Criptografía y Seguridad Computacional **2016-01**

Clase 4: 23/03/2016

Profesor: Fernando Krell *Notas: Manuel Cartagena*

1. Modos de operación para cifradores de bloque

En las clases pasadas hemos estudiadio funciones pseudo-aleatorias. Estas son usadas, entre otras cosas, para construir esquemas de cifrado simétricos que son seguros bajo ataques de texto plano escogido (CPA). Las funciones pseudo-aleatorias también son llamadas cifradores de bloque, ya que permiten cifrar un mensaje arbitrario dividiendolo en bloques de largo fijo y utilizar la función bloque por bloque.

Sea $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ una función psuedo-aleatoria. En la clase anterior vimos que si el mensaje es de largo fijo n podemos construir esquema seguro en el sentido CPA aplicando la función pseudo'aleatoria sobre un string aleatorio, y ejecutando la función XOR entre el resultado y el mensaje. El texto cifrado contiene también el string aleatorio. Formalmente:

- $\mathsf{Gen}(1^\lambda)$: Output $k \sim U_\lambda$.
- $\mathsf{Enc}_k(m)$:
  1. $r \sim U_n$.
  2. $c_1 = F_k(r) \oplus m$
  3. output $c = \langle c_0 = r, c_1 \rangle$
- $\mathsf{Dec}_k(c = \langle c_0, c_1 \rangle)$: Ourput $m = F_k(c_0) \oplus c_1$.

Podemos observar que el esquema anterior tiene dos incovenientes. En primer lugar, el tamaño del texto cifrado es el doble del tamaño original del texto plano. A la vez, el esquema sólo esta definido para mensajes de largo fijo n. A continuación veremos varias propuestas para superar estas limitaciones

1.1. EBC (Electronic code book)

El primer esquema es llamada EBC por Electronic Code Book en inglés. La idea es dividir el mensaje en bloques de largo n y aplicar la función pseudo-aleatoria bloque a bloque. Este esquema tiene la ventaja de que el texto cifrado tiene tamaño igual al del texto plano y el largo del mensaje sólo tiene que ser multiplo del tamaño del bloque. Además el esquema es paralelizable. Sin embargo, es fácil de ver que el esquema no es seguro en el sentido CPA, ya que, para empezar, el algoritmo de cifrado es determinístico. Otro punto interesante de observar es que la función pseudo'aleatoria debe ser también una permutación, ya que si no, seria imposible descifrar.

- $\mathsf{Gen}(1^\lambda)$: Output $k \sim U_\lambda$
- $\mathsf{Enc}_k(m)$:
  1. $m_1||m_2||\ldots||m_\ell \leftarrow m,\ |mi| = |m_j| = n \forall i, j$
  2. $c_i = F_k(m_i) \forall i \in [1..\ell]$
  3. Output $c = \langle c_1, c_2, \ldots c_\ell \rangle$
- $\mathsf{Dec}_k(c = \langle c_1, c_2, \ldots c_\ell \rangle)$:
  1. $m_i = F_k^{-1}(m_i) \forall i \in [1..\ell]$
  2. Output $m = c_1||c_2||\ldots||c_\ell \rangle$

Adversario CPA: Consultar oráculo por m de largo n, obtener $\mathsf{Enc}_k(m) = F_k(m)$. Devolver par $\langle m_0 = m, m_1 = 0^n \rangle$. Al obtener $c = \mathsf{Enc}_k(m_b) = F_k(m_b)$, si $c = F_k(m)$ output 0, en otro caso output 1.

1.2. CBC (Cipher-block chaining)

El segundo esquema es llamado cadena de bloques cifrados (o CBC). La idea es formar una cadena en donde cada bloque del texto plano es cifrado aplicando primero XOR con el texto cifrado correspondiente el bloque anterior, y luego aplicando la funcion pseudo aleatoria. El primer bloque es cifrado utilizando un string uniformemente aleatorio en vez el texto cifrado del bloque anterior. Intuitivamente, el esquema es seguro pues el mensaje es cifrado utilizando un string pseudo-aleatorio como pad, y luego utilizando la función pseudo-aleatoria para esconder el pad y el mensaje para cifrar el siguiente bloque. El string aleatorio tulizado para cifrar el primer mensaje es comúnmente llamado el vector de inicializacion (o IV). La gran ventaja de este cifrador es que es CPA seguro.

