

Criptografía basada en PLD.

Implementación de algoritmos de cifrado basados en el problema matemático del logaritmo discreto.

Por Marta Pastor Puente

La criptografía se considera un pilar fundamental en la seguridad a la hora de establecer comunicaciones. Los métodos de encriptación que se fundamentan en técnicas de cifrado criptográficas han ido evolucionando a lo largo de la historia, volviéndose cada vez más efectivas y seguras.

Este trabajo propone un estudio más exhaustivo y la posterior implementación de algunos métodos de encriptación no vistos en clase, en concreto los algoritmos de Diffie-Hellman y ElGamal que se fundamentan en el modelo de logaritmo discreto del álgebra aplicada, así como del algoritmo de Pohlig-Hellman que surge como contraataque a dichos algoritmos de encriptación y que permite computar logaritmos discretos de grupos abelianos, aunque con una elevada complejidad, especialmente si los algoritmos de cifrado emplean como módulo números primos.

Índice:

1. Teoría de la complejidad
2. Problema del logaritmo discreto
3. Intercambios de claves
4. Criptosistemas
5. Algoritmos de factorización
6. Implementación en Java

Teoría de la complejidad:

Es una rama de la teoría de la computación que se centra en la clasificación de los problemas computacionales de acuerdo a su dificultad inherente y en la relación entre dichas clases de complejidad, es decir, jerarquiza la dificultad de los problemas en relación a los algoritmos que los resuelven.

Hay que hacer una distinción entre el concepto de análisis de algoritmos y la teoría de la complejidad, ya que mientras el análisis se encarga de determinar la cantidad de recursos requeridos por **un algoritmo en particular** para resolver un problema, la teoría por su parte analiza **todos los posibles algoritmos** que pudieran ser usados para resolver el mismo problema.

La teoría de la complejidad abarca dos grandes grupos de problemas: los problemas de decisión y los problemas de búsqueda. Dentro de los primeros a su vez podemos subdividir los problemas en dos clases:

- La clase P que contempla los problemas para los que existe un algoritmo conocido de complejidad polinomios que los resuelve.
- La clase NP que contempla los problemas para los que hay un algoritmo que los resuelve pero no es de complejidad polinomial y, no obstante, dada una posible solución del problema, sí se puede determinar si efectivamente es solución en tiempo polinomial.

Algunos problemas de esta última clase NP han servido para desarrollar potentes sistemas criptográficos, donde pueden destacarse el problema de factorización de enteros que da lugar al sistema RSA (River-Shamir-Adleman) o el problema del logaritmo discreto en el que nos centraremos en este trabajo y del que surgen sistemas como el DHM (Diffie-Hellman-Merkle).

Problema del logaritmo discreto:

En Álgebra, se conoce como logaritmo discreto de y en base g , donde g e y son elementos de un grupo cíclico finito G^1 , a la solución x de la ecuación $g^x = y$, tal que

$$x = \log_g(y) \iff g^x = y$$

Repaso

Un grupo es un conjunto de elementos sobre los cuales se define una operación binaria, que notaremos como $*$, para la cual se verifican las siguientes propiedades:

- Asociativa: para cualquier $a, b, c \in G$, $(a * b) * c = a * (b * c)$
- Identidad: existe un elemento $e \in G$ tal que para todo $x \in G$, $e * x = x * e = x$
- Inversa: para todo $x \in G$, existe un $x^{-1} \in G$ tal que $x * x^{-1} = x^{-1} * x = e$

siendo a, b y c elementos del grupo G .

Un elemento a se dice que genera un grupo G si $a * a = a^2$, $a * a * a = a^3$, etc, son todos los elementos del grupo. En ese caso, decimos que a es el generador de G y G es un grupo cíclico.

Un grupo G se dice que es cíclico si existe un elemento $a \in G$ tal que para todo $b \in G$, $b = a^j$ para algún entero j .

El orden de un elemento x es n , si n es el menor entero tal que $x^n = e$. El número de elementos de un grupo es el orden del grupo.

