TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 4301 — Arquitectura de Computadores I

Proyecto Individual
Diseño e implementación de una aplicación de desencriptación RSA

Fecha de asignación:	24 septiembre 2020	Fecha de entrega:	15 octubre 2020
Grupos:	1 persona	Profesor:	Jason Leitón Jiménez

Mediante el desarrollo de este proyecto, el estudiante aplicará los conceptos de arquitectura de computadores en el diseño e implementación de una aplicación capaz de realizar *image sharpening* y *oversharpening* empleando un ISA específico. Atributos relacionados: **A**nálisis de **P**roblemas (AP), el cual se encuentra en **A**vanzado (A).

1. Descripción General

El RSA (Rivest-Shamir-Adleman) es un sistema criptográfico de clave pública desarrollado en 1979. Es asimétrico, lo que significa que cuenta con una llave pública (Public Key) y una llave privada (Private Key). El funcionamiento se muestra a continuación:

1. Un cliente (e.g., buscador) envía su llave pública a un servidor y solicita información. El cliente también tiene una llave privada que solo tiene él (esta se debe guardar celosamente).
2. El servidor encripta la información usando la llave pública y se la envía al cliente.
3. El cliente recibe la información y la desencripta usando la llave privada.

El funcionamiento a nivel de esquema de las llaves públicas y privadas se muestra en la Figura 1. Como se observa, cualquier ente que tenga acceso a la llave pública puede encriptar un mensaje, pero no cualquiera puede desencriptarlo pues necesita la llave privada. En este vídeo se encuentra una explicación interactiva del algoritmo RSA.

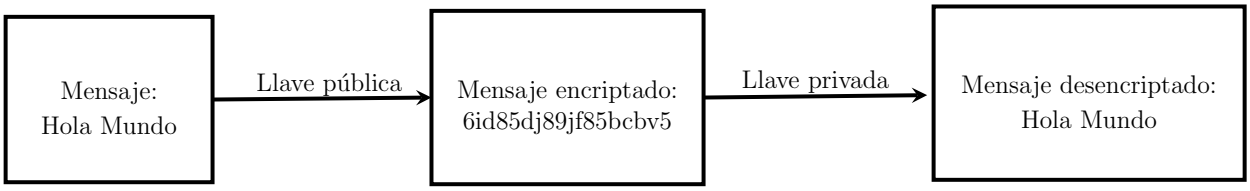

Figura 1: Esquema de encripción y desencripción

Para encriptar un mensaje se tiene que realizar la operación

$$c = m^e \mod n, \tag{1}$$

donde m es el mensaje, c es el mensaje encriptado y (e y n) son los parámetros de la llave pública.

Para desencriptar el mensaje se tiene que realizar la operación

$$m = c^d \mod n, \tag{2}$$

donde c es el mensaje encriptado, m es el mensaje desencriptado o recuperado y (n[1] y d) son los parámetros de la llave privada.

Para la generación de llaves se usa el siguiente algoritmo:

1. Seleccione dos números primos aleatorios (p y q). Por ejemplo 7 y 2.
2. Calcule $n = pq$. Este es el módulo de la llave pública y privada. Por ejemplo, en este caso $n = 7 \times 2 = 14$.
3. Calcule la función de Euler $\phi = (p-1)(q-1)$. Por ejemplo, en este caso $\phi = (7-1)(2-1) = 6$.
4. Se selecciona un entero positivo e menor que ϕ (no puede ser igual), este debe ser coprimo a ϕ. Este es el exponente de la llave pública. Un número es coprimo de otro si el máximo común divisor es igual a 1. Dos números coprimos no tienen que ser primos, por ejemplo 6 y 19 son coprimos (19 es primo, 6 no es primo). Si e es muy pequeño podría suponer una vulnerabilidad importante para el sistema. Por ejemplo, en este caso $e = 5 < \phi = 6$.
5. Se selecciona un entero positivo d que satisfaga: $de \mod \phi = 1$. Por ejemplo, en este caso $de = 11 \times 5 = 55 \rightarrow 55 \mod 6 = 1$. Entonces $d = 11$ cumple el requerimiento.

De esta manera se genera el siguiente par de llaves:

- Llave pública o de encripción: (5,14).
- Llave privada o de desencripción: (11,14).

