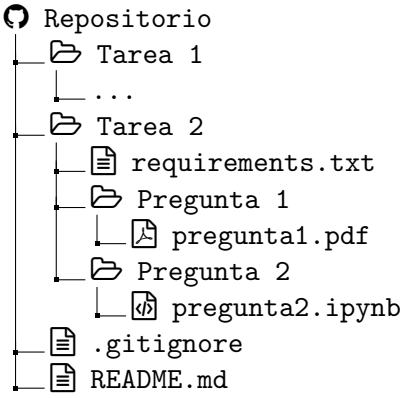
PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

Criptografía y Seguridad Computacional - IIC3253
Solución Pregunta 1
Tarea 2

Instrucciones

Cualquier duda sobre la tarea se deberá hacer en los *issues* del repositorio del curso. Si quiere usar alguna librería en sus soluciones debe preguntar primero si dicha librería está permitida. El foro es el canal de comunicación oficial para todas las tareas.

Entrega. Para entregar esta segunda tarea usted deberá utilizar el mismo repositorio que usó para la tarea 1. Al entregar esta segunda tarea, la estructura de archivos de su repositorio se deberá ver exactamente de la siguiente forma:



Además, deberá considerar lo siguiente:

- El archivo `requirements.txt` deberá especificar todas las librerías que se necesitan instalar para ejecutar todas las respuestas en código de su tarea. Este archivo debe seguir la especificación de Pip, es decir se debe poder ejecutar `pip install -r requirements.txt` suponiendo una versión de `Pip` mayor o igual a 20.0 que apunta a la versión 3.9 de Python.
- La solución de cada problema de programación debe ser entregada como un Jupyter Notebook (esto es, un archivo con extensión `ipynb`). Este archivo debe contener comentarios que expliquen claramente el razonamiento tras la solución del problema, idealmente utilizando

markdown. Más aun, su archivo deberá ser exportable a un módulo de python utilizando el comando de consola

```
jupyter-nbconvert --to python preguntaX.ipynb
```

Este comando generará un archivo `preguntaX.py`, del cual se deben poder importar las funciones que se piden en cada pregunta.

- Para cada problema cuya solución se deba entregar como un documento (en este caso la pregunta 1), usted deberá entregar un archivo `.pdf` que, o bien fue construido utilizando LaTeX, o bien es el resultado de digitalizar un documento escrito a mano. En caso de optar por esta última opción, queda bajo su responsabilidad la legibilidad del documento. Respuestas que no puedan interpretar de forma razonable los ayudantes y profesores, ya sea por la caligrafía o la calidad de la digitalización, serán evaluadas con la nota mínima.

Pregunta

En clases se presentó en detalle la estructura del protocolo criptográfico simétrico DES (Data Encryption Standard). En particular, vimos que DES es una red de Feistel de 16 pasos, cada uno de los cuales consiste en una red de sustitución/permutación de una sola ronda. El único punto que no se discutió en detalle fue la forma en la que se derivan las sub-llaves: en cada paso de la red de Feistel es necesario aplicar una función $f(k_i, R_i)$, donde f es la red de sustitución/permutación, R_i es un estado interno de la red y k_i es la *sub-llave* correspondiente al paso i. Esta sub-llave se deriva de forma determinista en base a la llave inicial de acuerdo a lo que se conoce como el *key schedule*.

a) Investigue y describa en detalle el *key-schedule*, es decir, explique cómo se deriva la llave correspondiente a cada ronda en base a la llave original.

b) Considerando que el *key-schedule* divide la llave en dos mitades que son prácticamente *independientes*, demuestre que si en vez de 16 rondas DES utilizara sólo 3 rondas, entonces existiría un ataque de texto plano que recupera la llave sin ejecutar la red de sustitución/permutación más de 2^{30} veces.

Solución

a) El *key-schedule* de DES utiliza, para cada ronda i, una llave k_i de 48 bits cuya definidicón es inductiva. Sea k la llave secreta utilizada. La llave k_i dependerá de dos valores provenientes de la ronda anterior, que denominaremos C_{i-1} y D_{i-1}.

