Ejercicio 1

Enunciado

El Sistema de seguridad del museo Oddisey está completamente automatizado. Cada cuadro o pieza de tamaño grande tiene adosada una etiqueta de $1mm^2$ que es un sensor de seguridad. Asimismo, cada vitrina que protege a objetos más pequeños o delicados tiene adosado al vidrio blindado otro sensor de iguales características.
Estos sensores no detectan movimiento, pero pueden transmitir una señal y escuchar la señal transmitida por sus vecinos (en un radio de aproximadamente un metro).
Cuando se activa el sistema, cada sensor mira que sensores tiene cerca y los deja registrados. Luego, cada 5 segundos vuelve a repetir la medición y si encuentra alguna discrepancia, emite una señal de alerta.
La señal de alerta, al ser detectada por cualquier sensor, es retransmitida a todos sus vecinos, que a su vez la retransmiten, y así sucesivamente hasta llegar a un panel que existe en cada habitación que está conectado a la alarma central del museo y da aviso a los guardias.
Los sensores cuentan con un par de claves pública y privada de 320 bits (para utilizar con criptosistemas asimétricos de curvas elípticas) propias provistas de fábrica y a priori no conoce a ninguno de los otros sensores.
La versión inicial del sistema transmitía todas las señales en plano, pero un análisis de seguridad encontró una serie de problemas. Para cada problema proponga cómo modificaría el comportamiento de los sensores al enviar y al recibir señales, de tal manera que cada modificación NO deshaga la seguridad conseguida por las modificaciones anteriores (dicho de otra manera, todos los cambios juntos deben terminar en un sistema que no tenga ninguna de las vulnerabilidades encontradas):

a. Fue posible reemplazar un elemento con su sensor por un transmisor programado para emitir las mismas señales luego de escuchar al sensor unos minutos (2 puntos).
b. Fue posible desactivar el sistema transmitiendo una señal de apagado (1 punto).
c. Fue posible disparar la alarma utilizando un transmisor que emita la señal que los sensores interpretaban como alarma (1 punto).

Al configurar el sistema, se comparten las claves publicas de los sensores a sus vecinos via un canal seguro como SSL. Se recomienda cambiar las claves que vienen de fabrica porque un atacante podria conocerlas de antemano.
Luego, las señales del sensor A al sensor B son de la siguiente forma:

$$\{señal, timestamp, S(señal, timestamp)_{SKA}\}_{PKB}$$

- A firma la señal y el timestamp con su clave privada
- A encripta todo el mensaje con la clave publica de B
- B, con su clave privada, desencripta el mensaje
- B, con la clave publica de A, verifica la firma

a. Este problema se resuelve con el timestamp que evita que se envien las mismas señales.
b. No se puede obtener la señal de apagado porque las señales estan encriptadas. Es decir, aunque el atacante capture el mensaje no lo puede desencriptar.
c. Por la misma razon que el punto b, un atacante no puede conocer la señal de alarma.
Asimismo, la firma digital evita que un atacante pueda enviar cualquier tipo de mensaje puesto que no conoce ninguna clave privada.

Ejercicio 2

Enunciado

Existe un tipo de criptosistema denominado Format Preserving Encryption (FPE). La idea de este tipo de criptosistemas es que P = C = {0, 1, ..., N} para un N arbitrario. Esto permite, que el texto plano y cifrado sean del mismo tipo, propiedad muy útil para cifrar datos sin modificar la forma en que se almacenan o transmiten.
Por ejemplo, se podría cifrar un número de tarjeta de crédito, de tal forma que el resultado sea otro número de tarjeta, o un número de x dígitos, de tal forma que el resultado sea otro número de esa misma cantidad de dígitos.
Además de criptosistemas específicos de este tipo, hay formas de convertir un criptosistema de bloque en un FPE. Una de ellas funciona de la siguiente manera:
Sea Ek(x) una primitiva de cifrado en bloque de tamaño n. Se define FPEk(x), primitiva de cifrado con P = C = {0, 1, ..., m-1} / $m < 2^n$ a través del siguiente pseudocódigo:

```
fpe = x
do {
      x = fpe
      fpe = Ek(x)
} while (fpe >= m)
```

⇒ encripto hasta que el cifrado tenga el mismo tamaño que el mensaje

Un uso interesante de estas funciones es la creación de códigos para cupones.
Por ejemplo, si consideramos los cupones como números de 20 dígitos, podemos generar 10.000 cupones cifrando los números del 0 al 9.999.
Siguiendo los parámetros del ejemplo, contestar:

a. ¿Cuál es la probabilidad de que un atacante genere un cupón válido? (1 punto)
b. ¿Cambia en algo la seguridad si se decide cifrar los números del 10.000 a 19.999 en lugar de los originales? Justificar (0.5 puntos)
c. ¿Varía el esfuerzo de un atacante para generar un cupón válido si construimos el fpe con una primitiva de cifrado con claves de 64 bits, 128 o 256 bits? Justificar (1 punto).
d. Dado un cupón de 20 dígitos, ¿Cómo se puede verificar si el cupón es válido? (0.5 puntos)

Nota: 10^{20} es aprox. $2^{66,4}$

a. Para generar un cupon valido, el atacante deberia tener la clave porque $E_k(x)$ es pseudo aleatorio i.e un atacante no puede determinar ningun patron de los numeros de los patrones. Luego, si la clave es de n bits, la probabilidad de que obtenga la clave es $1/2^n$.

