Instituto Tecnológico de Costa Rica

Área Académica Ingeniería en Computadores

Curso: Arquitectura de Computadores I

Profesor: Ronald García Fernández

Proyecto Individual: Diseño e Implementación de un sistema de encriptación y desencriptación RSA empleando lenguaje ensamblador

Manual de usuario de la aplicación

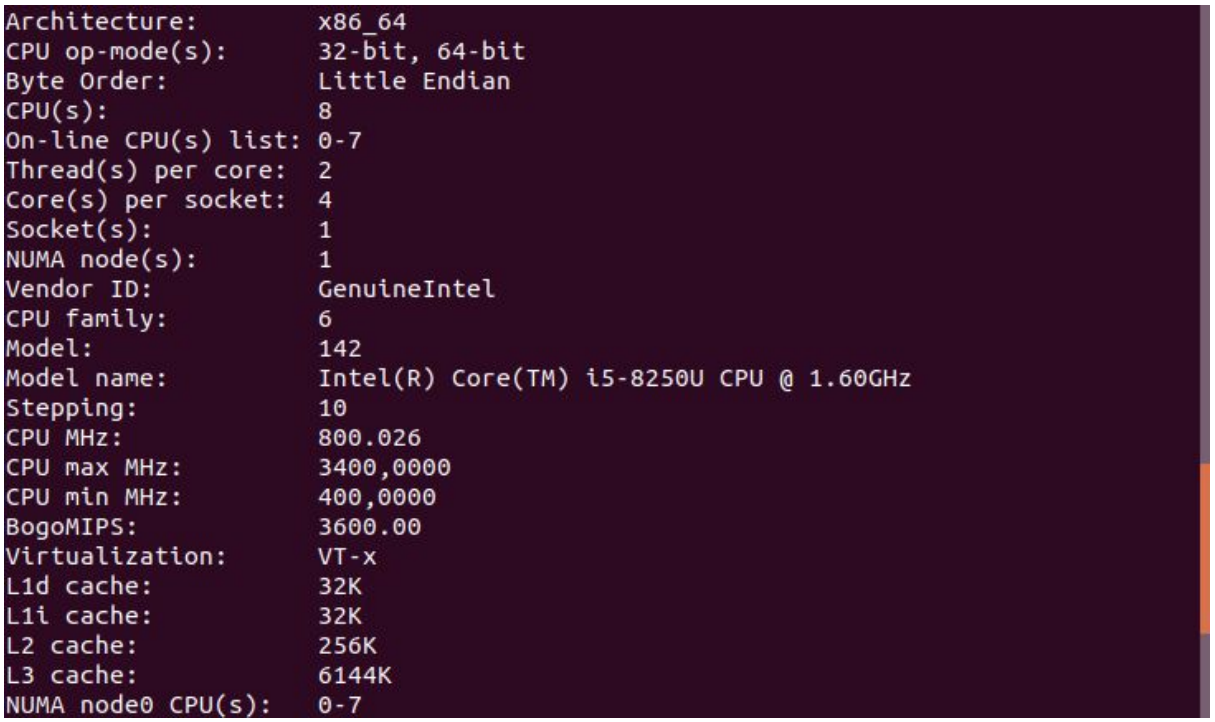
Estudiante: Roberto Sánchez Gutiérrez

I Semestre 2019

1. Índice

2. Configuración ambiente de desarrollo

La aplicación se ha probado en el siguiente ambiente de desarrollo: Sistema operativo Ubuntu 18.04 LTS, con la siguiente información del procesador utilizado al correr el comando “lscpu” en la consola:



Las bibliotecas necesarias para correr la aplicación en consola y ver el desempeño son las siguientes:

Biblioteca	**Instalación**
build-essential	sudo apt install build-essential
nasm	sudo apt install nasm
gdb	sudo apt install gdb

