Universidad Nacional Autónoma de México

Facultad de Ciencias

Programación Paralela

Reporte de Exposición

Alumno:
Andrade Hernández Carlos

Semestre 2023-2

Historia En el ámbito de la criptografía, RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1979. Este algoritmo se basa en el desafío matemático de la factorización de números enteros. Es ampliamente utilizado y considerado como el primer algoritmo de clave pública.

La seguridad de RSA se basa en la dificultad de factorizar números enteros grandes. En este sistema, los mensajes se representan como números y se utilizan dos números primos grandes cuidadosamente seleccionados y mantenidos en secreto. Estos números primos son del orden de 10^300 en la actualidad, y se espera que su tamaño aumente continuamente a medida que avanza la capacidad de cálculo de los ordenadores. Esta propiedad garantiza la seguridad y confidencialidad de las comunicaciones encriptadas mediante RSA.

Algoritmo RSA El algoritmo RSA es un algoritmo de criptografía asimétrica. Asimétrico significa que funciona con dos claves diferentes: la clave pública y la clave privada. Como su nombre lo indica, la clave pública se proporciona a todos, mientras que la clave privada se mantiene en secreto.

Un ejemplo de criptografía asimétrica:

- Un cliente (por ejemplo, un navegador) envía su clave pública al servidor y solicita algunos datos.
- El servidor encripta los datos utilizando la clave pública del cliente y envía los datos encriptados. El cliente recibe estos datos y los desencripta. Dado que esto es asimétrico, nadie más que el navegador puede desencriptar los datos, incluso si un tercero tiene la clave pública del navegador.

La idea de RSA se basa en el hecho de que es difícil factorizar un número entero grande. La clave pública consta de dos números, donde uno de ellos es el resultado de la multiplicación de dos números primos grandes. Y la clave privada también se deriva de los mismos dos números primos. Por lo tanto, si alguien puede factorizar el número grande, la clave privada se ve comprometida. Por lo tanto, la fortaleza de la encriptación depende por completo del tamaño de la clave, y si duplicamos o triplicamos el tamaño de la clave, la fortaleza de la encriptación aumenta exponencialmente.

Ejemplo:

Selecciona dos números primos. Supongamos que P = 53 y Q = 59. Ahora, la primera parte de la clave pública es: n = P*Q = 3127. También necesitamos un exponente pequeño, digamos e: Pero e debe ser:

- Un número entero.
- No ser un factor de Φ(n).
- 1 < e < Φ(n)

> Generando Clave Privada:

Necesitamos calcular Φ(n): Tal que Φ(n) = (P-1)(Q-1) Entonces, Φ(n) = 3016 Ahora calculamos la Clave Privada, d: d = (k*Φ(n) + 1) / e para algún entero k Para k = 2, el valor de d es 2011.

Ahora estamos listos con nuestra - Clave Pública (n = 3127 y e = 3) y Clave Privada (d = 2011). Ahora vamos a encriptar "HI":

Convertimos las letras a números: H = 8 e I = 9 Por lo tanto, Datos Encriptados c = (89^e)mod n Así que nuestros Datos Encriptados resultan ser 1394

Ahora vamos a desencriptar 1394: Datos Desencriptados = (c^d)mod n Así que nuestros Datos Desencriptados resultan ser 89 8 = H e I = 9, es decir, "HI".

RSA.hs El archivo RSA.hs funciones relacionadas con el algoritmo RSA (Rivest-Shamir-Adleman) para encriptación asimétrica. A continuación, se proporciona una documentación para cada función:

- `primos :: Integer -> [Integer]`
  - **Descripción**: Esta función devuelve una lista de los primeros `n` números primos utilizando el método del tamiz de Eratóstenes.
  - **Entrada**: Un entero `n` que representa la cantidad de números primos que se desean obtener.
  - **Salida**: Una lista de enteros que contiene los `n` primeros números primos.
- `elementoAleatorio :: [Integer] -> StdGen -> (Integer, [Integer], StdGen)`
  - **Descripción**: Esta función toma una lista de enteros y una semilla aleatoria, y devuelve un elemento aleatorio de la lista junto con la lista actualizada sin ese elemento.
  - **Entrada**:
    - Una lista de enteros.
    - Un generador de números aleatorios `StdGen` .
  - **Salida**: Una tupla que contiene el elemento aleatorio seleccionado, la lista actualizada sin ese elemento y el nuevo generador de números aleatorios.
- `generaLlaves :: Int -> (Integer, Integer, Integer)`
  - **Descripción**: Esta función genera las llaves pública y privada junto con el valor `n` requerido para el algoritmo RSA. Utiliza primos generados aleatoriamente y encuentra los valores adecuados para `e` y `d` según las reglas del algoritmo.
  - **Entrada**: Un entero que se utiliza como semilla para la generación aleatoria de primos.
  - **Salida**: Una tupla que contiene la llave pública, la llave privada y el valor `n` .
- `encripta :: Integer -> Integer -> Integer -> Integer`
  - **Descripción**: Esta función encripta un mensaje utilizando la llave pública y el valor `n` proporcionados en el algoritmo RSA.
  - **Entrada**:
    - Un entero que representa la llave pública.
    - Un entero que representa el valor `n` .
    - Un entero que representa el mensaje a encriptar.
  - **Salida**: El mensaje encriptado como un entero.
- `desencripta :: Integer -> Integer -> Integer -> Integer`
  - **Descripción**: Esta función desencripta un texto encriptado utilizando la llave privada y el valor `n` proporcionados en el algoritmo RSA.
  - **Entrada**:
    - Un entero que representa la llave privada.
    - Un entero que representa el valor `n` .
    - Un entero que representa el texto encriptado.
  - **Salida**: El texto desencriptado como un entero.
- `encriptaCadena :: Integer -> Integer -> String -> [Integer]`
  - **Descripción**: Esta función encripta una cadena de texto utilizando la llave pública y el valor `n` proporcionados en el algoritmo RSA. Convierte cada