¹ Sea un grupo cíclico finito $G = \langle g \rangle$ con $\text{ord}(g) = n$ y $x \in G$.

Intercambios de claves:

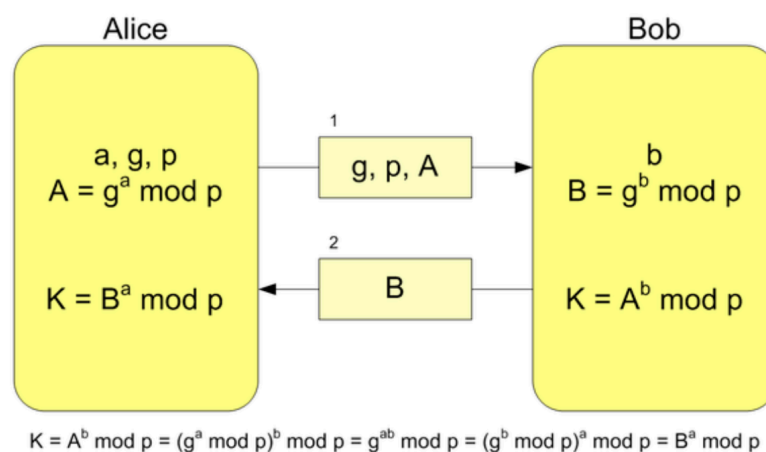
Antes de nada, debemos definir el concepto de protocolo como el conjunto de pasos ordenados que los usuarios de un sistema tienen que seguir para realizar una tarea.

En concreto, el protocolo de intercambio de claves es un **protocolo criptográfico** por el que dos o más participantes se ponen de acuerdo en el valor de una información secreta compartida. A la información secreta compartida se le suele llamar **clave** debido a que esa información se suele usar como clave posterior en algún algoritmo criptográfico.

El **algoritmo de Diffie-Hellman** para intercambio de claves se basa en la dificultad de resolución del problema del logaritmo discreto en los grupos F_q^* .

Repaso

Un cuerpo finito F_q^* , campo finito o campo de Galois es un cuerpo definido sobre un conjunto finito de elementos. Además, todos los cuerpos finitos tienen un número de elementos $q = p^n$, siendo p un número primo y n un número entero.

**Ejemplo**

1. Alice elige un número secreto $a = 6$, un número primo $p = 23$ y la base $g = 5$. Le envía a Bob los siguientes datos:

$$A = (g^a \bmod p)^5^6 \bmod 23 = 8$$

$\begin{matrix} p & 23 \\ g & 5 \end{matrix}$

2. Bob elige un número secreto $b = 15$ y calcula la clave secreta compartida:

$$K = (g^a \bmod p)^b \bmod p \quad 8^{15} \bmod 23 = 2$$

3. Bob cifra el texto plano del mensaje usando la clave secreta compartida.

4. Bob envía a Alice el texto cifrado y su clave pública:

$$B = (g^b \bmod p)^{5^{15}} \bmod 23 = 19$$

5. Alice calcula $K = (g^b \bmod p)^a \bmod p$ por las propiedades del grupo Z_q^* .²

$$K = (g^a \bmod p)^b \bmod p = 19^6 \bmod 23 = 2$$

y usa la clave generada para descifrar los datos enviados por Bob.

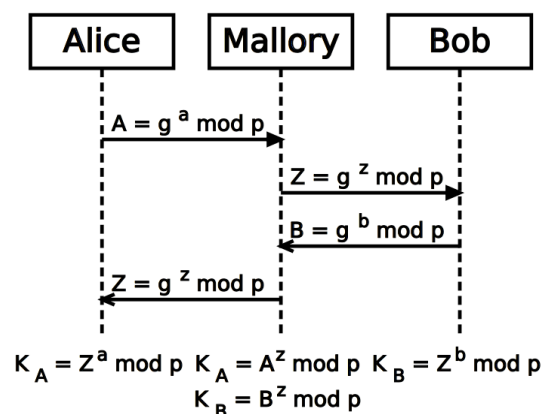
Así pues, un intruso que interceptara la comunicación entre Alice y Bob podría calcular la clave secreta compartida únicamente si tuviera también uno de los números secretos a o b , los cuales en ningún momento se envían por el canal entre Alice y Bob.