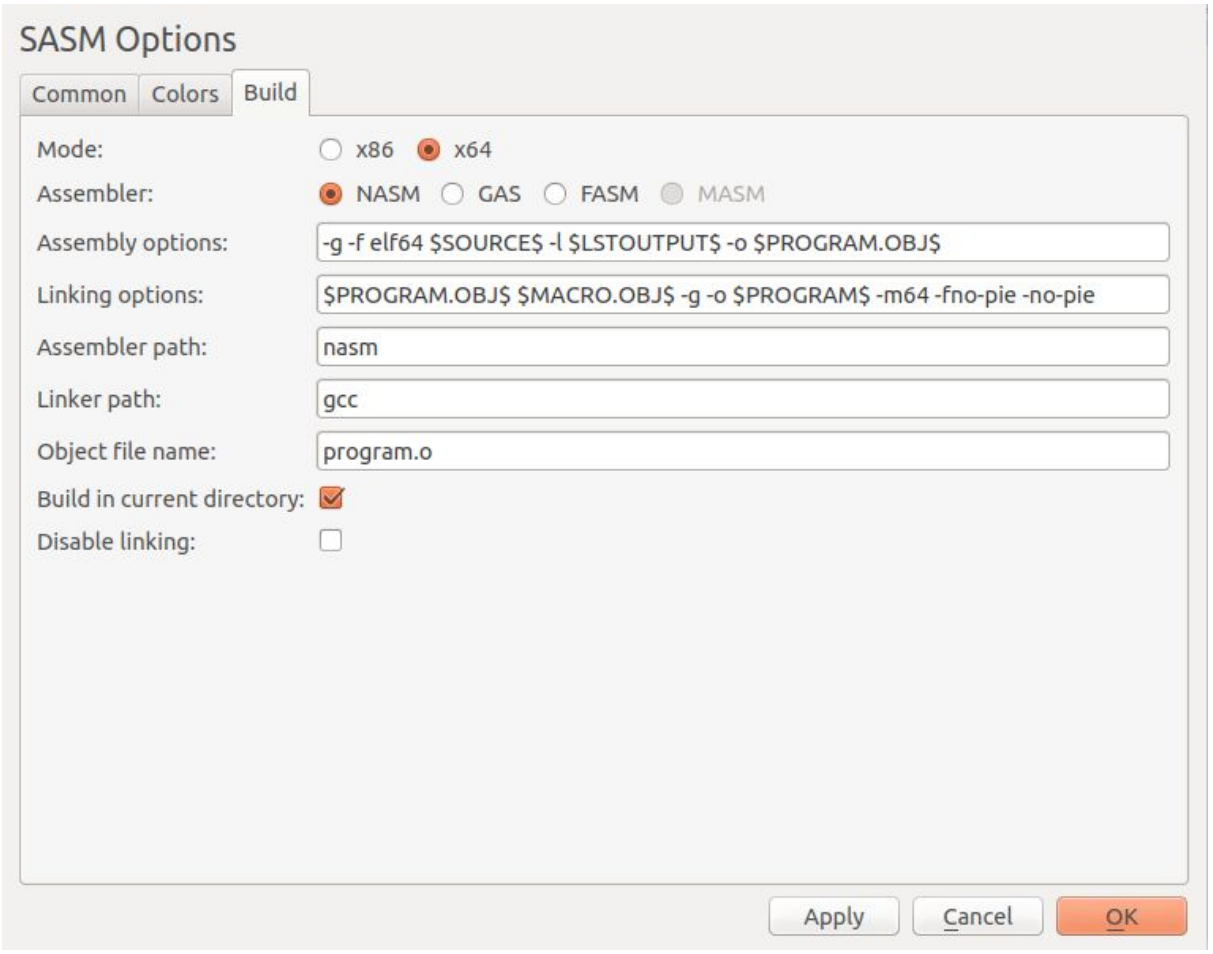
3. Configuración IDE SASM

NOTA: Omita este paso si no desea utilizar ningún IDE y quiere correr la aplicación en consola

Para configurar sasm se deben de seguir los siguientes pasos:

1. Instalar sasm con el siguiente comando:
    a. $sudo apt install sasm
2. Abrir sasm
3. Abrir desde la barra del menú de sasm Settings > Settings

4. En la nueva ventana, escoger build entre las opciones de Common, Colors y Build
5. La siguiente ventana deberá aparecer, y todos sus datos deben quedar idénticos a los mostrados



6. Presionar Apply y luego Ok

4. Archivos de la aplicación

4.1. Mensajes entrada y salida

El funcionamiento general para la encriptación de mensajes es el siguiente:

La carpeta src/messages contiene 3 archivos por defecto:

Nombre archivo	**Contenido**
msg.txt	Mensaje a encriptar por la aplicación
msge.bin	Mensaje encriptado por la aplicación, depende del tamaño de los primos
msgd.txt	Mensaje a desencriptar por la aplicación

IMPORTANTE: Los archivos de salida msge.bin y msgd.txt de la aplicación se cambian cada vez que se corre la aplicación, por lo que si se va a correr la aplicación varias veces se debe de hacer un respaldo de los mismos si se desea.