carácter en su correspondiente valor ASCII y luego encripta esos valores.

- **Entrada**:
  - Un entero que representa la llave pública.
  - Un entero que representa el valor `n`.
  - Una cadena de texto a encriptar.
- **Salida**: Una lista de enteros que representa la cadena de texto encriptada.

- `desencriptaCadena :: Integer -> Integer -> [Integer] -> String`
  - **Descripción**: Esta función desencripta una lista de enteros encriptados utilizando la llave privada y el valor `n` proporcionados en el algoritmo RSA. Convierte cada entero en su correspondiente carácter ASCII y luego concatena los caracteres para formar la cadena desencriptada.
  - **Entrada**:
    - Un entero que representa la llave privada.
    - Un entero que representa el valor `n`.
    - Una lista de enteros encriptados.
  - **Salida**: La cadena de texto desencriptada.

Main.hs

El código en Main.hs implementa la encriptación y desencriptación de archivos utilizando el algoritmo RSA. A continuación, se explica cada parte del código:

- Importaciones:
  - `System.Environment` y `System.IO`: Importan módulos para trabajar con argumentos de línea de comandos y operaciones de entrada/salida de archivos.
  - `System.Random`: Importa el módulo para generar números aleatorios.
  - `Data.Char (isDigit)`: Importa la función `isDigit` del módulo `Data.Char`, que se utiliza para verificar si un carácter es un dígito.
- `appendLinesToFile :: FilePath -> [String] -> IO ()`: Esta función toma una ruta de archivo y una lista de líneas de texto, y agrega las líneas al final del archivo especificado.
- `processArgs :: [String] -> IO ()`: Esta función toma una lista de argumentos de línea de comandos y los procesa. Si no se proporcionan banderas, se muestra un mensaje. Si se proporciona la bandera `-e` o `--encripta`, se llama a la función `encriptacion` con el nombre de archivo proporcionado como argumento. Si se proporciona la bandera `-d` o `--desencripta`, se llama a la función `desencriptacion` con los argumentos de llave privada, `n` y nombre de archivo proporcionados. En caso contrario, se muestra un mensaje de banderas desconocidas.
- `integerListToString :: [Integer] -> String`: Esta función convierte una lista de enteros en una cadena de texto. Concatena los enteros separados por comas.
- `stringToIntList :: String -> [Integer]`: Esta función convierte una cadena de texto en una lista de enteros. Utiliza la función `wordsWhen` para dividir la cadena en palabras y luego convierte cada palabra en un entero.
- `encriptacion :: String -> IO ()`: Esta función realiza la encriptación de un archivo. Toma el nombre del archivo como argumento. Lee el contenido del archivo y luego solicita al usuario que ingrese un número de semilla para generar las llaves. Utiliza la función `generaLlaves` para generar las llaves y las guarda en un archivo llamado "llaves.txt". Luego, encripta el contenido del archivo utilizando las llaves generadas y guarda el texto encriptado en un archivo llamado "encriptado.txt".
- `desencriptacion :: String -> String -> String -> IO ()`: Esta función realiza la desencriptación de un archivo. Toma el nombre del archivo, la llave privada y el valor `n` como argumentos. Lee el contenido del archivo encriptado, utiliza las llaves y el valor `n` proporcionados para desencriptar el texto y guarda el resultado en un archivo llamado "desencriptado.txt".
- `main :: IO ()`: La función principal del programa. Obtiene los argumentos de línea de comandos utilizando la función `getArgs` y luego llama a la función `processArgs` para procesar los argumentos.

Referencias

- https://www.geeksforgeeks.org/rsa-algorithm-cryptography/
- https://www.youtube.com/watch?v=j2NBya6ADSY -https://es.wikipedia.org/wiki/RSA