Por lo tanto, la única forma de conseguir esos números secretos a o b sería a partir de A o B invirtiendo la función

$$a = \log_{\text{disc}_q}(A) \quad \text{y} \quad b = \log_{\text{disc}_q}(B)$$

y es aquí donde yace la dificultad de resolución del problema del logaritmo discreto en Z_q^* , un problema que se cree actualmente intratable computacionalmente.

Sin embargo, este sistema de intercambio de claves es vulnerable a ataques man-in-the-middle, ya que si el intruso intercepta la comunicación antes de que Alice y Bob hayan intercambiado la clave secreta compartida K , la seguridad de la transmisión se vería comprometida.



Criptosistemas:

La forma en que la criptografía protege la información es variando su forma. Dado un texto original se llama **cifrado** a una transformación del mismo al llamado texto cifrado. La transformación que permite recuperar el texto original a partir del texto cifrado se llama descifrado.

² Si q es un número primo (como es nuestro caso), $Z_q^* = F_p^*$.

La familia de transformaciones posibles se denomina criptosistema y cada una de ellas está determinada por un parámetro llamado clave. De manera más formal:

Un criptosistema es una terna (M, C, K) , donde M es el conjunto de mensajes originales, C es un conjunto de mensajes cifrados y K es un conjunto finito de claves, junto con dos funciones:

$$c: M \times K \rightarrow C \quad \text{y} \quad d: C \times K \rightarrow M$$

tales que $d(c(M, k), k) = M$ para todo $(M, k) \in M \times K$. Las funciones c y d reciben el nombre de funciones de cifrado y descifrado respectivamente.

Los criptosistemas pueden dividirse en sistemas de clave privada y sistemas de clave pública. En los sistemas de clave privada los participantes guardan en secreto las funciones c y d (o, al menos, la clave privada de las que dependen). Por el contrario, en los sistemas de clave pública, cada usuario i tiene un par de claves (c_i, d_i) . La primera de ellas, c_i , es la clave pública y es la que utiliza cualquier otro usuario j que quiera transmitir un mensaje M a i . Esto quiere decir que j cifra el mensaje en la forma $C = c_i(M)$. La segunda clave, d_i , es privada y empleada por i para recuperar los mensajes cifrados que recibe. Esto es, $M = d_i(C) = d_i(c_i(M))$.

El **criptosistema ElGamal** hace referencia a un esquema de cifrado asimétrico basado en el problema matemático del logaritmo discreto, propuesto por Taher ElGamal en 1985 y que se basa en que se conocen el cuerpo F_q y un elemento primitivo g del mismo. Se usa en software GNU Privacy Guard, versiones recientes de PGP, y no se encuentra bajo ninguna patente, lo que lo convierte en software de uso libre.

Repaso

Pierre de Fermat, en 1636, enunció lo que actualmente se conoce como el Pequeño Teorema de Fermat:

Si p es un número primo, entonces, para cada número natural a , con $a > 0$, y coprimo con p , $a^{p-1} \equiv 1 \pmod{p}$.

$$y_1^{-a} = y_1^{p-1-a}$$

es decir, que si se eleva un número a a la p -ésima potencia y al resultado se le resta a , lo que queda es divisible por p .

Primero, cada participante necesita generar dos valores para su clave: uno valor público y otro privado.

Público

- 1 Un número primo p y un entero generador g de \mathbb{Z}_p
- 2 La clave pública $y \equiv g^x \pmod{p}$

Privado

- 1 Un entero x tal que $1 < x < p - 1$.