4.2. Números Primos

Los números primos son el componente principal para la correcta generación de llaves para el algoritmo RSA.
En el caso de la aplicación, los números primos tienen las siguientes características:

- Los números deben tener un tamaño de bits múltiplo de 32 bits, por ejemplo 256, 1024 bits, etc.
- Los números se guardan en los archivos de la carpeta src/keys
- Por defecto, los números primos almacenados son de 256, 512, 1024, y 2048 bits
- Cada archivo de número primo debe tener la extensión .bin. En la carpeta se muestran los números asociados en hexadecimal en un archivo .txt con el mismo nombre.
- Los archivos .bin para cada número primo pueden tener cualquier nombre, esto se explica en la sección de modificación de paths.

4.3. Llaves para el algoritmo

Las llaves para el algoritmo se encuentran guardadas en archivos .bin en la carpeta src/keys, estos archivos son keyn.bin, keye.bin y keyd.bin, los cuales son respectivamente la modulación para las llaves, la llave pública y la privada.
Las mismas tienen de igual forma las siguientes consideraciones:

- Las llaves deben ser del doble de tamaño en bits que los números primos, sino el flujo del programa no será correcto
- Los archivos con las llaves deben tener un tamaño en bytes del doble del de la llave, con un padding al inicio que será ignorado por la aplicación. Si esta restricción no se cumple, no se tendrá un flujo correcto del programa.

IMPORTANTE: Los archivos keyn.bin, keye.bin y keyd.bin se cambian cada vez que se corre la aplicación en el modo generar llaves, por lo que si se va a correr la aplicación para ello se debe de hacer un respaldo de los mismos si se desea.

5. Correr en consola o SASM

Para correr la aplicación se deben de tener las primeras tres bibliotecas mencionadas en el punto de configuración del ambiente de desarrollo, además se deben de tener los paths configurados correctamente como se indica en el punto de modificación de paths.
La aplicación tiene dos maneras de ejecutar, desde consola o desde el IDE de SASM, que se explicaran a continuación
NOTA: Las acciones que se hablan a continuación las cuales se ejecutan con las letras se describen en el punto de acciones disponibles

5.1. Consola

Los pasos para correr desde consola son los siguientes:

- Abrir el archivo src/rsa.asm y asegurarse que el punto de entrada tiene como nombre _start, de la siguiente manera:

```
section .text
global _start
_start:
```

- Ir a la carpeta src y abrir la consola aquí
- Ejecutar el archivo run.sh, de manera que se realicen los siguientes comandos

```
#!/bin/bash
clear
mkdir -p build
nasm -f elf64 -o build/rsa_ob.o -l build/rsa_l.lst rsa.asm -g -F dwarf
ld -s -m elf_x86_64 build/rsa_ob.o -o build/rsa_app -lc --dynamic-linker /lib64/ld-linux-x86-64.so.2
./build/rsa_app
```

- Escoger una acción con la letra y presionando enter y esperar a que termine el algoritmo

Se puede observar que este script hará una nueva carpeta para el build en src

5.2. IDE SASM

Para ejecutar el algoritmo en el IDE de SASM, se deben seguir los siguientes pasos

- Abrir el archivo src/rsa.asm y asegurarse que el punto de entrada tiene como nombre main, de la siguiente manera:

```
section .text
global main
main:
```

- Abrir SASM
- Seleccionar Open project
- Abrir el archivo src/rsa.asm con doble click
- En la ventana de input colocar la acción que se desea realizar
- Presionar F9 o darle en el botón de correr, esperar a que termine de ejecutarse

6. Modificación del path

Para el funcionamiento correcto del algoritmo, se debe cambiar el path absoluto del código fuente que se encuentra en src/data.asm

De manera que la línea 10, la variable path debe cambiarse por la capeta donde se encuentre la carpeta src con el código fuente.

