

Técnicas de aceleración de RSA

Utilizar exponentiación (incluso usando el método de multiplicación y elevar al cuadrado) es muy costoso computacionalmente. Para esto se desarrollaron técnicas para acelerar este proceso.

Cifrado rápido con llave pública chica

El "truco" aquí es utilizar como llave pública un número "chico" con el que sea necesario hacer pocos cálculos. En particular, si e es la llave pública, se mencionan los valores

	valor base 2	Operaciones
3	11_2	2
17	10001_2	5
$2^{16} + 1$	1000000000000001_2	17

Notemos que estos valores son más fáciles de calcular, pues tienen distancia de Hamming chima.

Ahora, nos podemos preguntar si elegir estos valores tiene el efecto de agilizar el cifrado al costo de sacrificar seguridad. El primero se obtiene, pero el segundo no. En este tipo de protocolos, la seguridad recae en realidad en la llave privada.

Así, el cifrado puede hacerse increíblemente rápido (en comparación a otros protocolos de llave pública).

Decifrado rápido con CRT

Para la llave privada no se puede hacer el mismo truco, pues entonces un atacante podría aplicar fuerza bruta para tener la llave.

De hecho, se recomienda que la longitud en bits de la llave privada sea al menos $0.3t$, donde t es la longitud en bits del módulo n que se utiliza en RSA.

Por lo tanto, en la práctica se usa una llave pública corta y una llave privada larga. Para hacer esto práctico se usa el Teorema Chino de Residuos

(CRT).

El objetivo es: teniendo el texto cifrado y , calcular $y^d \bmod n$, con d la llave privada y n el módulo.

→ Recorremos $n = p \cdot q$

→ Reajustamos $y_p := y \bmod p$

$$y_q := y \bmod q$$

→ Cambiamos exponentes $d_p := d \bmod (p-1)$

$$d_q := d \bmod (q-1)$$

→ Deciframos parcialmente $x_p := y_p^{d_p} \bmod p$

$$x_q := y_q^{d_q} \bmod q$$

Observación: $d_p \leq p$ & $d_q \leq q$

→ Calculamos coeficientes $c_p := q^{-1} \bmod p$

$$c_q := p^{-1} \bmod q$$

→ Decifrado completo

$$x = [q c_p] x_p + [p c_q] x_q \bmod n$$

Ejemplo: $p=11, q=13 \rightsquigarrow n=143; e=7 \rightsquigarrow d=e^{-1}=7^{-1} \bmod 120$

$$\text{Sea } y=15 \rightsquigarrow y_p=15 \bmod 11 \Rightarrow y_p=4$$

$$y_q=15 \bmod 13 \Rightarrow y_q=2$$

103

$$\text{Luego } d_p = 103 \bmod 10 = 3 \bmod 10$$

$$d_q = 103 \bmod 12 = 7 \bmod 12$$

$$\Rightarrow x_p = 4^3 \bmod 11 = 64 \bmod 11 = 9 \bmod 11$$

$$x_q = 2^7 \bmod 13 = 128 \bmod 13 = 11 \bmod 13$$

$$\rightsquigarrow c_p = 13^{-1} \bmod 11 \equiv 2^{-1} \bmod 11 = 6 \bmod 11$$

$$c_q = 11^{-1} \bmod 13 \equiv 6 \bmod 13$$

$$\Rightarrow x = [13 \cdot 6] \cdot 9 + [11 \cdot 6] \cdot 11 \bmod 143$$

$$= 702 + 726 \bmod 143$$

$$\equiv 141 \bmod 143.$$

Complejidad de RSA con CRT

Analizando las operaciones en RSA con CRT, la complejidad de RSA con CRT es "tragada" por la complejidad de $x_p = y_p^{d_p} \bmod p$ & $x_q = y_q^{d_q} \bmod q$

Si n tiene longitud en bits de $t+1$, ent

$$- l(p), l(q) \sim t/2$$

$$- l(x_p), l(x_q), l(d_p), l(d_q) \sim t/2$$

(Como el proceso de multiplicación y elevar al cuadrado)

usa aproximadamente $\frac{3}{2}t$ operaciones, para números de longitud t

$$\Rightarrow \# \text{Mult.} + \# \text{Squared} \sim 2 \cdot \left[\frac{3}{2} \cdot \frac{t}{2} \right] = \frac{3}{2}t$$

