Tarea 2

Emilio Junoy de Juambelz

3 de abril de 2025

1. Para un alfabeto de 26 letras, ¿Cuántas matrices de 2x2 hay que nos permitan cifrar mediante Hill? Justifica tu repuesta.
   Para que una matriz de 2x2 pueda ser utilizada para cifrar con el método de Hill, se necesita que ésta sea invertible módulo el tamaño del alfabeto, es decir 26. Para que una matriz sea invertible módulo 26 necesitamos que su determinante sea un primo relativo a 26. Podemos hacer un pequeño programa de Python que implemente el algoritmo de Euclides para calcular el máximo común divisor entre los posibles determinantes de las matrices de 2x2 y el 26, y así determinar el número de matrices de 2x2 invertibles módulo 26.
   Realizando esto, vemos que hay un total de 163535 matrices que podemos usar para cifrar con el método de Hill, sin embargo, una de estas matrices es la identidad, asì que tenemos 163535 matrices para cifrar no trivialmente con el método de Hill.

2. Investiga un caso de la vida real donde se rompió la seguridad con máximas de Kerckoffs.
   Un caso muy importante en la historia es el de la máquina Enigma. Recordemos que una de las máximas de Kerckoffs es que la seguridad no debe residir en la obsuridad del método, sino en la llave utilizada. Esta máxima fue rota por los Alemanes en la segunda guerra mundial. Las claves en la máquina Enigma eran las posiciones en las que se colocaban los rotores, pero los Alemanes frecuentaban usar la misma clave o usar claves con patrones predecibles, lo que llevó a que los Aliados lograran descifrar el código Enigma. Así, las llaves que elegimos para cifrar son fundamentales para la seguridad del método, pues quebrantar esta máxima de Kerckoffs puede tener consecuencias irremediables.

3. Usando el polinomio x^4+x+1, da la lista de bits de salida de un LFSR asociado usando como semilla una secuencia de ceros, y cualquier otra que escojas.
   Si empezamos el LFSR con una secuencia de ceros, obtenemos la sigueinte secuencia de bits: $0000, 1000, 0100, 1010, 0101, 0010, 1001,$ $1100, 0110, 1011, 1101, 1110, 0111, 0011, 0001$ De modo que los bits de salida son 000010100110111.
   Si empezamos con la secuencia 1010 obtenemos la sucesión $1010, 0101,$ $0010, 1001, 1100, 0110, 1011, 1101, 1110, 0111, 0011, 0001, 0000, 1000,$ 0100. Así que los bits de salida son 010100110111000.

4. Supón que se manda un criptotexto $c = c_1, c_2, \ldots$, pero se pierde el bloque c_2, así que se recibe $c = c_1, c_3, \ldots$. Al descifrar el mensaje, ¿Cuál es el efecto de un bloque no recibido al usar los modos de operación CBC, OFB y CTR?
   Si usamos el modo CBC, el mensaje claro m_i depende de los textos cifrados c_i y c_{i+1} de la siguiente manera $m_i = D_K(c_i) \oplus c_{i-1}$. Si se pierde el bloque c_2 esto afectará a los mensajes claros m_2 y m_3, el mensaje m_2 lo descifraríamos como $m_2 = D_K(c_3) \oplus c_1$, lo que no tiene sentido. El mensaje m_3 lo descifraríamos como $m_3 = D_K(c_4) \oplus c_3 = m_4$, de esta manera, lo que pasaría es que sólo el mensaje m_2 dejaría de tener sentido y el mensaje m_3 no se recibe, los restantes mensajes se descifrarían de manera correcta.
   Si usamos el modo OFB, el mensaje claro m_i depnde de los anteriores mensajes de la siguiente manera $m_i = c_i \oplus E_K(c_{i-1} \oplus E_K(c_{i-2} \oplus \cdots))$. Asì, si se pierde el bloque c_2 el único bloque de texto que podríamos descifrar correctamente es el primero.
   Si usamos el modo CTR, también el único mensaje que podemos descifrar correctamente es el primero, pues en los restantes, el contador que usamos para descifrar el bloque c_i es en realidad el correspondiente al bloque c_{i+1}.

5. Investiga qué es el cifrado RC4. Da la descripción de su funcionamiento y sus debilidades.
   El cifrado RC4 (Rivest Cipher 4) es un cifrado de stream. Aunque es muy sencillo y rápido, se le han encontrado múltiples vulnerabilidades.
   Descripción:
   RC4 genera un stream de bits pseudoaleatorio que se puede usar para

cifrar combinandolo con un mensaje mediante la operación xor. En este sentido, es similar al cifrado de Vernam, excepto que se usa un stream pseudoaleatorio de bits en lugar de un stream previamente fijado.
Para generar el stream, el cifrado usa un estado interno que consiste de dos partes: una permutación S de los 256 posibles bytes y dos apuntadores de 8 bits i y j.
Algoritmo de key-scheduling
El algoritmo de key-scheduling se usa para inicializar la permutación S a partir de una llave inicia. Primero el array S es incializado con la permutación identidad. Después S es procesado por 256 iteraciones de la siguiente manera: Se inicia el apuntador j en el byte 0 y se inica un ciclo desde 0 hasta 255 en el apuntador i. El sigueinte valor de j es igual al valor inicial de j sumado con el valor de la permutación en el lugar j y el valor de la llave en la posición i (módulo la longitud de la llave) y todo esto se hace módulo 256 (para que la permutación siga siendo de los 256 bytes). Después, se cambia el valor de la permutación en el lugar i por el valor de la permutación en el lugar j (el cual calculamos). Tras repetir esto 256 veces, obtenemos la permutación incial S.
Algoritmo de generación pseudo-aleatorio (PRGA)
En cada iteración, el algoritmo hace lo siguiente:

- Incrementa el valor de i.
- Le añade a j el valor de la permutación S en la posición i.
- Cambia los valores de S en i por el de S en j y usa la suma de estos módulo 256 para obtener un tercer elemento de S, el cual es el siguiente bit de la llave.

Este proceso produce un stream de bits que se puden usar para hacer xor con el mensaje que queremos cifrar.

6. Determina si a es residuo cuadrático módulo n. Muestra tu procedimiento