Matemática Computacional – Laboratorio 5

1. Pequeño teorema de Fermat

En 1640, el matemático Pierre de Fermat enunció el teorema siguiente:

Teorema 1 *Para cualquier primo p y cualquier entero a tal que $\gcd(a,p) = 1$, tenemos*

$$a^{p-1} \equiv a \text{ mód } p.$$

Este teorema fue generalizado por Euler en 1736 como:

Teorema 2 *Para cualquier entero positivo n y cualquier entero a tal que $\gcd(a,n) = 1$, tenemos*

$$a^{\phi(n)} \equiv a \text{ mód } n$$

donde $\phi(n)$ es el número de enteros entre 0 y n tales que $\gcd(a,n) = 1$.

Además, la función $\phi(n)$ se puede calcular facilmente:

- para cualquier primo p, $\phi(p) = p - 1$;
- si $\gcd(a,b) = 1$, entonces $\phi(a \cdot b) = \phi(a) \cdot \phi(b)$;
- para cualquier primo p y cualquier entero positivo k, $\phi(p^k) = (p-1)p^{k-1}$;

2. Criptosistema RSA

El esquema de cifrado RSA de R. L. Rivest, A. Shamir y L. M. Adleman en 1978, es actualmente el esquema criptográfico asimétrico más ampliamente utilizado, a pesar de que las curvas elípticas y esquemas de logaritmos discretos están ganando terreno. Hay muchas aplicaciones para RSA, pero en la práctica se utiliza con más frecuencia para:

- Cifrado de pequeñas piezas de datos, especialmente para la distribución de claves.
- Firmas digitales, por ejemplo, para los certificados digitales en Internet.

El funcionamiento del sistema se basa en el pequeño teorema de Fermat: Para preparar el sistema, un usuario $\mathcal{B}$ elige dos primos p y q y se calcula $n = p \cdot q$ y $\phi(n) = (p-1) \cdot (q-1)$. Además, $\mathcal{B}$ elige un valor e tal que $\gcd(n,e) = 1$ y calcula $d \equiv e^{-1}$ mód n (via el algoritmo Euclidiano extendido). Los valores de n y e son disponibles públicamente (la *clave publica* de $\mathcal{B}$), y el valor de d se guarda secreto (la *clave secreta* de $\mathcal{B}$).

Para enviar un mensaje m (tal que $\gcd(n,m)=1$) en secreto a $\mathcal{B}$, se calcula (y manda)

$$x \equiv m^e \text{ mód } n.$$

Para leer el mensaje, $\mathcal{B}$ calcula

$$y \equiv x^d \text{ mód } n,$$

lo que, por el pequeño teorema de Fermat, da, ya que $e \cdot d = 1 + r\phi(n)$ por construcción,

$$y \equiv (m^e)^d \equiv m^{ed} \equiv m^{1+r\phi(n)} \equiv m \cdot (m^{\phi(n)})^r \equiv m \cdot 1^r \equiv m \text{ mód } n.$$

El argumento principal para la seguridad del sistema RSA es la dificultad de factorizar un entero n en números primos, dado que una vez que se conocen p y q es sencillo calcular d y leer el mensaje.

3. Algoritmo $p-1$ de Pollard

En 1974, Pollard presentó un algoritmo que permite factorizar un entero n si uno de sus factores primos p es tal que $p-1$ se factoriza en números pequeños, en particular si $p-1$ divide a $B!$ para un valor de B suficientemente pequeño.

Se basa en el pequeño teorema de Fermat. Sean p y q dos factores primos ($p \neq q$) de n tales que $(p-1)$ divide a $B!$ pero $(q-1)$ no divide a $B!$, entonces existe un entero $a \in \{1, 2, \ldots, n-1\}$, $\gcd(a,n) = 1$ tal que

$$a^{B!} \equiv 1 \text{ mód } p \qquad \text{y} \qquad a^{B!} \not\equiv 1 \text{ mód } q,$$

lo que permite de concluir que

$$\gcd(a^{B!} - a, n) \neq 1 \text{ o } n,$$

ya que p divide a $\gcd(a^{B!} - a, n)$ y q no divide a $\gcd(a^{B!} - a, n)$, por lo tanto, $\gcd(a^{B!} - a, n)$ nos da un factor (no-trivial) de n.

Cuando n es el producto de dos primos distintos (como es el caso para RSA), obtenemos la factorización completa de n. La versión siguiente del algoritmo $p-1$ de Pollard busca el valor mínimo de B para cual existe un factor primo p de n tal que $(p-1)$ divide a $B!$, y en general los otros factores primos de n nos satisfarán esta condición.

En general, se puede reemplazar la elección aleatoria de a con una selección determinada: los primos en orden creciente desde 2. Según la hipótesis de Riemann generalizada (una conjetura de teoría de número), el primo a más grande que necesita el algoritmo $p-1$ es de la forma $(\log n)^6$.