Con esto podríamos pensar que con o sin CRT hay la misma complejidad. Sin embargo, la multiplicación y elevar al cuadrado estarán usando factores de longitud $\sim t/2$ y su complejidad es cuadrática con respecto a t

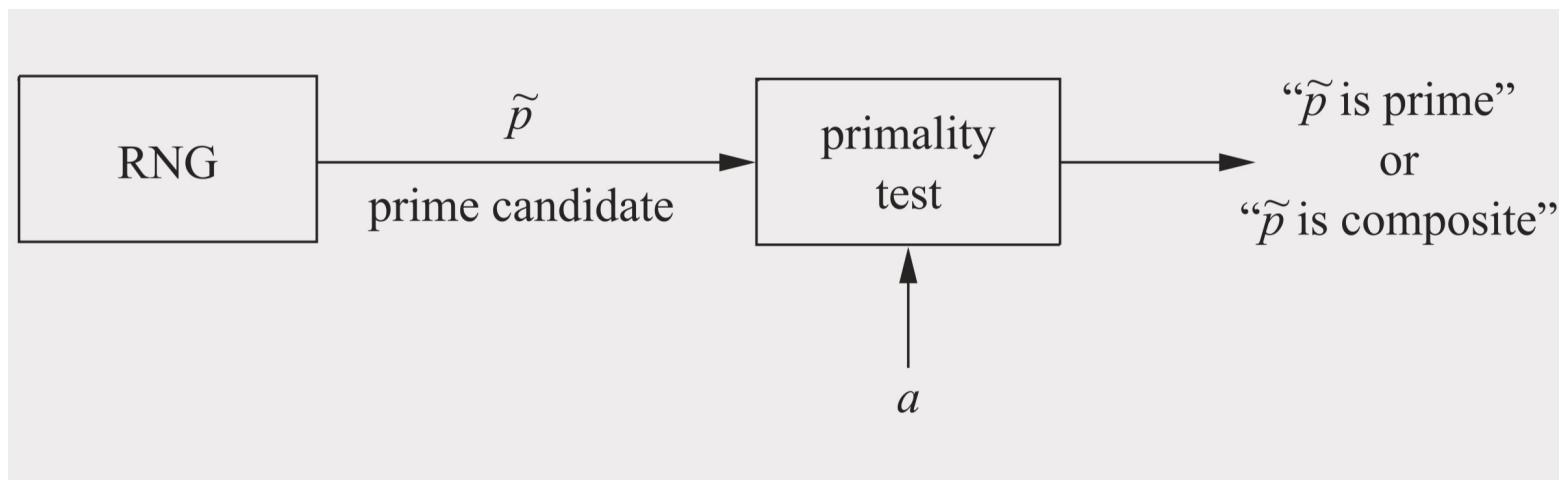
\Rightarrow Con CRT el RSA es al menos 4 veces más rápido

Encontrando primos grandes

Como hemos visto, es de suma importancia el poder encontrar primos p y q de la longitud apropiada. Por ejemplo, si queremos que n sea de longitud 2048, queremos encontrar primos p y q de longitudes (aproximadas) de 1024.

¿Qué tan grande es esto? Si $l(p)=1024$, ent. $p \geq 2^{1024}$

Lo que hacemos entonces es crear números aleatorios y revisarlos para ver si son primos.



El generador de números aleatorios no debe ser predecible.

Detalle: Los tests de primalidad son probabilísticos.

Esto nos lleva a dos preguntas:

- 1) ¿Cuántos números hay que probar antes de llegar a un primo?
- 2) ¿Qué tan rápidos son los tests?

¿Qué tan comunes son los primos?

Dado \tilde{p} un número aleatorio impar, se tiene que:

$$P(\tilde{p} \text{ es primo}) \approx \frac{2}{\ln(\tilde{p})}$$

\Rightarrow Si $\ell(\tilde{p}) \approx 1024$, ent:

$$\begin{aligned}
 P(\tilde{p} \text{ es primo}) &\approx \frac{2}{\ln(\tilde{p})} \\
 &\approx \frac{2}{\ln(2^{1024})} \\
 &= \frac{2}{1024 \ln(2)} \\
 &\approx \frac{1}{354}
 \end{aligned}$$

Con 354 tests se esperaría al menos un primo.

Tests de primalidad

Vamos a ver en esencia los tests (Fermat & Miller-Rabin). Dado que son tests probabilísticos, existe la posibilidad de un falso positivo. La forma de "solucionarlo" es: si sale positivo bajo cierto parámetro (el a de la figura de arriba), volver a pasarlo por el test con un parámetro diferente. Esto disminuye la probabilidad de un falso positivo. Nos gustaría llegar a menos de 2^{-80} .

