Laboratorio:
Criptografía y seguridad
Práctica 3

Andrea Itzel González Vargas
Carlos Gerardo Acosta Hernández

Facultad de Ciencias UNAM

Preguntas

1. Describe los tipos de ataques pasivos y activos y las soluciones que aplicarías para evitar cada tipo de ataque.

- **Ataque pasivo**
  Un ataque pasivo en un sistema criptográfico es aquel en el que el criptoanalista o atacante no interviene con los mecanismos involucrados en el sistema, es decir, no modifica ni altera la información que se busca proteger. Este tipo de ataque tiene por objetivo perpetrar en el sistema utilizando únicamente la observación y monitoreo de los datos transmitidos por el canal de comunicación.

- **Ataque activo**
  Son aquellos que afectan la operación de un sistema criptográfico mediante la modificación del flujo de datos transmitido o la suplantación de éste por un flujo falso.

Detectar un ataque pasivo puede resultar muy dificil por su naturaleza. Sin embargo, si no podemos asegurar cuando alguien observe nuestra información, podemos acudir a los algoritmos de cifrado de información, de esa manera no tendrá relevancia si nuestro canal de comunicación es susceptible a las intervenciones pasivas, sino que la seguridad del sistema dependerá de nuestro esquema de cifrado.

Por el lado de los ataques activos, al alterar la información, es más probable que sean detectados. En el caso de *suplantaciones de identidad*, puede manejarse con un sistema de autenticación de dos pasos; si se trata de una *modificación de mensajes*, pueden verificarse la integridad de los datos mediante algún mecanismo como las funciones de *hash* y de esa manera asegurar que el mensaje que esperábamos no ha sido alterado; finalmente, si se trata de una *replicación de mensajes* legítimos en el sistema, se puede establecer un límite de recepciones provenientes de cada nodo transmisor y de esa forma detectar un ataque y posiblemente al atacante.

2. ¿Cuál es la diferencia más importante entre un algoritmo de cifrado y un algoritmo de autenticación?

Que un algoritmo de autenticación, por definición, no conserva la información completa de la entrada en la salida que devuelve. Es decir, que a partir de un elemento de la imagen de la función *hash*, no es posible recuperar el elemento del dominio del que proviene contando únicamente con la salida; mientras que en un algoritmo de cifrado, debe ser posible recuperar el texto claro a partir de su criptotexto correspondiente.

3. ¿Cuáles son las diferencias entre las funciones de hash y las funciones de hash seguras?

Las funciones de hash en general buscan minimizar el número de colisiones posibles, sin embargo las funciones de hash seguras tienen requerimientos más estrictos que las otras funciones menos seguras no necesariamente cumplen:
Sea $H(m)$ una función de hash segura.

- Debe de ser muy difícil [1] encontrar cualesquiera dos mensajes m, m' tales que $H(m) = H(m')$.
- Dado un hash h, debe de ser muy difícil encontrar cualquier mensaje m tal que $h = H(m)$.
- Dado un mensaje m debe de ser muy difícil encontrar otro mensaje m' tal que $H(m) = H(m')$.

4. Supongamos que H(m) es una función hash resistente a colisiones que mapea un mensaje de una longitud de bits arbitraria en un valor hash de n bits. ¿Es cierto que para todos los mensajes x, x' con x $\neq$ de x', tenemos ¿H(x) $\neq$ H(x')? Argumenta tu respuesta.

Sea M el conjunto de mensajes y H el conjunto de todos los hashes posibles con $H(m)$ tal que $m \in M$, supongamos que la longitud l_m de los mensajes m es a lo más n, i.e. $0 \leq l_m \leq n$. Entonces sabemos que $|H| = 2^n$ y $|M| = \sum_{i=0}^{n} 2^i$, por lo tanto $|M| > |H|$. Sin embargo, sabemos que l_m puede ser mayor a n, por lo que $|M|$ es mucho mayor a $|H|$. Podemos ver entonces que como el dominio de $H(m)$ es mayor a la imagen, necesariamente sucede que $\exists x, x' \in M$ tales que $H(x) = H(x')$, por el *principio del palomar*, o sea que el que una función sea resistente a colisiones no significa que no existan mensajes que causen colisiones, es más, estamos seguros de que existen dichas colisiones, sin embargo la función debe de asegurar que el encontrarlas sea muy difícil.

5. ¿Qué es "birthday attack"? ¿Cómo se puede evitar? ¿Qué propiedad de las funciones hash seguras protege contra este tipo de ataques?