Se pueden encriptar todos los números mayores a 1 [2]. Por ejemplo se encriptará el número 3 con las llaves públicas y privadas expresadas anteriormente

$$c = m^e \mod n = 3^5 \mod 14 = 243 \mod 14 = 5.$$

Para desencriptar se muestra la operación

$$m = c^d \mod n = 5^{11} \mod 14 = 48828125 \mod 14 = 3,$$

[1] note que este parámetro n es el mismo que el de la llave pública.
[2] dado (1) y (2): $1^x = 1 \forall x \in R$

se verifica como se obtiene $m = 3$ tanto en el mensaje original como en la desencripción. También resulta bastante evidente que como la cantidad de números a encriptar es relativo al módulo.

Es evidente como la dificultad del algoritmo RSA radica en el manejo de números extensos esto se hace evidente viendo las operaciones involucradas en (1) y (2). No es un secreto que entre más extensos sean los números más difícil es encontrar la combinación llave pública y privada. Por esta razón se usan llaves públicas y privadas de 1024 bits ($\approx 1{,}8 \times 10^{308}$ combinaciones, necesitando 309 dígitos en base decimal), 2048 bits ($\approx 3{,}2 \times 10^{616}$ combinaciones, necesitando 617 dígitos en base decimal) o incluso más. En este enlace hay un generador de llaves RSA.

Se vuelve bastante claro como no se pueden procesar las operaciones exponente y modular de forma tradicional, sino el sistema requeriría mucha memoria o sería muy lento. Por esta razón se usa el proceso de exponenciación modular. En el vídeo 1 y vídeo 2 se explica este proceso matemático.

2. Especificación

Se le solicita desarrollar **en ensamblador** (asm) un programa que desencripte información en RSA. Este programa recibe solamente la llave privada (puede ser introducida en consola o en un archivo llamado **llaves.txt**). El programa debe realizar la siguiente funcionalidad:

1. **Imagen:** El usuario le indica al programa la dirección de la imagen a desencriptar. Realiza el proceso de desencriptación y le muestra al usuario en una cuadrícula 1×2 la imagen de entrada (encriptada) y la procesada (desencriptada).

Este proceso **NO** se realizará completamente en ASM, se realizará bajo los siguientes requisitos generales de funcionalidad:

1. La imágenes deben ser visualizadas en algún software de alto nivel (e.g., Python, Matlab, Octave, etc), solo la visualización. El **procesamiento** de la imagen es **completamente en ensamblador (asm)**.

2. Una imagen encriptada de 640×480 con representación de 2 bytes cada píxel[3] con una llave pública. El formato de la imagen es el siguiente:

   - Imagen en escala de grises con píxeles con valores entre [0, 255].
   - La representación de cada píxel está dada por 2 bytes sucesivos.
   - La imagen encriptada se observa en el visualizador de imágenes como una de 640×960.

[3]la imagen original es de 8 bits, pero como el n de la llave pública y privada determina el rango de números a encriptar, se procuró un $256 < n < 65536$ o n representable con a lo sumo 16 bits, esto también para guardar el alineamiento de la memoria

- **Importante: ¿es una equivocación del profesor que diga que la imagen es de 640×480 y luego dice 640×960?** No, recuerde que se encriptó una imagen de 640×480, como se indicó anteriormente el resultado de la encriptación siempre genera un dato más grande, por esta razón un píxel original de un byte se convierte en dos bytes (doblando el tamaño de la imagen). El píxel 0 (MSB) y 1 (LSB) serán el píxel encriptado 0, luego el 2 (MSB) y el 3 (LSB) serán el píxel encriptado 1.

3. Se debe implementar una optimización de código, sino durará mucho tiempo desencriptando 307200 datos [4]. **Pista:** El usuario sabe de antemano que se encriptó un valor de 8 bits se sabe que habrá a lo sumo 255 combinaciones [5], aumentando el rendimiento 1204 veces.

4. El simulador del ISA (e.g., ARM, x86, RISC-V, otros) y otras herramientas de desarrollo podrán ser elegida abiertamente por cada estudiante. Todo el procesamiento de la imagen debe ser en este simulador seleccionado, recuerde que el lenguaje de alto nivel es solo para visualización, si usa lenguaje de alto nivel en procesamiento obtendrá nota de cero.

