VÍDEO intypedia014es

LECCIÓN 14: FUNCIONES UNIDIRECCIONALES Y HASH

AUTOR: Dr. Hugo Krawczyk

IBM, Estados Unidos

BERNARDO

Hola, bienvenidos a intypedia. Hoy vamos a estudiar el interesante mundo de las funciones unidireccionales y su utilidad en la criptografía. ¡Acompáñanos!

ESCENA 1. FUNCIONES UNIDIRECCIONALES

ALICIA

Hola Bernardo. No sé si te has dado cuenta, pero habitualmente los algoritmos criptográficos se basan en procedimientos que son fáciles de calcular en una dirección pero muy difíciles de invertir si no se conoce una trampa…

BERNARDO

Perdona Alicia, no entiendo bien lo que quieres decir.

ALICIA

Imagina, por ejemplo, esta situación en criptografía simétrica. El emisor puede generar un texto cifrado a partir de un texto en claro porque conoce la clave de cifrado, mientras que un atacante no puede invertir el proceso fácilmente y recuperar el texto en claro o la clave partiendo sólo del texto cifrado.

BERNARDO

Ah… es eso. Si no recuerdo mal, un atacante no debería poder deducir la clave incluso aunque tuviera una gran cantidad de parejas texto en claro / texto cifrado. Siendo puristas, no es que sea matemáticamente imposible invertir el proceso de cifrado y encontrar la clave, en realidad esta tarea sería computacionalmente muy costosa de realizar.

Alicia, dame otro ejemplo de funciones en criptografía que sean fáciles de calcular en una dirección pero computacionalmente costosas en la dirección contraria.

ALICIA

Por supuesto. Un excelente ejemplo es la criptografía de clave pública. En ésta se generan un par de claves, pública y privada, pero dada la clave pública resulta computacionalmente muy costoso obtener la clave privada sin una información adicional o trampa. Del mismo modo, en la firma digital es muy fácil verificar una firma, pero resulta computacionalmente complejo falsificarla sin conocer la clave privada del firmante.

Este paradigma de "fácil de calcular difícil invertir" es tan común en criptografía que las funciones que tienen esta propiedad son denominadas funciones unidireccionales o de una sola dirección.

BERNARDO

Muy interesante. Por tanto, una función unidireccional es una función matemática en la cual se conoce un procedimiento de cálculo eficiente y rápido para computar esa función, mientras que no se conoce un procedimiento eficiente para realizar ese mismo cálculo pero a la inversa, ¿no?

ALICIA

Correcto, lo has definido muy bien. Existen muchos ejemplos de funciones unidireccionales. Uno famoso es el problema de la factorización entera: multiplicar dos primos grandes de centenas o miles de bits es viable y fácil para los ordenadores actuales; sin embargo, no se conoce un algoritmo eficiente para invertir esta multiplicación, es decir, recuperar los dos primos partiendo exclusivamente de su producto.

BERNARDO

¿No es precisamente esa la función unidireccional sobre la que se basa la seguridad del criptosistema RSA?

ALICIA

Exacto. En cualquier caso, ya hablaremos de RSA y de otras funciones unidireccionales, como el problema del logaritmo discreto, en futuras lecciones.

El tema de las funciones unidireccionales es una de las preguntas abiertas más importantes en computación teórica y una ciencia muy práctica en el campo de la criptografía. De momento, no se tienen pruebas matemáticas de la existencia de funciones de una dirección, pero

nosotros conjeturamos que tales funciones existen y usamos algunos candidatos razonables en la construcción de algoritmos criptográficos.

BERNARDO

¡Vaya!, esto parece interesante. ¿Podrías darme más ejemplos reales donde las funciones unidireccionales tienen utilidad?

ALICIA

Claro Bernardo, hay muchas. Una de mis favoritas es la protección de claves. Cuando tú envías tu clave a una máquina remota, por ejemplo a tu banco, ¿cómo crees que hace el banco para verificar tu clave?

BERNARDO

Bueno... supongo que el banco tiene almacenada una copia de mi clave.