- Comenzamos definiendo los valores iniciales C_0 y D_0. Para obtenerlos, la llave k es operada de acuerdo a una función llamada PC_1 (Permuted-Choice 1), que tiene como output dos valores, precisamente C_0 y D_0. Suponiendo que la llave original (de 64 bits) es $k = b_1 b_2 \cdots b_{64}$, tenemos:

$$C_0 = b_{57}b_{49}b_{41}b_{33}b_{25}b_{17}b_9b_1b_{58}b_{50}b_{42}b_{34}b_{26}b_{18}b_{10}b_2b_{59}b_{51}b_{43}b_{35}b_{27}b_{19}b_{11}b_3b_{60}b_{52}b_{44}b_{36}$$

$$D_0 = b_{63}b_{55}b_{47}b_{39}b_{31}b_{23}b_{15}b_7b_{62}b_{54}b_{46}b_{38}b_{30}b_{22}b_{14}b_6b_{61}b_{53}b_{45}b_{37}b_{29}b_{21}b_{13}b_5b_{28}b_{20}b_{12}b_4$$

- Ahora mostramos cómo, en base a C_{i-1} y D_{i-1}, podemos obtener k_i, C_i y D_i. Comenzamos aplicando una rotación de bits hacia la izquierda a los valores C_{i-1} y D_{i-1}, lo que nos dará precisamente los valores C_i y D_i, respectivamente. Dicha rotación es de 1 bit si $i \in \{1, 2, 9, 16\}$ y de dos bits en todos los otros casos. Esto significa, por ejemplo, que para la ronda 1 tendremos

$$C_1 = b_{49}b_{41}b_{33}b_{25}b_{17}b_9b_1b_{58}b_{50}b_{42}b_{34}b_{26}b_{18}b_{10}b_2b_{59}b_{51}b_{43}b_{35}b_{27}b_{19}b_{11}b_3b_{60}b_{52}b_{44}b_{36}b_{57}$$

$$D_1 = b_{55}b_{47}b_{39}b_{31}b_{23}b_{15}b_7b_{62}b_{54}b_{46}b_{38}b_{30}b_{22}b_{14}b_6b_{61}b_{53}b_{45}b_{37}b_{29}b_{21}b_{13}b_5b_{28}b_{20}b_{12}b_4b_{63}$$

  Finalmente, para obtener la llave k_i, simplemente aplicamos una función denominada PC_2 (Permuted-Choice 2) sobre la concatenación de C_i y D_i, es decir $k_i = PC_2(C_i||D_i)$. Para generar una llave de k_i de 48 bits, PC_2 deja fuera 8 bits de $C_i||D_i$, que son los bits 9, 18, 22, 25, 35, 38, 43 y 54. Suponiendo que $C_i||D_i = b_0^i b_1^i \cdots b_{56}^i$ entonces tenemos

$$k_i = PC_2(C_i||D_i) = b_{14}^i b_{17}^i b_{11}^i b_{24}^i b_1^i b_5^i b_3^i b_{28}^i b_{15}^i b_6^i b_{21}^i b_{10}^i b_{23}^i b_{19}^i b_{12}^i b_4^i b_{26}^i b_8^i b_{16}^i b_7^i b_{27}^i b_{20}^i b_{13}^i b_2^i$$
$$b_{41}^i b_{52}^i b_{31}^i b_{37}^i b_{47}^i b_{55}^i b_{30}^i b_{40}^i b_{51}^i b_{45}^i b_{33}^i b_{48}^i b_{44}^i b_{49}^i b_{39}^i b_{56}^i b_{34}^i b_{53}^i b_{46}^i b_{42}^i b_{50}^i b_{36}^i b_{29}^i b_{32}^i$$

Con esto tenemos completamente definido de forma inductiva el key-schedule de DES. Es importante notar que cada uno de estos valores C_i y D_i tiene 28 bits, mientras que la llave original tenía 64. El último bit de cada byte (es decir los bits 8, 16, 24, 32, 40, 48, 56 y 64) no son utilizados, puesto que son simplemente bits de paridad de cada byte de la llave. Esto significa que la llave efectivamente es de 56 bits. También es importante mencionar que C_0 y D_0 no comparten información, lo cual también será cierto para cada par C_i y D_i.

b) Para describir nuestro ataque utilizaremos los siguientes tres lemas:

Lema 1. *Los primeros 24 bits de cada sub-llave k_i se pueden determinar en base a 28 bits de la llave k. De la misma forma, los segundos 24 bits de k_i se pueden determinar en base a los 28 bits en las posiciones restantes de la llave k.*

