Proyecto 3

Elliptic-Curve factorization Method (ECM)

Información del curso

Criptografía y Seguridad - Facultad de Ciencias, UNAM.

- Profesor: Manuel Díaz Díaz
- Ayudante: Gerardo Rubén López Hernández
- Laboratorio: José Canek García Aguilar

Descripción de la práctica

Implementación el algoritmo de Elliptic-Curve factorization Method para la factorización de un número `n` de la forma `n=pq`.

En el archivo de especificación de la prática viene todo explicado `doc.pdf`.

Entorno

- **`OS`**: Ubuntu 18.04.2 LTS o macOS Mojave 10.14.5
- **`SDKMAN`**: SDKMAN 5.7.3+337 *(Para la instalación de `Kotlin` y `Java`)*
- **`Java`**: OpenJDK Runtime Environment Zulu11.31+11-CA (build 11.0.3+7-LTS)
- **`Kotlin`**: Kotlin 1.3.31
- **`make`**: GNU Make 3.81

Instalación de SDKMAN!

Recomiendo instalar `Java` y `Kotlin` con `SDKMAN!` porque es más sencilla su instalación.

```
$ curl -s "https://get.sdkman.io" | bash
```

Abre una nueva terminal y pon lo siguiente:

```
$ source "$HOME/.sdkman/bin/sdkman-init.sh"
```

Por último, para verificar que fue instalado satisfactoriamente, ejecutar el siguiente comando:

```
$ sdk version
```

Si la instalación fue satisfactoria se mostrará un mensaje como el siguiente:

> SDKMAN 5.7.3+337

Si eres curioso puedes ver la página oficial y ver la parte de la instalación.

Instalación de Java

Omitirse este paso si se tiene instalado `Java` con al menos la versión 11.

Instalación de Java:

```
$ sdk install java 11.0.3-zulu
```

Verificar si se instaló correctamente:

```
$ java -version
```

Si se instaló correctamente, deberá aparecer un mensaje similar al siguiente: > openjdk version "11.0.3" 2019-04-16 LTS > > OpenJDK Runtime Environment Zulu11.31+11-CA (build 11.0.3+7-LTS) > > OpenJDK 64-Bit Server VM Zulu11.31+11-CA (build 11.0.3+7-LTS, mixed mode)

Instalación de Kotlin

Instalación de Kotlin:

```
$ sdk install kotlin 1.3.31
```

Verificar si se instaló correctamente:

```
$ kotlin -version
```

Si se instaló correctamente, deberá aparecer un mensaje similar al siguiente: > Kotlin version 1.3.31-release-197 (JRE 11.0.3+7-LTS)

Ejecución del programa

Para facilitar la compilación, ejecución y limpieza del proyecto se optó por usar la herramienta llamada `make` con diferentes *targets*.

Para compillar es programa, se deberá tener situada una terminal en la raíz del proyecto y ejecutar:

```
$ make compile
```

Para la ejecución del programa:

```
$ make run [n=N] [limit=M]
```

Donde `N` es el número producto de dos primos y `M` es la cota superior para la generación de números primos usando la criba de Eratosthenes.

Para limpiar el proyecto se deberá ejectar el comando:

```
$ make clean
```

Ejemplo de ejecución:

```
$ make run n=97343 limit=1500
```

Comentarios

Los valores deberán ser pasados como argumento en el `make run`.

Algunos valores de prueba:

- 1460783 = 1559 * 937
- 548857 = 457 * 1201
- 12731 = 439 * 29
- 26867 = 401 * 67

Al finalizar la ejecución del programa se mostrará los factores del número `n`.

Para la implementación del algoritmo se basó en la siguiente bibliografía:

- Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. 1996. Handbook of Applied Cryptography (1st ed.). CRC Press, Inc., Boca Raton, FL, USA.
- Daniel Parker, *ELLIPTIC CURVES AND LENSTRA'S FACTORIZATION ALGORITHM*.

Integrante(s)

- Ángel Iván Gladín García - *angelgladin@ciencias.unam.mx*
- Alinka Atenas Fragoso Martínez - *alinkafm@ciencias.unam.mx*