Ejercicio 1

Enunciado

Para un conjunto de pruebas estadísticas se requiere una base de datos que contiene usuarios con sus datos personales (DNI, teléfono, etc) y su historial de navegación (URLs). Se desea usar un juego de datos productivos que reflejan el resultado de la recolección de dicha información durante 5 años, pero por la naturaleza sensible de los datos, estos no pueden disponibilizarse para las pruebas directamente. Se decide, pues, transformar los datos para que no puedan ser correlacionados, atendiendo los siguientes requisitos:

- Si dos valores son iguales, también deben serlo en el dataset final.
- No debe ser computacionalmente posible recuperar del dataset final NINGÚN valor del original.
- DNIs usados como clave primaria, deben mantener su formato de 9 dígitos.
- Si dos URLs apuntan al mismo servidor, con paths distintos, esto debe ser visible.

a. Explicar por qué un criptosistema CPA-Secure no puede ser utilizado para la transformación. (1 punto)
b. Explicar por qué una función de hash no puede ser utilizada para la transformación. (1 punto)
c. Mostrar cómo usar una MAC de algún tipo para realizar la transformación y que cumpla los requisitos. (1 punto)

a. Un criptosistema CPA-Secure no puede ser utilizado para la transformacion porque esto implicaria que la funcion sea no deterministica. Sin embargo, aca se nos pide que dos valores iguales tengan la misma salida i.e que la funcion sea deterministica.

b. Si se desea usar una funcion de hash, habria que usar algun modelo de SHA-3. De ser asi, no se estaria cumpliendo el tercer requisito i.e el DNI no se almacenaria con 8 digitos. Por otro lado, las funciones de hash no garantizan autenticidad. Es decir, no se puede saber quien ingreso los datos.

c. Se podria usar AES-GCM-SIV. Dado que la funcion de encripcion es deterministica con el mismo nonce, dos valores iguales resultan el mismo cifrado. Por otro lado, el mensaje y el cifrado tienen el mismo formato lo cual cumple con la restriccion del DNI. Finalmente, este cifrado autenticado solo pierde seguridad en el sentido que un atacante puede detectar si dos mensajes son iguales.

Ejercicio 2

Enunciado

Se armó un algoritmo para la elección de un nodo supervisor en caso de que se caiga un nodo en un sistema distribuido. Para que ningún nodo pueda influenciar en la selección del nuevo supervisor intencionalmente se usa un *commitment schema* donde cada uno de los n nodos genera un identificador aleatorio de 64 bits $\mathbf{L_i}$ y un secreto de 64 bits también aleatorio $\mathbf{Z_i}$ y usan una función criptográfica resistente a colisiones h cada nodo emite $(L_i, h(Z_i))$ y cuando todos los nodos envían estos datos, todos mandan (L_i, Z_i) y se calcula:

$$L_x = Z_1 \otimes Z_2 \otimes \dots \otimes Z_n$$

y se selecciona como nodo supervisor a aquel nodo cuyo identificador es el más cercano a L_x.

a. Explicar con una prueba criptográfica por qué si se altera el nodo 1 maliciosamente no se altera la elección del nuevo supervisor. (1.5 puntos)
b. Si todos los nodos siempre usan el mismo Z_i ¿cómo el nodo 1 podría manipular la elección? (1.5 puntos)

a. El problema es que el atacante nunca va poder saber el valor del identificador porque como se manda $(L_i, h(z_i))$ no se puede conocer L_x. Esto se debe a que las funciones de hash no son reversibles entonces es imposible que el atacante conozca todos los z_i.

b. Si todos los nodos mandan siempre el mismo z_i, el atacante en una primera ronda obtiene z_2 xor ... xor z_n y envía $z_1 = 1$. Ahora, sabiendo esto, en la segunda ronda envía $L_1 = 0$ y $z_1 = z_2$ xor ... xor z_n.

Ejercicio 3

Enunciado

Se tiene un sistema de apuestas online disponible en https://www.lacasagana.com en la cual usuarios pueden ingresar fondos con tarjetas de crédito y crear apuestas en base a un evento o unirse a apuestas y cuando el evento concluye se deposita en la cuenta bancaria informada por cada usuario la plata ganada. Se hace un pentest formular 4 hipótesis de falla pertinentes y explicar cómo probarlas. (4 puntos)

Hipotesis 1:
- Amenaza: spoofing ⇒ viola autenticacion
- Vulnerabilidad: SQL injection
- Prueba: ingresar 'pass OR 1=1' y ver si efectivamente me deja acceder a la cuenta

Hipotesis 2:
- Amenaza: denial of service y tampering ⇒ viola disponibilidad e integridad
- Vulnerabilidad: Integer Overflow

Esta vulnerabilidad puede causar un system crash o tambien puede causar buffer overflow ocasionando corrupcion de memoria.
- Prueba: apostar mas que MAX INTEGER

Hipotesis 3:
- Amenaza: spoofing ⇒ viola autenticacion
- Vulnerabilidad: Improper Authentication

Esta vulnerabilidad puede originarse por muchas causas. Por ejemplo, si los requisitos del password son pocos, encripcion pobre de la pass, etc.
- Pruebas:
  - Probar usuarios y contraseñas comunes
  - Intentar acceder al sitio sin autenticacion
  - Registrarse con contraseñas debiles

Hipotesis 4:
- Amenaza: information disclosure ⇒ viola confidencialidad
- Vulnerabilidad: Sensitive data exposure
- Pruebas:
  - Revisar trafico interno
  - Downgrade el protocolo a alguno vulnerable
  - Ejecutar un man-in-the-middle attack