} - m = \delta \cdot P \cdot Q$

Tenemos que $m^{e \cdot d} - m = \delta \cdot N$, y finalmente concluimos que:

$$m^{e \cdot d} \equiv m \mod N$$

Demostración de la correctitud de RSA

¿Qué hacemos con el caso $MCD(m, N) > 1$?

Si $m = 0$, entonces concluimos trivialmente que $m^{e \cdot d} \equiv m \mod N$

Consideramos entonces dos casos:

- P divide a m pero Q no divide a m
- Q divide a m pero P no divide a m

Demostración de la correctitud de RSA

Si P divide a m y Q no divide a m, entonces:

- Concluimos que $m^{e \cdot d} \equiv m \mod P$ ya que $m \equiv 0 \mod P$
- Concluimos que $m^{e \cdot d} \equiv m \mod Q$ como en la demostración inicial, usando el hecho de $MCD(m, Q) = 1$

Concluimos que $m^{e \cdot d} \equiv m \mod N$ como en la demostración inicial

Demostración de la correctitud de RSA

Si Q divide a m y P no divide a m, concluimos que $m^{e \cdot d} \equiv m \mod N$ como en el caso anterior

Esto concluye la demostración del teorema 🙂

¿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
$$e \cdot d \equiv 1 \mod \phi(N)$$
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
$$e \cdot d \equiv 1 \mod \phi(N)$$
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
$$e \cdot d \equiv 1 \mod \phi(N)$$
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
$$e \cdot d \equiv 1 \mod \phi(N)$$
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
$$e \cdot d \equiv 1 \mod \phi(N)$$
5. Defina $S_A = (d, N)$ y $P_A = (e, N)$

¿Qué algoritmos eficientes necesitamos para ejecutar RSA?

Funciones de cifrado y descifrado:

$$\begin{aligned} Enc(P_A, m) &= m^e \bmod N \\ Dec(S_A, c) &= c^d \bmod N \end{aligned}$$

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: $\lim_{n\to\infty} \frac{\pi(n)}{\frac{n}{\ln(n)}} = 1$

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

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

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

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

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

¿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?

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

¿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

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

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 ϕ 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 $a \in \{0, \ldots, n-1\}$ y verificamos si $MCD(a, n) = 1$

¿De qué depende la seguridad de RSA?

Obviamente la repuesta 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 $a \in \{2, \ldots, N-1\}$ 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

$$N = P \cdot Q$$

$$\begin{aligned}
\Pr_{x \sim \mathbb{U}(2, N-1)} (MCD(x, N) = 1) &= \frac{N - \phi(N) - 1}{N-2} \\
&= \frac{P + Q - 2}{N - 2} \\
&\leq \frac{2 \cdot 10^{400}}{10^{798} - 2} \approx \frac{1}{10^{398}}
\end{aligned}$$

Este evento no va a suceder

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