Tarea 4

Criptografía y seguridad 2020-2

Fecha límite de entrega: 29 de abril

Indicaciones

- Resuelve los ejercicios 1 y 2, con valor de 2.5 puntos cada uno. Elige entre los demás ejercicios para sumar un total de 10 puntos.
- Puede hacerse de forma individual o en pareja, en caso de hacerla en pareja basta que se entregue una solución.
- Sube tu tarea solo cuando estés completamente seguro de que es correcta, ya que solo se puede subir una vez.
- Escribe los cálculos que realizaste, o en caso de haber usado otra herramienta (como un programa) indícalo en tu respuesta.
- Organiza tus archivos en un archivo `.zip`, incluyendo los datos de los alumnos, y súbelo en `https://forms.gle/5YsnLNv6BULXM4no8`

Ejercicios

1. Sea $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ una función de cifrado por bloques (una permutación pseudoaleatoria). Considera las siguientes formas de usar F para cifrar un bloque $m \in \{0,1\}^n$ con la clave k:

   - $c = F_k(m)$.
   - Se elige $r \leftarrow \{0,1\}^n$ y el cifrado es $c = (r, F_k(r) \oplus m)$.

   ¿Cuál de las dos formas consideras que es más segura? ¿Por qué? ¿Preferirías una de estas formas o algún modo de operación de los vistos en clase?

2. Es posible que cuando se manda un criptotexto $c = c_1, c_2, c_3 \ldots$ en la transmisión se pierda el bloque c_2, así que se recibe el mensaje $c' = c_1, c_3, c_4, \ldots$ Describe el efecto que causa un bloque perdido en el criptotexto, cuando se descifra usando los modos de operación CBC, OFB y CTR.

3. (2.5 pts.) Existen varias formas de usar un cifrador de bloques para construir funciones hash.

   a) Describe como hacer una función hash usando el método de Davies-Meyer.

   b) Recuerda que en DES existen llaves k para las que es fácil encontrar m tal que $DES_k(m) = m$. Muestra cómo usar esta propiedad para encontrar una colisión en la construcción de Davies-Meyer aplicada a DES.

4. (2.5 pts.) El cifrado de AES consiste en aplicar las siguientes transformaciones

$$\begin{aligned} &\mathsf{AddRoundKey} \to \mathsf{SubBytes} \to \mathsf{ShiftRows} \to \mathsf{MixColumns} \\ \to &\mathsf{AddRoundKey} \to \mathsf{SubBytes} \to \mathsf{ShiftRows} \to \cdots \end{aligned}$$

   mientras que en el proceso de descifrado se aplican las transformaciones

$$\begin{aligned} &\mathsf{AddRoundKey} \to \mathsf{ShiftRows}^{-1} \to \mathsf{SubBytes}^{-1} \\ \to &\mathsf{AddRoundKey} \to \mathsf{MixColumns}^{-1} \to \mathsf{ShiftRows}^{-1} \to \cdots \end{aligned}$$

   a) Explica por qué en el descifrado puede intercambiarse el orden de las transformaciones $\mathsf{ShiftRows}^{-1}$ y $\mathsf{SubBytes}^{-1}$.

b) Tanto MixColumns como su inversa $\mathsf{MixColumns}^{-1}$ son transformaciones lineales sobre el bloque de bytes. Explica cómo puedes intercambiar el orden de $\mathsf{MixColumns}^{-1}$ y AddRoundKey en el proceso de descifrado. (Hay que hacerle una modificación a la clave de ronda.)

c) Usando los incisos anteriores, muestra que se puede realizar el descifrado de AES mediante la sucesión

$$\begin{aligned}&\mathsf{AddRoundKey} \to \mathsf{SubBytes}^{-1} \to \mathsf{ShiftRows}^{-1} \to \mathsf{MixColumns}^{-1}\\ \to&\mathsf{AddRoundKey} \to \mathsf{SubBytes}^{-1} \to \mathsf{ShiftRows}^{-1} \to \cdots\end{aligned}$$

Nota que el orden de estas transformaciones es el mismo que el de las transformaciones correspondientes en el proceso de cifrado.

5. (2.5 pts) Extrae el archivo `mis_archivos.zip`, que contiene el directorio `Mis archivos`. Desde este directorio ejecuta `juego.py` con Python 3:

```
Mis archivos$ python3 juego.py
```

Este programa es un ransomware de juguete. Haz un programa que funcione como «vacuna» para el ransomware, es decir, que revierta los cambios hechos por `juego.py`. Cuando se ejecute tu programa en un directorio donde anteriormente se ejecutó el ransomware, se recuperarán (exactamente) los archivos originales.

6. (5 pts.) En este ejercicio programarás el ataque del oráculo de padding, usando AES en modo CBC y una clave de 16 bytes.

a) Implementa el oráculo de padding, es decir, una función llamada padding_correcto que recibe un criptotexto (con el IV incluido al inicio) y devuelve un valor booleano: verdadero cuando el mensaje descifrado tiene un relleno válido y falso en caso contrario. Esta función usará AES en modo CBC con la llave global k para descifrar el criptotexto, usando el IV incluido en el criptotexto. Nota que k no es parte de la entrada, pues la idea es simular una caja negra que solamente responde si el descifrado tuvo un padding correcto.

b) Haz una función llamada recupera_padding que recibe una cadena C de $16(n+1)$ bytes, que equivale a un bloque IV y n bloques de datos, y suponemos que $M = AES_k^{-1}(C)$ contiene un padding de entre 1 y 16 bytes. La función devolverá la longitud de dicho padding. Esta función no puede usar ningún método de AES ni la llave k, solamente el oráculo padding_correcto.

c) Implementa una función llamada recupera_mensaje_original que recibe una cadena C de $16(n+1)$ bytes, que equivale a un bloque IV y n bloques de datos. La función devuelve $AES_k^{-1}(C)$ usando el modo CBC, es decir, recupera el mensaje claro que corresponde al mensaje cifrado C. Esta función no puede usar ningún método de AES ni la llave k, solamente el oráculo padding_correcto y opcionalmente la función recupera_padding.

d) Para probarlo, tu programa se deberá poder ejecutar de la siguiente forma

```
$ python attack.py archivo_llave archivo_mensaje
```

donde `archivo_llave` es un archivo que contiene exactamente los 16 bytes de la llave k, y `archivo_mensaje` contiene un mensaje cifrado cuyos primeros 16 bytes son el valor IV con el que fue cifrado. El programa guardará la salida de recupera_mensaje_original en un archivo llamado `mensaje_descifrado`.