Test de Fermat

Fermat Primality Test

Input: prime candidate \tilde{p} and security parameter s

Output: statement " \tilde{p} is composite" or " \tilde{p} is likely prime"

Algorithm:

- ```

1 FOR $i = 1$ TO s
1.1 choose random $a \in \{2, 3, \dots, \tilde{p}-2\}$
1.2 IF $a^{\tilde{p}-1} \not\equiv 1 \pmod{\tilde{p}}$
1.3 RETURN (" \tilde{p} is composite")
2 RETURN (" \tilde{p} is likely prime")

```

Observación: No hay falsos negativos.

Problema: los números de Carmichael<sup>C</sup> pasan el test para todo  $a \neq 1$  tq  $\gcd(a, C) = 1$

Aún con dicho problema, los números problemáticos son raros. Dicho eso, con los números de Carmichael pueden haber complicaciones (puede que sea primo relativo con muuuuchos números).

## Test de Miller-Rabin

Este test se basa en el siguiente teorema

**Teorema:** Sea  $\tilde{p}$  un candidato a primo y tómese la descomposición en primos de  $\tilde{p}-1$  siguiente:  $\tilde{p}-1 = 2^u r$ , con  $r$  impar.

Si  $\exists a \in \mathbb{Z} \forall j \in \{0, 1, \dots, u-1\}$  tq  $a^r \not\equiv 1 \pmod{\tilde{p}}$  &  $a^{r^{2^j}} \not\equiv \tilde{p}-1 \pmod{\tilde{p}}$ , ent.  $\tilde{p}$  no es primo. En caso contrario, es probable que sea primo.

Con este teorema se tiene el algoritmo siguiente.

## Miller–Rabin Primality Test

**Input:** prime candidate  $\tilde{p}$  with  $\tilde{p} - 1 = 2^u r$  and security parameter  $s$

**Output:** statement “ $\tilde{p}$  is composite” or “ $\tilde{p}$  is likely prime”

**Algorithm:**

```

1 FOR $i = 1$ TO s
 choose random $a \in \{2, 3, \dots, \tilde{p} - 2\}$
1.2 $z \equiv a^r \pmod{\tilde{p}} \rightsquigarrow S_q + \text{Mult.}$
1.3 IF $z \not\equiv 1$ AND $z \not\equiv \tilde{p} - 1$ ← Teorema con $j = 0$
 $j = 1$
1.4 WHILE $j \leq u - 1$ AND $z \not\equiv \tilde{p} - 1$ ←
 $z \equiv z^2 \pmod{\tilde{p}}$
 IF $z \equiv 1$ RETURN (“ \tilde{p} is composite”) ← Teorema, $j = 1, \dots, u-1$, con
 ELSE $j = j + 1$
1.6 IF $z \not\equiv \tilde{p} - 1$ RETURN (“ \tilde{p} is composite”) ← Teorema, $j = u$
2 RETURN (“ \tilde{p} is likely prime”)

```

La probabilidad de falsos positivos es muy baja conforme  $l(\tilde{p})$  crece.

**Table 7.2** Number of runs within the Miller–Rabin primality test for an error probability of less than  $2^{-80}$

| Bit length of $\tilde{p}$ | Security parameter $s$ |
|---------------------------|------------------------|
| 250                       | 11                     |
| 300                       | 9                      |
| 400                       | 6                      |
| 500                       | 5                      |
| 600                       | 3                      |

Ejemplo:  $\tilde{p} = 91 \rightsquigarrow \tilde{p}-1 = 2 \cdot 45$ . Tomamos  $s = 4$ .

Números a generar:

1)  $a = 12 \Rightarrow z = 12^{45} \equiv 90 \pmod{91} \rightsquigarrow$  probablemente sí

$$2) a = 17 \Rightarrow z = 17^{45} \equiv 90 \pmod{91} \rightsquigarrow \text{probablemente s\'i}$$

$$3) a = 38 \Rightarrow z = 38^{45} \equiv 90 \pmod{91} \rightsquigarrow \text{probablemente s\'i}$$

$$4) a = 39 \Rightarrow z = 39^{45} \equiv 78 \pmod{91} \rightsquigarrow \text{es compuesto.}$$

Si a da un falso positivo se le llama a ese valor un mentiroso para  $\tilde{p}$ .