A continuación, se lleva a cabo el proceso de cifrado y descifrado siguiendo los siguientes pasos:

Cifrado (Acceso a la parte pública)

- 1 Escoger el mensaje a cifrar $1 < M < p$.
- 2 Generar número aleatorio $1 < k < p - 1$ con $(k, p - 1) = 1$.
- 3 Generar el par (r, s) para enviar con

$$r \equiv g^k \pmod{p}$$

$$s \equiv My^k \pmod{p}$$

Descifrado (Acceso a la parte privada)

- 1 Obtener $C = (r, s)$
- 2 $M \equiv \frac{s}{r^x} \pmod{p}$

Ejemplo

1. Alicia elige los valores:

p = 15485863 (primo aleatorio y para $p-1 = 15485862 = 3 * 2 * 72 * 52673$)
g = 7 (generador aleatorio)
a = 21702 (clave privada aleatoria)

Alicia calcula:

$$K = g^a \pmod{p} = 7^{21702} \pmod{15485863} = 8890431$$

La clave pública será **(7, 15485863, 8890431)**

2. Bob quiere transmitir el número **m = 128688**. Bob elige un **b = 480** aleatorio entre 2 y $p-1$ y calcula:

$$y_1 = g^b \pmod{p} = 7^{480} \pmod{15485863} = 12001315$$

$$y_2 = Kbm \pmod{p} = 8890431^{480} * 1286889 \pmod{15485863} = 826$$

El texto cifrado es **C_b (m, b)** con los valores **(y₁ = 12001315, y₂ = 8263449)**

3. Para descifrarlo, Alicia usa su clave privada **a = 21702** y por el Pequeño Teorema de Fermat:

$$y_1^{-a} y_2 \pmod{p} = 14823281 * 8263449 \pmod{15485863} = 128688$$

Algoritmos de factorización:

Actualmente, no se conocen algoritmos de factorización con complejidad polinomial que pudieran resolver el problema del logaritmo discreto. Un método eficaz para resolver el dicho problema pondría en jaque la seguridad de todos los sistemas basados en él que son utilizados hoy en día.

El mejor algoritmo conocido para atacar aquellos sistemas criptográficos basados en el logaritmo discreto actualmente es probabilista de complejidad subexponencial. El **algoritmo de Silver-Pohlig-Hellman** es un método utilizado para resolver el problema del logaritmo discreto en grupos cuyo orden n tiene divisores primos pequeños. Por lo tanto, no puede aplicarse en cualquier grupo, pero en aquellos en los que puede utilizarse resulta más eficiente que otros métodos genéricos.

La ineficiencia de este algoritmo de factorización reside en la elección de un número primo lo suficientemente elevado como para que el grupo generado tenga un orden tan alto que la complejidad sea como poco probabilista.

Implementación en Java:

A la hora de buscar un enfoque más práctico del trabajo, se optó por llevar a cabo la implementación de los algoritmos previamente presentados en un lenguaje de programación que permitiera una implementación rápida y lógica.

Por ello, se ha llevado a cabo la implementación en Java³ de los algoritmos de intercambio de claves Diffie-Hellman y el criptosistema ElGamal.

En el caso de Diffie-Hellman:

```
alice.generateKeys();
bob.generateKeys();
```

```
alice.receivePublicKeyFrom(bob);
bob.receivePublicKeyFrom(alice);
```

```
alice.generateCommonSecretKey();
bob.generateCommonSecretKey();
```

```
alice.encryptAndSendMessage("Bob! Guess who I am!", bob);
```

```
bob.tellTheMessage(secretMessage); // Message is still not decrypted
```

```
bob.receiveAndDecryptMessage(secretMessage);
bob.tellTheMessage(); // Message is already decrypted
```

Alice genera la clave secreta compartida usando su clave privada y la clave pública de Bob.

