Criptografía y Seguridad
Tarea 4

Ailyn Rebollar Pérez Ángela Janín Ángeles Martínez

Ejercicios

1. Sea $F : K \times \{0,1\}^n \to \{0,1\}^n$ 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 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 éstas formas o algún modo de operación de los vistos en clase?

   Solución:
   Consideramos la segunda forma más segura porque el cifrado de cada bloque es una tupla donde tenemos una cadena aleatoria diferente y a ella se le aplica la función F además de un $\oplus$, entonces esto impide que aunque alguien logre obtener la primera cadena aleatoria, no garantiza que pueda obtener la segunda, ni la tercera, etc.

   Y para elegir el modo de operación depende de si podemos asegurar de que no se perderá algún bloque porque si se llega a perder uno, como en CBC pues probablemente no se pueda descifrar el texto claro completo pero sino no se puede asegurar, elegiríamos el modo de la tupla pues los bloques de mensajes no dependen de la salida del cifrado del bloque anterior porque el $\oplus$ que se aplica depende de la cadena aleatoria que se va generando.

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

   - **CBC**: En CBC, cada bloque de texto plano se le aplica XOR con el bloque cifrado anterior. Esto implica que, si perdemos un bloque, en realidad perdemos 2 (el primer bloque que se perdió y el siguiente a éste). ¿Por qué? Pues, el bloque perdido no será recuperable, además de que el bloque siguiente a este será incorrectamente decifrado (pues no pudo hacer XOR con el bloque anterior).
   - **OFB**: Actúa como cifrado de flujo, por lo tanto, al perder un bloque, únicamnete se perdería el texto plano de ese bloque. Esto se debe a que, en estos modos, cada bloque de texto cifrado solo afecta el bloque correspondiente de texto sin formato.
   - **CTR**: Actúa como cifrado de flujo, CTR convierte una unidad de cifrado por bloques en una unidad de flujo de cifrado. Esto se debe a que, en estos modos, cada bloque de texto cifrado solo afecta el bloque correspondiente de texto sin formato.

3. 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.**

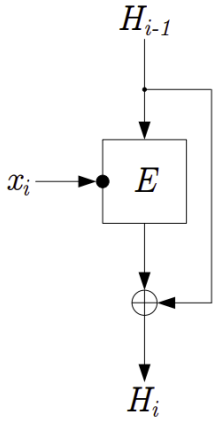
Se trata de una función hash basada en un cifrado de bloque, donde la longitud en bits del resultado de hash es igual a la longitud del bloque del cifrado de bloque.

La función de compresión de un single-block-lenght alimenta cada bloque del mensaje m_i como la llave para un cifrado de bloque. Eso alimenta el valor de hash previo H_{i-1} como texto plano que se cifrará. El texto cifrado es también XOR-ed con el valor hash anterior (H_{i-1}) para prudcir el nuevo valor hash H_j. En la primera ronda, donde no existe un valor hash previo, se usa una constante pre-especificada, un valor inicial H_0.

$$H_i = E_{xi}(H_{i-1}) \oplus H_{i-1}$$

(b) **Recuerda que 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.**

Considerando la debilidad en la llave de DES, donde $DES_k(m) = m$, entonces consideremos el esquema de Davis Meyer:



Donde E es el cifrado DES y la otra entrada es el bloque de texto a cifrar, entonces sabemos que el resultado, es bloque de texto a cifrar perse. Hacemos XOR. Si ciframos otro mensaje, obtenemos por medio de DES el mismo mensaje perse, entonces, notamos que al hacer XOR de dos cosas iguales, obtenemos 0. Por lo tanto, para dos mensaje iguales tendríamos el mismo hasheo.

4. 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.

Solución:
En el archivo vacuna.py se implementa el programa que actua como vacuna para el ransomware el cual recupera los archivos. Basta con ejecutar en la terminal en la carpeta donde estén los archivos:

python3 vacuna.py