Otra alternativa: fuerza bruta

Para generar un cupon valido de un numero valido, la probabilidad es $1/10^{20}$. Entonces, para 10000 numeros, la probabilidad es de $\frac{10000}{10^{20}} = \frac{1}{10^{16}}$.

b. No, porque la seguridad no depende de los numeros a cifrar sino de la clave.

c. Si queremos generar un cupon valido a partir de la clave, entonces mientras mas larga sea la clave, mayor sera el esfuerzo para encontrarla. Sin embargo, si se utiliza la otra alternativa, sin encontrar clave, el esfuerzo es el mismo

d. Habria que hacer la inversa y verificar que el resultado este entre 0 y 9999.

```
msg = c
do {
    c = msg
    msg = Dec_k(c)
} while (msg < m)
```

Ejercicio 3

Enunciado

Honoris es un sistema online que estima la reputación de un jugador a través de integraciones con más de 100 juegos en línea.
Cada juego define hitos con los cuales otorga puntuación. Los administradores de cada juego determinan qué hitos otorgan puntuación. Los administradores de honores determinan para cada juego un coeficiente que se multiplica al valor del hito a la hora de sumar reputación (con esto balancean juegos que tienen hitos que se consiguen en días, de aquellos en los cuales se tardan meses años). Los desarrolladores utilizan una API REST expuesta por el sistema para notificar la concreción de hitos por parte de los jugadores. Dicha API usa TLS v1.0 y requiere que los headers lleven un API Secret (64 caracteres aleatorios otorgados a cada empresa al registrarse).
A fines de mantener simple la plataforma, todavía no se consideró la posibilidad de que la puntuación de los hitos varíe o que se agreguen nuevos hitos para juegos existentes.
Honoris fue un éxito y consiguió llegar a un millón de suscripciones en dos meses, pero los administradores comenzaron a notar la aparición de muchísimas cuentas que tenían una reputación imposible de alcanzar siquiera en un año de jugar todos los días 8 horas.
Luego de descartar un bug en el sistema, la empresa decide contratar una prueba de seguridad.

a. Con la información conocida, establecer 4 hipótesis plausibles, ordenadas de la más probable a la menos probable, que haya que revisar para encontrar el/los problemas. Explicar por qué considera que cada hipótesis es altamente probable y qué esperaría encontrar (2 puntos).
b. Explicar cómo probaría las dos hipótesis más probables siguiendo la metodología de un pen-test (1 punto)

a. 4 vulnerabilidades

(1) Sensitive data exposure
Amenazas: tampering, information disclosure y spoofing
Esta vulnerabilidad se debe a que el protocolo utilizado es viejo. Un atacante podría interceptar los paquetes y desencriptar el API secret y luego replicar los mensajes con los datos de su cuenta.

(2) Improper authentication
Amenazas: tampering, information disclosure y denial of service
Esta vulnerabilidad se debe a que las funciones de las aplicaciones relacionadas con autenticacion no estan implementadas correctamente. Un atacante podria hacerse pasar por un admin y notificar hitos a su cuenta.

(3) SQL Injection
Amenazas: information disclosure, spoofing y tampering
Esto se debe a que no se hacen las verificaciones necesarias en cuanto a los inputs de los usuarios. Entonces, un atacante podria modificar los hitos de la base de datos y asignarse a si mismo mas hitos.

(4) Improper authorization
Amenazas: information disclosure y tampering
Esta vulnerabilidad ocurre cuando el sistema no hace los controles necesarios para los accesos a recursos. En este caso, puede pasar que el sistema no haga los controles sobre quien accede a los hitos. Luego, un atacante podria acceder a los hitos de otros juegos.

b. Pruebas

(1) Sensitive data exposure

- Se puede hacer SQL Injection sobre los API secret y ver si los devuelve desencriptados
- Hacer un mitm attack y reenviar el paquete con otros datos

(2) Improper authentication
- Probar contraseñas simples en los usuarios admin
- Intentar acceder sin autenticacion a URLs que deberian estar restringidas
- SQL Injection sobre la pass