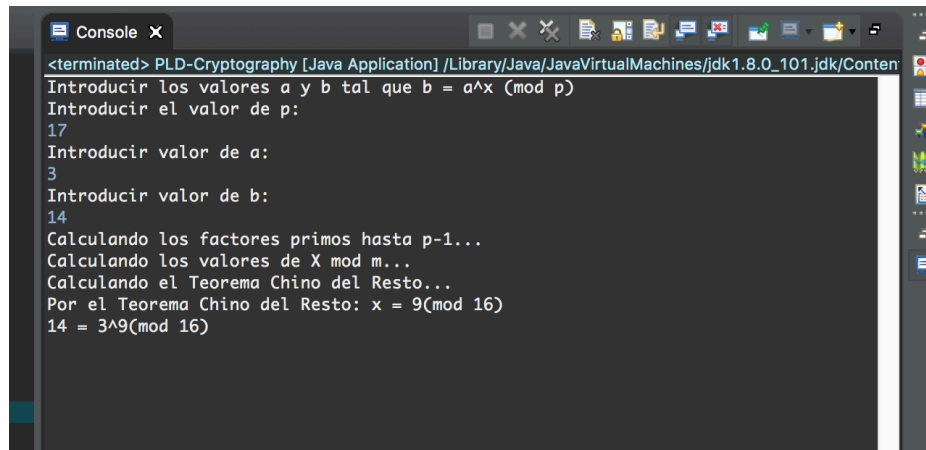
Bob genera la clave secreta compartida usando su clave privada y la clave pública de Alice.

Ambas claves secretas compartidas son iguales sin necesidad de transferirlas, y es aquí donde reside el encanto del algoritmo Diffie-Hellman.

³ El código fuente de la implementación se puede encontrar en [Github](#).

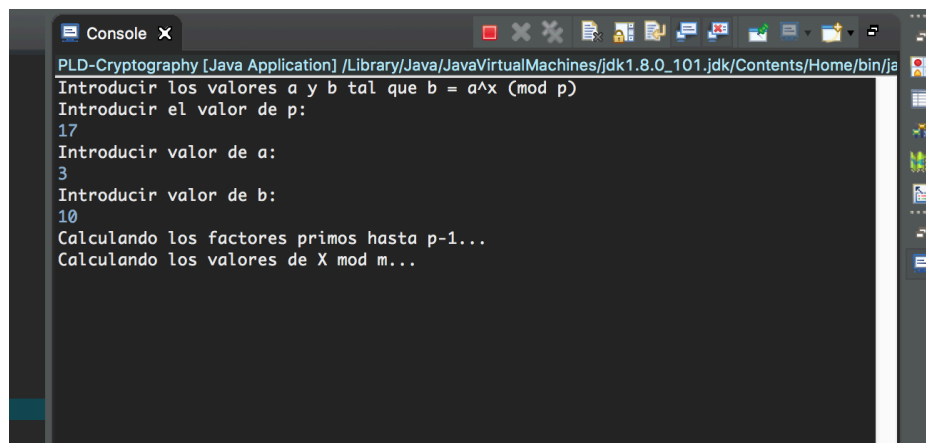
En el caso de EIGamal:

Implementación incompleta.

En el caso del algoritmo de factorización Pohlig-Hellman:

```
<terminated> PLD-Cryptography [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java
Introducir los valores a y b tal que b = a^x (mod p)
Introducir el valor de p:
17
Introducir valor de a:
3
Introducir valor de b:
14
Calculando los factores primos hasta p-1...
Calculando los valores de X mod m...
Calculando el Teorema Chino del Resto...
Por el Teorema Chino del Resto: x = 9(mod 16)
14 = 3^9(mod 16)
```

Podemos comprobar que para casos en los que la x no es un factor primo, como en **$14 = 3^x \pmod{17}$** , el programa encuentra el valor de dicha x en apenas unos milisegundos, mientras que para un valor primo, por muy bajo que sea, como **$x = 3$** , el programa no es capaz de encontrar la solución gracias a la dificultad de resolución inversa del problema de logaritmo discreto.



```
PLD-Cryptography [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java
Introducir los valores a y b tal que b = a^x (mod p)
Introducir el valor de p:
17
Introducir valor de a:
3
Introducir valor de b:
10
Calculando los factores primos hasta p-1...
Calculando los valores de X mod m...
```