Presentación

En esta práctica estudiaremos el algoritmo RSA y las funciones de hash. En primer lugar, implementaremos las funciones que nos permitirán encriptar, desencriptar, firmar y validar firmas con el algoritmo de clave pública RSA. Los detalles de los algoritmos se pueden volver al módulo 5: "Firmas digitales". Posteriormente, utilizaremos la función hash SHA-256 para ver qué dificultad tiene la obtención de mensajes diferentes que generan valores hash iguales. Por último, realizaremos un ataque contra las firmas digitales RSA.

Objetivos

Los objetivos de esta práctica son:

1. Comprender cómo funciona el algoritmo de RSA y cómo su seguridad se basa en la dificultad de resolver el problema basado en la factorización de enteros
2. Entender las propiedades de las funciones hash y los ataques que pueden realizarse sobre ellas.

Descripción de la práctica a realizar

1. El algoritmo de firma RSA (7 puntos))

En este ejercicio implementaremos el algoritmo RSA. Para ello puede utilizar la librería Sympy, que le proporcionará funciones útiles como isprime o mod inverse. Asimismo, tenga en cuenta que la función pow() de Python, permite realizar operaciones modulares proveyendo el módulo como tercer parámetro.

1. Función que implementa la generación de un par de claves RSA. (**3 puntos**)

   Esta función implementará el funcionamiento de la generación de claves RSA tal y como se explica en el módulo didáctico 5 de los materiales de la asignatura.

   La función tomará como variable de entrada la longitud de la clave que se desea generar, `length`; y devolverá la clave pública y privada en formato `(e,n)` y `(d,n)`. Para ello será necesario generar dos valores primeros aleatorios 'p' y 'q' que cumplan el teorema de Euler y un valor 'e' que cumpla la condición del mínimo común divisor (mcd).

   - La variable `p` será un valor primero incluido dentro de la longitud indicada.
   - La variable `q` debe ser del mismo estilo.
   - la variable `e` debe cumplir que el mínimo común divisor del resultado del teorema de Euler y ella mismo sea '1' (es decir, que ambos sean primeros).
   - la función devolverá las claves pública y privada en formato ((e, n), (d, n))

2. Función que implementa el cifrado de un mensaje con RSA. (**1 punto**)

   Esta función recibe como parámetros el mensaje a encriptar y la clave pública generada anteriormente. La función devuelve en mensaje encriptado utilizando el algoritmo RSA.

3. Función que implementa el desencriptado de un mensaje con RSA. (**1 punto**)

   Esta función inversa recibe como parámetros el mensaje encriptado y la clave privda generada en el primer punto. El objetivo te puede desencriptar el mensaje encriptado y conseguir el original.

4. Función que implementa la firma digital con RSA. (**1 punto**)

   Esta función implementará la firma RSA tal y como se explica en el módulo didáctico 5 de los materiales de la asignatura. La función tomará como variables dos parámetros: Clave privada previamente generada (Kpriv) y mensaje (m); y devolverlo al mensaje firmado.

   - La variable Kpriv contendrá la clave privada en formato [d, n]
   - La variable `m` contendrá el mensaje original a firmar.
   - La función devolverá el nuevo estado.

5. Función que implementa la validación de la firma RSA. (**1 punto**)

   La función recibirá como argumentos una clave pública, el mensaje a validar (que al igual que en la función anterior, será un valor numérico) y el valor de la firma.

   La función tomará como variables de entrada dos parámetros: `state` y `inverse`; y devolverá la variable `state` con el nuevo estado.

   - La variable `Kpub` contendrá la clave pública en formato [e, n].
   - La variable `message` contendrá un número que representa el mensaje del que se quiere validar la firma.
   - La variable `signature` contendrá la firma a validar.
   - La función True o False, indicando si la firma se ha validado correctamente o no.

2. Análisis de funciones hash (3 puntos)

1. Función que implementa la generación de la clave. (**1,5 puntos**) En este ejercicio utilizaremos la función hash SHA-256 para analizar la importancia de la seguridad de las funciones hash. Para ello utilizaremos la implementación del algoritmo SHA-256 de la librería hashlib de Python.

   - La variable message contendrá un número entero o el mensaje de texto en ASCII del que calcularemos el hash.
   - Creación de un objeto SHA-256
   - generación hash en formato hexadecimal

2. Función que encuentra segundas preimágenes para la función hash. (**1,5 puntos**)

   La función recibirá como argumentos un mensaje y el número de bits

   La función tomará como variables de entrada el parámetro: `message`; y devolverá una variable de confirmación

- La variable message contendrá el mensaje del que queremos encontrar una preimagen de su hash.
- Se deben programar un número de iteraciones del rango de '1000000'
- La función devolverá un mensaje diferente indicando si ha encontrado preimagen o no.

Criterios de valoración

La puntuación de cada ejercicio se encuentra detallada en el enunciado.

Sólo se pueden utilizar las librerías de Python incluidas en el Skeleton con un importe
tt.

Por otra parte, cabe remarcar que el código que se entregue de la práctica debe contener los comentarios necesarios para poder seguirlo y entenderlo. En caso de que el código no incluya comentarios, la corrección de la práctica se realizará únicamente de forma automática y no se proporcionará una corrección detallada. La no inclusión de comentarios puede ser también motivo de reducción de la nota.

Se pueden utilizar herramientas de IA generativa pero sólo como soporte a la generación de ideas y ampliación del conocimiento, no como reemplazo para el pensamiento crítico y la creación original. Recuerde revisar y testear el código generado e introducir sus propias modificaciones. Es importante destacar que puedes colaborar con tus compañeros e incluso realizar la actividad por parejas o en grupo de trabajo, pero la práctica deberá ser original de cada alumno.

Formato y fecha de entrega

La fecha máxima de entrega de la práctica es el **08/06/2025** (a las 23:59 horas).

Junto con el enunciado de la práctica encontrará el esqueleto de la misma (fichero con extensión .py). Este archivo contiene las cabeceras de las funciones que debe implementar para resolver la práctica. Este mismo archivo es el que debe entregar una vez que codifique todas las funciones. No se debe entregar el archivo comprimido. No se aceptarán archivos en otros formatos que no sean .py.

Adicionalmente, también le proporcionaremos un archivo con tests unitarios para cada una de las funciones que debe implementar. Puede utilizar estos test para comprobar que su implementación gestiona correctamente los casos principales, así como para obtener más ejemplos concretos de lo que se espera que devuelvan las funciones (más allá de los que ya se proporcionan en este enunciado). Note, sin embargo, que las macetas no son exhaustivas (no se prueban todas las entradas posibles de las funciones). Recuerde que no se puede modificar ninguna parte del archivo de test de la práctica.

La entrega de la práctica constará de un único archivo Python (extensión .py) donde haya incluido su implementación.