Observación 1 *En la práctica, en vez de aumentar la potencia de "a poco", se elige una cota B y se calcula*

$$a^{M_B} \text{ mód } n$$

donde

$$M_B = \prod_{j=1}^{B} j,$$

Algorithm 1 Algoritmo $p-1$ de Pollard

Require: Entero no-primo (y no una potencia de primo) n.
Ensure: Factores n_1 y n_2 de n, ambos > 1.

```
1: Elegir a al azar,   1 < a < n − 1,   gcd(a, n) = 1
2: B ← 2,   a ← a^B mód n,   n_1 = gcd(a − 1, n)
3: while (n_1 = 1) do
4:     B ← B + 1,   a ← a^B mód n,   n_1 = gcd(a − 1, n)
5: end while
6: if (n_1 = n) then
7:     volver al paso 1
8: else
9:     return n_1 y n/n_1.
10: end if
```

utilizando la representación binaria de M_B para reducir el costo computacional de la exponenciación. Se puede considerar que la primera descripción del algoritmo corresponde a un aumento progresivo del valor de B.

Observación 2 *Para mejorar un poco el algoritmo (hacerlo un poco más rápido), se puede reemplazar M_B con*

$$M_B = \mathrm{lcm}\{1, 2, \ldots, B-1, B\}$$

donde $\mathrm{lcm}\{S\}$ *es el mínimo común multiplo (*least common multiple*) de los enteros en S. Para eso, se utiliza que*

$$\mathrm{lcm}\{a, b\} = \frac{a \cdot b}{\gcd\{a, b\}}.$$

y

$$\mathrm{lcm}\{a, b, c\} = \mathrm{lcm}\{\mathrm{lcm}\{a, b\}, c\}$$

4. Laboratorio

- Desarrolle un programa en lenguaje C para factorizar enteros con el algoritmo $p-1$.
- Si utiliza GMP, factorizar los enteros siguiente:

$$n = 45524252104894451218081,$$

$$n = 28742705413$$

y

$$\begin{aligned} n = {} & 1765068412026960157182063042134794330393647441981256398386787605436591089603 1\\ & 3511850769801430545505304905236884925496942520836944891853086319792640094911 4\\ & 7114269906793409728751442583707710531811860634976466174213205195219829769309 9\\ & 7232269627523745193592835712014272970401943641392537038020023689053156818295 61. \end{aligned}$$

- Si no utiliza GMP, factorizar los enteros siguiente:

$$n = 691357153,$$

$$n = 2877921703$$

y

$$n = 3601478521$$

5. Se solicita

1. Implementar los algoritmos en lenguaje C.
2. Dar la factorización de los 3 enteros correspondientes a su implementación, y el valor de B para cual el algoritmo terminó.
3. Observar los tiempo de cálculo para los enteros dados.
4. Se debe entregar:
   - Código fuente de los algoritmos.
   - Archivo "Readme" con las instrucciones (detalladas) de compilación.
   - Informe en LaTeX que contiene:
     - Detalles de los algoritmos implementados.
     - Formulación de los experimentos.
     - Información del hardware (procesador) y software (librerias) utilizado.
     - Conclusiones.
5. Normas a cumplir:
   a) Número de Integrantes: máximo 2.
   b) Plazo de Entrega: 14 días.
   c) Forma de envío: se envía directorio comprimido zip, gz, tar, o tgz (no se aceptan otro formatos). a: **nicolas.theriault@usach.cl**
6. El nombre del archivo (y directorio) debe indicar el laboratorio ("Lab1") y a lo menos las iniciales (nombre y apellido) de cada integrante.
   - Se puede utilizar solamente el primer nombre y primer apellido de cada integrante.
   - No deber haber acentos (á → a, ñ → n, etc).
   - No deber haber espacios en blanco, utilizar "_" para separar palabras e iniciales.
   - Ejemplo: Rodrigo German Abarzúa Ortiz y Nicolas Thériault → Lab4_RA_NT.

6. Evaluación

La nota del laboratorio se calculará según la ponderación siguiente:

- Informe [**20 %**]:
  El informe está en LaTeX, en lenguaje formal, y contiene todos los detalles exigidos.

- Algoritmos y análisis [**40 %**]:
  Describe todas las funciones (de libreria) utilizadas en la implementación. Factoriza los enteros pedidos. Llega a una hipótesis razonable sobre cual es el factor dominante en la complejidad (costo) del algoritmo.

- Implementación [**40 %**]
  El código fuente no presenta errores o advertencias de compilación. El programa está escrito de forma que puede ser leído y/o re-utilizado fácilmente por otros programadores: la redacción es limpia (con espacios y divisiones claras) y bien documentada, las sub-funciones y las variables tienen nombres naturales (que indican a que sirven) o acompañadas de comentarios aclarando a que sirven.