5. La imagen que se le da a la/el estudiante es en formato 'crudo', es decir como si fuese un txt con los datos en formato decimal de un byte cada uno.

6. El programa automáticamente debe generar un archivos de salida (imagen desencriptada).

7. Por facilidad todo este programa debe ser integrado en un solo *framework*. Que tome la imagen en formato 'crudo', procese la imagen en ASM y la visualice al usuario en un arreglo 1×2.

El profesor le provee un archivo para que realice pruebas con una imagen encriptada en enlace. Está conformado por los siguientes archivos:

- Un archivo de texto plano con la imagen encriptada en formato 'crudo' en una sola dimensión. El formato de la defensa será así.

- Los parámetros de llave pública y privada, esto NO es una sugerencia del archivo **llaves.txt** pues ese debe tener solo la llave privada. Los valores de llave privada en la defensa no serán superiores a 65536 [6].

Se sugiere revisar el funcionamiento

[4] $640 \times 480 = 307200$
[5] $2^8 = 256 - 1 = 255$, no se usa el cero
[6] $2^{16} = 65536$

3. Evaluación y entregables

La evaluación del proyecto se da bajos los siguientes rubros:

- Presentación proyecto 100 % funcional (65 %): Una defensa de 30 minutos donde el profesor evaluará el rendimiento del algoritmo. El/la estudiante recibirá minutos antes o en la misma defensa un archivo en formato 'crudo' con la imagen encriptada y con los parámetros de llave privada. El profesor evaluará las pruebas según rúbrica correspondiente los requerimientos ya indicados.

- Articulo científico tipo *paper* (17.5 %): El paper a realizar deberá tener una extensión no mayor a 4 páginas completas (incluyendo bibliografías), deberá ser realizado con LaTeX, siguiendo un formato establecido (IEEE Transactions o ACM, por ejemplo). Se les provee un ejemplo de paper en el enlace. En general el *paper* deberá contar con las siguientes secciones:

  1. Abstract (en inglés): Un buen abstract tiene las siguientes características:
     a) Un abstract permite a los lectores obtener la esencia o esencia de su artículo o artículo rápidamente, para decidir si leer el artículo completo.
     b) Un abstract prepara a los lectores para seguir la información detallada, los análisis y los argumentos en su artículo completo.
     c) Un abstract ayuda a los lectores a recordar puntos clave de su paper.
     d) Un abstract es de entre 150 y 250 palabras.
  2. Palabras clave significativas (a lo sumo 6).
  3. Introducción: Una buena introducción muestra el contexto del problema o lo que se va a solucionar, introduce el tema al lector. Al final de la introducción se indica la organización del documento (primero se muestra el algoritmo, luego....).
  4. Algoritmo desarrollado.
  5. Resultados.
  6. Conclusiones escritas en prosa.
  7. Bibliografía, en formato IEEE y referenciadas en el texto (usar cite). Referencia bien para evitar problemas de plagio. Una documento no referenciado en el texto no existe.

- Documentación de diseño (17.5 %): Este documento se encuentra directamente ligado con el atributo AP. La documentación del diseño deberá contener las siguientes secciones:

  1. Listado de requerimientos del sistema: Cada estudiante deberá determinar los requerimientos de ingeniería del problema planteado, considerando partes involucradas, estado del arte, estándares, normas, entre otros.

2. Elaboración de opciones de solución al problema: Para el problema planteado deberán documentarse al menos dos opciones de solución. Cada solución deberá ser acompañada de algún tipo de diagrama.
3. Comparación de opciones de solución: Se deberán comparar explícitamente las opciones de solución, de acuerdo con los requerimientos y otros aspectos aplicables de salud, seguridad, ambientales, económicos, culturales, sociales y de estándares.
4. Selección de la propuesta final: Se deberá evaluar de forma objetiva, válida y precisa las soluciones planteadas al problema y escoger una solución final.
5. Archivo tipo README donde especifiquen las herramientas que usaron. **Es un documento README.MD aparte.**

- Los documentos serán sometidos a control de plagios para eliminar cualquier intento de plagio con trabajos de semestres anteriores, actual o copias textuales, tendrán nota de cero los datos detectados. Se prohibe el uso de referencias hacia sitios no confiables.