- $\mathsf{Gen}(1^\lambda)$: Output $k \sim U_\lambda$
- $\mathsf{Enc}_k(m)$:
  1. $m_1||m_2||\dots||m_\ell \leftarrow m,\ |mi| = |m_j| = n \forall i, j$
  2. $c_0 = IV \sim U_n$
  3. $c_i = F_k(m_i \oplus c_{i-1}) \forall i \in [1..\ell]$
  4. Output $c = \langle c_0, c_1, c_2, \dots c_\ell \rangle$
- $\mathsf{Dec}_k(c = \langle c_0, c_1, c_2, \dots c_\ell \rangle)$:
  1. $m_i = F_k^{-1}(m_i \oplus c_{i-1}) \forall i \in [1..\ell]$
  2. Output $m = m_1||m_2||\dots||m_\ell \rangle$

Podemos ver que el algoritmo de cifrado es sequencial. Sin embargo, el descifrado es paralilizable ya que no es necesario descifrar ningún bloque como condición para descifrar otro. En CBC también es necesario que F sea una permutación pseudo-aleatoria.

1.3. OFB

El output feedback mode es un modo de operación muy similar a CBC. La principal diferencia es que cada bloque es cifrado primero computando la funcion pseudo-aleatoria sobre un string pseudo-aleatorio, y luego aplicando XOR sobre el mensaje. En este caso requerimos que el string pseudo-aleatorio no sea el cifrado del bloque anterior, sino el "pad" utilizado como XOR para cifrar el bloque anterior.

- $\mathsf{Gen}(1^\lambda)$: Output $k \sim U_\lambda$
- $\mathsf{Enc}_k(m)$:
  1. $m_1||m_2||\dots||m_\ell \leftarrow m,\ |mi| = |m_j| = n \forall i, j$
  2. $r_0 = IV \sim U_n$
  3. $r_i \leftarrow F_k(r_{i-1}),\ c_i = m_i \oplus F_k(r_i) \forall i \in [1..\ell]$
  4. Output $c = \langle r_0, c_1, c_2, \dots c_\ell \rangle$
- $\mathsf{Dec}_k(c = \langle r_0, c_1, c_2, \dots c_\ell \rangle)$:
  1. $r_i \oplus F_k(r_{i-1}), m_i = c_i \oplus F_k^{-1}(r_i) \forall i \in [1..\ell]$

2. Output $m = m_1||m_2||\ldots||m_\ell\rangle$

Este tipo de cifrador es un tipo de *cifrador stream*. En el cual el mensaje completo es cifrado aplicando XOR sobre pad pseudo-aleatorio. El pad utilizado aqui es r, $F(r)$, $F(F(r))$,$F(F(F(r)))$, etc. Una ventaja de OFB sobre CBC es que el pad puede ser precomputado. A la vez, una vez precomputado el pad, los algoritmo de cifrado y descifrado son paralelizables. Podemos observar que este mode de operación no es necesario que F sea una permutación (no se requiere computar F^{-1}).

1.4. CTR (Modo contador)

El último modo de operación que veremos es el modo contador. El modo contador es una pequeña modificación al mode OFB de la sección anterior. La diferencia en este caso es que el pad en cada bloque "no depende" del pad para el bloque anterior. Como en los modos anteriores elegimos un vector de inicialización uniformemente aleatorio en $\{0,1\}^n$ y el pad del bloque i es computado como $F_k(IV + i)$.

- $\mathsf{Gen}(1^\lambda)$: Output $k \sim U_\lambda$
- $\mathsf{Enc}_k(m)$:
  1. $m_1||m_2||\ldots||m_\ell \leftarrow m,\ |mi| = |m_j| = n \forall i, j$
  2. $IV \sim U_n$
  3. $c_i = m_i \oplus F_k(IV + i) \forall i \in [1..\ell]$
  4. Output $c = \langle r_0, c_1, c_2, \ldots c_\ell\rangle$
- $\mathsf{Dec}_k(c = \langle IV, c_1, c_2, \ldots c_\ell\rangle)$:
  1. $m_i = c_i \oplus F_k(IV + i) \forall i \in [1..\ell]$
  2. Output $m = m_1||m_2||\ldots||m_\ell\rangle$

La gran ventaja de este modo sobre los anteriores es que es paralelizable en preprocesamiento y online.

Ejercicio 1. *Demuestre que los modos CTR, OFB y CBC son seguros bajo ataques de texto plano escogid (CPA)*

Ejercicio 2. *¿Cuales son las ventajas y desventajas de los modos de operación anteriores si un adversario puede modificar el texto cifrado?. ¿Que tan bien cada uno de estos modos de operación vistos protege frente al cambio de un sólo bit en el texto cifrado?*

2. Construcción práctica de cifradores de bloque

En esta sección introduciremos un diseño de como construir cifradores de bloque en la práctica. Cabe mencionar que estos cifradores son puramente heuristicas y no existe demostración alguna que sean realmente seguros.

Redes de sustitución y permutación : Sea $x \in \{0,1\}^n$, el mensaje se dividirá en ℓ bloques de tamaño m. Cada uno de estos bloques pasara por una función de sustitución. Luego los nuevos bloques se concatenan para los n bits son permutados de manera de repartir la susticion realizada en cada bloque a lo largo del output. La idea es que si un bit de input cambia. Varios bits del outut repartidos en diferentes partes de veran afectados.