Es un ataque basado en la *Paradoja del cumpleaños* de la Teoría de probabilidad. Este problema describe un escenario en el que dado un grupo de personas, encontrar a dos individuos que comparten cumpleaños no es tan poco probable, contrario a la intuición inmediata. De hecho, con tan

[1] Por "muy difícil" se entiende que no sea factible encontrar la respuesta con cualquier algoritmo que corra en tiempo polinomial.

sólo 23 personas, se tiene una probabilidad del 50 % y con 70 personas del 99 %. Este problema trasladado a criptología, refiere a la probabilidad de que dos conjuntos de datos distintos tengan una misma imagen dada una función de *hash*. Sea H el número de posibles salidas de una función de hash f. El número de entradas necesarias para encontrar la primer colisión en f, es aproximado por

$$Q(H) \approx \sqrt{\frac{\pi}{2}H} \tag{1}$$

Por ejemplo, en una función de *hash* de 64 bits, $H = 1{,}8 \times 10^{19}$, entonces $Q(H) \approx 5{,}38 \times 10^9$ (5 billones de entradas) serían necesarios para generar una colisión por fuerza bruta. A esto se le conoce como la *cota del cumpleaños*. Dicho esto, es posible evitar la vulnerabilidad a este ataque, con un *hash* de salida lo suficientemente grande para que sea inviable computacionalmente encontrar una colisión. Precisamente es el objetivo de las funciones de *hash* criptográficas definir un tamaño de bloque suficientemente largo para que sea muy difícil encontrar una colisión lo que los protege de este tipo de ataques, sin embargo, por el principio del palomar, es seguro que existan colisiones; además, para algunas funciones como *MD* y *SHA*, cuyas imágenes no están distribuidas uniformemente, una colisión puede encontrarse más prontamente por un ataque de este estilo.

Kerberos

6. ¿Cómo evita kerberos los ataques de repetición?

El principal problema que considera *Kerberos* en potencialmente ser propenso a un ataque de repetición es la duración de los *tickets* proveedores. Tanto el *ticket proveedor de tickets*[2] -pues debe tener una duración considerable por la naturaleza de su definición: ser reutilizable en la solicitud de múltiples *tickets* de servicio-, como el *ticket* proveedor de servicios. Específicamente cuando el tiempo de vida de tales *tickets* es largo (en el orden de horas, por ejemplo), un posible atacante tendría oportunidad de capturar una copia de alguno de ellos. En caso de ser el primer *ticket* mencionado, podría esperar a que el usuario terminase su sesión, luego falsificar la dirección de red del usuario y mandar el mensaje[3] al *servidor emisor de tickets* (**TGS**), accediendo de tal forma a todos los servicios del usuario legítimo, sin necesidad de mayor información. En caso de que el atacante obtenga el segundo *ticket*, tendría acceso al servicio correspondiente, que continúa representando un peligro para la seguridad del sistema.

Para resolver esto es necesario el siguiente requerimiento en el sistema: Un servicio de red debe ser capaz de asegurar que el usuario que está haciendo uso de un *ticket* es el mismo usuario para el que el fue creado tal *ticket*. Kerberos logra esto, mediante el *servicio de autentificación* (**AS**), que provee tanto al **TGS** como al cliente una *llave de cifrado o de sesión* que sólo ellos conocerán, que al menos el cliente deberá presentar ante el **TGS** para probar su identidad.

La mecánica para la distribución de dichas llaves de sesión es la siguiente:
El cliente manda un mensaje al **AS** solicitando acceso al **TGS**. El **AS** responde con un mensaje

[2] Nuestra terrible traducción para *Ticket-granting ticket*
[3] $ID_c||ID_v||Ticket_{tgs}$

que contiene el ticket y además una copia de la *llave de sesión*; al estar cifrado con una llave derivada de la contraseña del usuario, sólo el cliente de dicho usuario puede descifrar el contenido. Esa misma llave está incluída en el ticket, por lo que sólo el **TGS** puede leerla. De esa manera, ambas partes obtienen su copia de la llave de manera segura.

Con esta adecuación, el cliente puede presentarse ante el **TGS** sin que nos preocupemos por un ataque de repetición:
El cliente manda un mensaje al **TGS** con el *ticket* y el *ID* del servicio al que quiere acceso, además de un *autentificador*, cifrado con la llave de sesión provista por el **AS**, que contiene el *ID* y la dirección del usuario, junto con un *timestamp*. El **TGS** procede entonces a descifrar la información del *ticket* con la llave que comparte con el **AS**, de esa forma obtiene la llave de sesión e intenta descifrar el *autentificador*. Si tiene éxito, sólo restará revisar el nombre y la dirección contenidos en el *autentificador* y compararlos con la información del *ticket* recibido y la dirección de donde procede el mensaje del cliente. Si coinciden, el **TGS** sabrá que tanto el cliente que hace uso del *ticket* es efectivamente su dueño original.

Dado que el *autentificador* sólo puede ser utilizado una ocasión y tiene un tiempo de vida corto, la amenaza de que un atacante lo robe y también obtenga una copia del *ticket* es contrarrestada.

7. ¿De qué manera kerberos asegura a un usuario que el servidor al que se está conectando, es un servidor auténtico?

8. ¿Qué sucede si el servidor TGS es sustituido por un servidor de un hacker?