Demostración. Podemos notar que C_0 depende exclusivamente de la mitad de los bits de la llave k (los bits en las posiciones 57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18, 10, 2, 59,

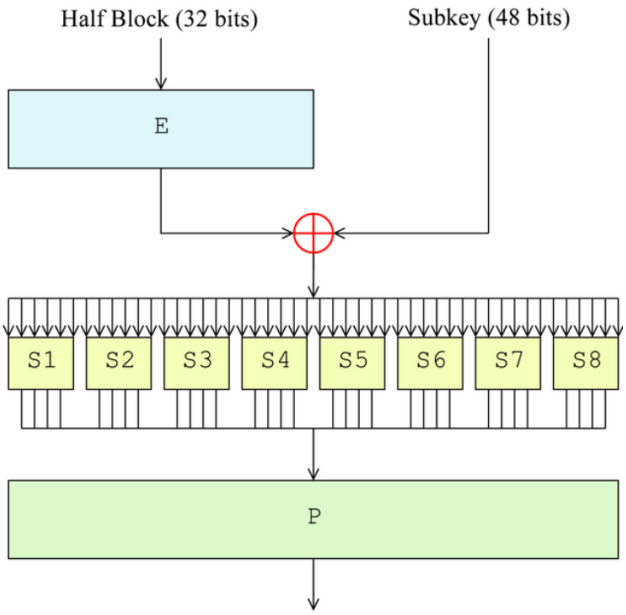


Figura 1: Esquema de la red de sustitución-permutación de DES. Fuente: Wikipedia.

51, 43, 35, 27, 19, 11, 3, 60, 52, 44 y 36). También vemos que todos los C_i se pueden escribir como rotaciones de C_0, por lo que los mismos bits de la llave original determinan por completo todos los valores C_i. Ahora, inspeccionando PC_2 podemos ver que los primeros 24 bits del output de PC_2 se obtienen en base a los primeros 28 bits del input. Como los primeros 28 bits del input en cada ronda i son exactamente C_i, tenemos que los primeros 24 bits del output son determinados por C_i, que como mencionamos es determinado por exactamente los 28 bits de k que se mencionan arriba. El análisis sobre la segunda mitad del output de PC_2 con D_i es análogo. Como k_i es exactamente el output de PC_2, tenemos el resultado esperado. □

Lema 2. *Sea f la red de sustitución/permutación utilizada en DES. Sea $x_i = f(k_i, R_{i-1})$ para cada ronda i. Para toda ronda i, existen 16 posiciones de x_i que no varían al modificar los últimos 24 bits de k_i. Las otras 16 posiciones no varían al modificar los primeros 24 bits de k_i.*

Demostración. En la Figura 1 se muestra la estructura de la red de sustitución/permutación utilizada en DES, donde *Half Block* se refiere a R_{i-1} y *Subkey* se refiere a k_i. Es fácil ver que los últimos 24 bits de k_i no afectan el input input de las 4 S-boxes de más a la derecha. Como P es una permutación sobre 32 bits cuyo output es x_i, 16 de dichos bits son determinados precisamente por el output de las primeras 4 S-boxes, lo que nos da el resultado esperado. El análisis para los otros 16 bits es análogo. □

Lema 3. *Sea k una llave para DES. Dado un par $(m, c := Enc(k, m))$ y 28 bits cualquiera, podemos decidir si dichos 28 bits corresponden a la mitad de llave k ejecutando dos veces la red de sustitución/permutación de DES con una probabilidad de error de $1/2^{16}$.*

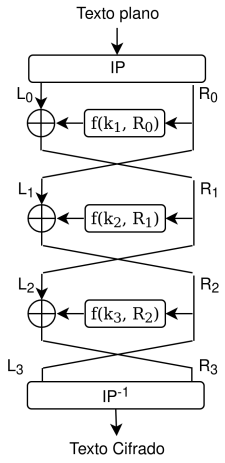


Figura 2: DES con tres rondas.

Demostración. Sea k una llave para DES (suponemos 8 bytes, cada uno con un bit de paridad) y (m, c) tal que $c = Enc(k, m)$. De acuerdo a la notación utilizada en la Figura 2, podemos obtener inmediatamente los elementos L_0, R_0, L_3 y R_3. Mirando la estructura de la red, nos damos cuenta de que además tenemos $R_2 = L_3$ y $L_1 = R_0$. La demostración se basará en que $R_1 = R_3 \oplus f(k_3, R_2)$ pero también $R_1 = L_0 \oplus f(k_1, R_0)$, usando los dos lemas anteriores.