Como se muestra en el siguiente caso:

```
 9  %define word_amount 1+primebits/8
10  %define path '/home/roberto/Documents/GitHub/RSAssembly/ROBERTO_GUTIERREZ_2016134351/'
11  %define keypath 'src/keys/'
```

La línea 10 indica que la carpeta src se encuentra en path, por lo tanto debe cambiarse esta variable para que funcione correctamente antes de compilarse

7. Tamaños

7.1. Primos

Por defecto, la aplicación tiene el tamaño de los números primos como 256 bits para una ejecución rápida, en donde el tamaño de las llaves será entonces de 512 bits.
Para cambiar este tamaño de los primos, se deben de realizar las siguientes acciones:

- Cambiar en src/data.asm la definición de primebits por la cantidad de bits que se quiera, siempre y cuando se un múltiplo de 32 para que funcione correctamente
- Cambiar además en la sección .data, el nombre de los archivos binarios donde se encuentran los números primos, por defecto son prime256_1.bin y prime256_2.bin

7.2. Mensajes

A partir del tamaño de los primos, el tamaño del mensaje a encriptar se determina multiplicando por 2 este tamaño para obtener el tamaño en bits de la llave, y luego dividiéndolo en 8, luego restarle 1 para mayor seguridad y así obtener la cantidad de caracteres que caben en la llave sin que se pierda información.

A continuación se muestra una tabla ejemplo de la cantidad de caracteres que puede contener el archivo del mensaje a encriptar, en caso contrario se muestra un error

Primos (bits)	Llaves (bits)	Caracteres (bytes)
256	512	63
512	1024	127
1024	2048	255
2048	4096	511

8. Acciones disponibles

La aplicación permite por defecto modificar los archivos de los primos y de las llaves correspondientes, como se menciona en archivos de la aplicación, de esta manera solamente se debe de introducir los archivos en binario para cada uno de los casos.

Por aparte, las acciones disponibles en la aplicación en sí, son las siguientes

Tecla	Acción	Archivos cambiados	Salida de la consola

k	Generación de llaves	keyn.bin keye.bin keyd.bin	Asteriscos indicando iteraciones de convergencia
e	Encriptación del mensaje	msge.bin	Asteriscos por cada 32 bits hasta llegar al doble de tamaño de las llaves
d	Desencriptación del mensaje	msgd.txt	Asteriscos por cada 32 bits hasta llegar al doble de tamaño de las llaves
a	Ejecuta todas las acciones anteriores	Todos los archivos anteriores	Todas las anteriores

9. Como modificar las llaves

Para modificar las llaves se deben de cambiar por su correspondiente archivo binario .bin como se menciona anteriormente en las secciones de archivos de la aplicación -> llaves para el algoritmo.

10. Referencias

Dman95.github.io. (2019). *SASM - Simple crossplatform IDE for NASM, MASM, GAS, FASM assembly languages*. [online] Available at: https://dman95.github.io/SASM/english.html [Accessed 21 May 2019].
Manushin, D. (2019). *Dman95/SASM*. [online] GitHub. Available at: https://github.com/Dman95/SASM [Accessed 21 May 2019].
Knuth, D., Knuth, D. and Knuth, D. (2011). *Seminumerical algorithms*. Upper Saddle River, N.J: Addison-Wesley, pp.272-273.

11. Apéndices

A continuación se documentan usos a los archivos adjuntos en src que no son necesarios para utilizar la aplicación pero pueden ser de ayuda para modificar llaves y primos

11.1. Chequeo del funcionamiento con alto nivel usando Python 3

En la carpeta src se encuentra además un código de Python 3 desarrollado para verificar el funcionamiento correcto del algoritmo de ensamblador.

El código solamente carga las llaves utilizadas, el mensaje ingresado, el mensaje encriptado y el mensaje desencriptado. A partir de estos evalúa si todo está correcto. Para correrlo se debe de ejecutar simplemente en consola $python3 check.py

11.2. Uso de Python para convertir .txt a .bin

En el caso que se deseen agregar más números primos, o llaves, en la carpeta src/keys se agrega un archivo de python que permite pasar un hexadecimal de un .txt a un .bin, como se tiene de ejemplo todos los primos en la misma carpeta.
Para convertir primos solamente se debe de insertar el nombre sin extensión con 0 bytes, y para una llave se debe agregar el nombre, agregando la cantidad de bytes de la llave, en este caso los primos entre 4.