Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Analisis, diseño y fabricación de Sistemas
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Hoja de trabajo #2

Fecha de entrega: 4 de Marzo, 2018 - 11:59pm

Instrucciones: Realizar cada uno de los ejercicios siguiendo sus respectivas instrucciones. El trabajo debe ser entregado a traves de Github, en su repositorio del curso, colocado en una carpeta llamada "Hoja de trabajo 2". Al menos que la pregunta indique diferente, todas las respuestas a preguntas escritas deben presentarse en un documento formato pdf, el cual haya sido generado mediante Latex. Los ejercicios de programación deben ser colocados en una carpeta llamada "Programas", la cual debe colocarse dentro de la carpeta correspondiente a esta hoja de trabajo.
"

Parte #1: Codificación y Decodificación

Para esta sección, debe escribir dos funciones: "Codificar" y "Decodificar". Las funciones deben aceptar como parametro un numero n y un `string`. El numero corresponde a la base que se esta utilizando y el string el mensaje que se intenta codificar o decodificar. La función "Codificar" debe transformar el `string` a un arreglo de numeros $n \in \mathbf{Z}^n$. De la misma forma, la función "Decodificar" debe transformar el arreglo de numeros al `string` original. Debe considerar que el numero n debe ser de precision arbitraria, de tal forma que utilizar un `int` no da a basto.

Parte #2: RSA

Crear una función llamada "RSAKeyGen" que acepta como parametro un numero l que corresponde a la longitud de la llave publica. Considerar que la llave privada debe ser aproximadamente de la misma longitud. Es importante utilizar un RNG (Random Number Generator) criptografico para que las llaves sean seguras. Esta función debe retornar 3 valores: La llave publica e, la llave privada d y el modulo o base n tal que permite encriptar y codificar mensajes en el espacio $\mathbf{Z}^n$.

Parte #3: Encriptar y Desencriptar

Crear dos funciones: "Encriptar" y "DesEncriptar". La función "Encriptar" acepta una llave publica e, el modulo n y un `string`. Esta función debe producir un mensaje encriptado y codificado utilizando RSA. De la misma forma, la función *DesEncriptar* acepta la llave privada d, el modulo n y el mensaje ecriptado (un array de numeros) y recupera el `string` original que fue encriptado.

Parte #4: Hackear

Crear una función llamada "Hackear". Esta función acepta una llave publica e, un modulo n, un tiempo t (en mili-segundos) y un mensaje encriptado (array de numeros) e intenta recuperar el mensaje (`string`) original utilizando fuerza bruta. Si la función no logra recuperar el mensaje en el tiempo dado en el parametro t, se debe generar un error.

Parte #5: Test de seguridad

Escribir una prueba unitaria llamada "TestSeguridadLineal". Esta prueba unitaria generara llaves RSA de longitud incremental empezando con una longitudo de 5. Luego debe encriptar un mensaje e intentar vencer la ecriptación mediante la función "Hackear". El tiempo limite para vencer la encriptación es la longitud de la llave publica en mili-segundos. Hasta que longitud de llave fue posible vencer la encriptación?

De la misma forma, escribir una prueba unitaria llamada "TestSeguridadPolinomial". Esta prueba es igual a la prueba "TestSeguridadLineal" excepto que el tiempo limite es la longitud l de la llave publica elevado a la 3ra potencia (l^3). Hasta que longitud de llave fue possible vencer la encriptación? Fue un incremento considerable comparado con el "TestSeguridadLineal"?