Sea B un string de 28 bits. Definimos k_B como la llave construida poniendo estos 28 bits en las posiciones 57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18, 10, 2, 59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44 y 36, y el bit cero en las 28 posiciones restantes. De acuerdo al Lema 1, los primeros 24 bits de k_1 y k_3 se determinan exclusivamente en base a los bits de B. Más aun, de acuerdo al Lema 2, existen 16 bits del output de $f(k_3, R_2)$ y de $f(k_1, R_0)$ que no dependen de los 28 bits que no fueron sacados de B. Esto significa que los bits en esas mismas 16 posiciones de $R_3 \oplus f(k_3, R_2)$ y $L_0 \oplus f(k_1, R_0)$ no dependen de los 28 bits que no fueron sacados de B. Ahora, como $R_3 \oplus f(k_3, R_2) = L_0 \oplus f(k_1, R_0)$, podemos verificar que los bits en dichas 16 posiciones que resultan de ambos XOR son iguales. Si son iguales, entonces diremos que los bits de B corresponden a los bits de la llave original k en las posiciones ya mencionadas. De lo contrario diremos que no corresponden.

Ahora necesitamos calcular la probabilidad de error del procedimiento anterior. Si B efectivamente tiene los bits de la llave en las posiciones correctas, entonces el procedimiento no se equivoca. De lo contrario, podría ser que bits incorrectos entreguen el mismo resultado. Para calcular la probabilidad de que eso pase podemos pensar simplemente que el resultado de $L_0 \oplus f(k_1, R_0)$ está fijo, y lo que tiene que pasar es que 16 bits de $R_3 \oplus f(k_3, R_2)$ correspondan a 16 bits fijos. Si suponemos que los mensajes y las llaves son aleatorias, esto ocurre con probabilidad $1/2^{16}$. Con esto la demostración queda completa, ya que tuvimos que ejecutar la red de sustitución-permutación (la función f) exactamente dos veces. □

Procedemos finalmente a describir nuestro ataque. Recordemos que un ataque de texto plano supone que el atacante tiene acceso a una cantidad polinomial de pares $(m_i, c_i := Enc(k, m_i))$ para una llave k fija. Para nuestro ataque bastará con tener acceso a dos de dichos pares, (m_0, c_0) y (m_1, c_1). Comenzamos utilizando el primer par. De acuerdo al Lema 3, podemos

usar m_0 y c_0 para verificar si 28 bits de la llave *podrían ser* correctos ejecutando dos veces la red de sustitución permutación. Si intentamos con todas las mitades de llaves posibles, tendremos que ejecutar la red de sustitución/permutación $2 \cdot 2^{28} = 2^{29}$ veces. Esto nos sirve para la mitad de la llave, por lo que si hacemos lo mismo para la otra mitad de la llave (los 28 bits restantes), estaríamos utilizando la red de sustitución/permutación exactamente $2 \cdot 2^{29} = 2^{30}$ veces. El problema con esto es que para cada mitad de la llave obtenemos (en valor esperado) $2^{28}/2^{16} = 2^{12}$ candidatos, por lo que tendremos 2^{24} llaves candidato. Lo que haremos ahora es simplemente encriptar con cada una de dichas llaves el mensaje m_1 y verificar si obtenemos c_1. La probabilidad de que esto ocurra usando una llave aleatoria es cercana a $1/2^{56}$, por lo que tras este proceso obtendremos con alta probabilidad una sola llave, la llave correcta.

Aunque nuestro ataque funciona, tuvimos que utilizar la red de sustitución-permutación $2^{30} + 3 \cdot 2^{24} > 2^{30}$ veces. Para bajar este número basta con notar que el ataque de fuerza bruta sobre cada mitad de la llave se puede hacer en paralelo, utilizando la red de sustitución-permutación sólo dos veces para probar con dos candidatos, uno para cada mitad de la llave. Esto significa que para obtener las aproximadamente 2^{24} llaves candidato basta con ejecutar la red de sustitución/permutación $2 \cdot 2^{28} = 2^{29}$ veces, con lo que el ataque completo necesita $2^{29} + 3 \cdot 2^{24} < 2^{30}$ ejecuciones de la red.