El proceso anterior se le llama una ronda, la cual se repite varias veces de manera de asegurar pseudo-aleatoriedad (por ejemplo, se esperaria que si un solo bit del input cambia, cada bit del output cambia con pbb 1/2).

AES : El estandar avanzado de cifrado (AES por sus siglas en inglés) es el cifrador de bloque ganadore del concurso del NIST en 2001. AES ha sido fuertemente estudiado durante ya decadas y no se ha encontrado ningún ataque mejor que un ataque de fuerza bruta. Este cifrador es considerado altamente seguro dentro de la comunidad criptográfica.

El tamaño de bloque soportado por AES es de 128 bits (16 bytes), y permite llaved de 128, 192 y 256 bits. AES esta basado en una red de permutación y sustitución. La llave k es expandida a N sub-llaves $k^1, k^2, \dots k^N$ de 16 bytes cada una. Cada sub-llave es utilizada en una ronda distanta.

Sustitución: Primero se aplica XOR a cada byte del input con la llave, y luego el resultado es pasado por una tabla llamada S-BOX, que garantiza que si un solo bit del input cambia, al menos 2 bits del output de S cambian también. Notar que esta S-BOX, puede ser implementada manteniendo una tabla de 256 bytes.

Permutación: Luego los 16 bytes resultantes son ordenados en una matriz de 4×4 bytes, la i-ésima fila es corrida ciclicamente a la izquierda en i bytes:

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_5 & x_6 & x_7 & x_4 \\ x_{10} & x_{11} & x_8 & x_9 \\ x_{15} & x_{12} & x_{13} & x_{14} \end{pmatrix}$$

Para teminar la ronda, una transformación lineal T es aplicada a cada columna de la matrix.

$\mathsf{Exp}^{mac-forge}_{\mathcal{A},\Pi}$

1. $k \leftarrow \mathsf{Gen}(1^\lambda)$ 2. $(m,t) \leftarrow \mathcal{A}^{\mathsf{Mac}_k(\cdot)}$ 3. output 1 ssi $\mathsf{Vrfy}_k(m,t) = 1$

Figura 1: Experimento Mac-Forge

3. Integridad de mensajes

Lo primero que vimos en el curso es como poder enviar un mensaje sin que algún adversario en medio del proceso de envío pudiera entender el contenido del mensaje.

Ahora queremos saber si el mensaje que recibié proviene efectivamente de la persona que dice que lo está enviando, ya que no sabemos si hay un adversario que pasa de ser un simple observador, a ser un adversario activo que puede alterar el mensaje, o enviar otro completamente distinto haciendose pasar por otra persona.

Tenemos que Alice y Bob poseen una llave secreta compartida k. Para asegurar integridad enviaremos el mensaje m en conjunto con un "tag" $t_k(m)$, el cual ayuda a verificar m es el mensaje originalmente enviado por Alice.

3.1. Códigos de autenticación de mensajes(MAC)

Un código de autenticación de mensajes Π consiste de los siguientes 3 algoritmos:

- $\mathsf{Gen}(1^\lambda) \to k$, es el algoritmo generador de llaves.
- $\mathsf{Mac}_k(m) \to t$ genera un tag valido para el mensaje m.
- $\mathsf{Vrfy}_k(m,t)$: output 1 si el tag t es válido para m, 0 en otro caso.

Se debe cumplir que para toda llave k y para todo mensaje m si $t \leftarrow \mathsf{Mac}_k(m)$, entonces $\mathsf{Vrfy}_k(m,t) = 1$. En términos de seguridad queremos que ningún adversario que observe y modifique el canal de comunicación, pueda computar un tag válido para algún mensaje de su elección. Para modelar esta situación, describiremos un experimento en el cual el adversario tiene acceso a un oráculo Mac que puede ser consultado para cualquier mensaje que el adversario quiera. El adversario gana si puede generar un tag válido para un mensaje que no fue consultado al oráculo.

Definición:

Definición 3. Π *es existencialmente infalsificable bajo ataques de mensajes adaptivamente escogidos (o simplemente seguro) si para todo adversario probabilista de tiempo polinomial* $\mathcal{A}$*, existe una función negligible* $\mathsf{negl}(\cdot)$ *tal que*

$$\Pr[\mathsf{Exp}^{mac-forge}_{\mathcal{A},\Pi} = 1] \leq \mathsf{negl}$$

Construcción para mesajes de largo fijo: Sea F una PRF con imagen en $\{0,1\}^n$. $\mathsf{Mac}_k(m)$:output $F_k(m)$, $\mathsf{Vrfy}_k(m,t)$: output 1 si $F_k(,m) = t$.

Seguridad: si construcción no es segura entonces adversario puede computar $F_k(m)$, pero dado que F es pseudoaleatoria, esto solo puede pasar con probabilidad 2^{-n}.

Ejercicio 4. *Demostrar seguridad formalmente.*