ALICIA

Sí, es una forma de hacerlo, pero en la práctica hay opciones más seguras. El banco no almacena una copia en claro de la clave sino el resultado de aplicar una función unidireccional a la misma, una especie de huella llamada resumen o hash. Más tarde, cuando el usuario envía su clave, el banco aplica esta función a la información recibida y la valida respecto a la almacenada. ¿Entiendes por qué esto es mejor?

BERNARDO

Pues creo que sí. De esta manera si alguien accede a esa base de datos, sólo podrá encontrar ese hash que supongo es difícil de invertir. ¡Qué astutos! Me quedo con la curiosidad de saber más sobre las funciones hash.

ALICIA

No te preocupes, vamos a verlo en el siguiente capítulo.

ESCENA 2. FUNCIONES HASH

ALICIA

Bernardo, imagina por un momento que un amigo tuyo quiere enviarte a través de Internet un fichero enorme, el cual es demasiado grande para adjuntarlo por mail. En su lugar, él lo sube a alguna Web pública. Sin embargo, estarás conmigo que alguien podría cambiar el fichero sin que tú te dieras cuenta.

BERNARDO

Ya, pero eso sería fácil de solucionar. Yo le diría a mi amigo que lo firmara; si yo tengo su clave pública, puedo verificar que es de él.

ALICIA

Correcto, eso lo vimos en la Lección 3. Pero existen soluciones todavía más sencillas. Por ejemplo, tu amigo podría tomar ese fichero grande que quiere enviar y utilizando una función hash, calcular un valor basado en él, por ejemplo de 20 caracteres alfanuméricos, al que nosotros llamaremos resumen del fichero. A continuación tu amigo te enviaría por email o por conversación telefónica este valor. Con esta información, únicamente tendrías que comprobar que el resumen del fichero descargado coincide con el que te envió tu amigo. De esta forma reducimos el problema de verificar la integridad de millones de bits a la integridad de sólo unos cientos de ellos. De hecho, este método se usa en ocasiones para distribuir claves públicas largas, actualizaciones de software, etc.

BERNARDO

Así que si alguien quiere reemplazar el fichero de mi amigo por otro diferente y no ser detectado, necesitaría encontrar un fichero cuyo resumen coincida con el del fichero original, pero como la función resumen es de una sola dirección, no podría encontrar ese segundo fichero. ¿Esa es la idea?

ALICIA

En efecto Bernardo, has dado en el clavo. En criptografía a este tipo de funciones unidireccionales que generan resúmenes se les denomina funciones hash. Puedes pensar en ellas como funciones que convierten una cantidad de bits de tamaño arbitrario a un conjunto reducido de bits denominado resumen, típicamente 128 ó 160 bits. Por tanto, como la salida es reducida y la entrada puede ser muy grande, existen muchas entradas diferentes que darán el mismo resumen. En criptografía, a estas entradas que dan el mismo resumen se les denomina pre-imágenes. Una buena función hash debe ser resistente a pre-imagen, es decir, debe ser computacionalmente inabordable que un atacante deduzca fácilmente las diferentes entradas que producen una misma salida.

BERNARDO

Me parece muy bien.

ALICIA

Sí, pero no sólo eso, sino que además deben ser resistentes a ataques de segunda pre-imagen, es decir, debe ser computacionalmente inabordable que partiendo de un mensaje sea posible encontrar otro que sea modificación del primero y cuyos resúmenes coincidan.

Por tanto, si una función hash es resistente a una segunda pre-imagen entonces la función unidireccional lo seguirá siendo incluso aunque se conozca la relación de que un mensaje deriva en un resumen concreto. Esto es exactamente la medida de seguridad que se necesita

en la aplicación que mencioné anteriormente, en la cual tu amigo te enviaba un fichero a través de una Web pública.

Otro tema importante es el de la resistencia a colisiones, que abordaremos a continuación.

ESCENA 3. RESISTENCIA A COLISIONES

BERNARDO

Alicia, la verdad es que las funciones hash parecen ser un claro ejemplo de la utilidad de las funciones unidireccionales en la práctica.

ALICIA

Así es. Por ejemplo, la idea de generar resúmenes de ficheros es muy útil criptográficamente para firmar ficheros o mensajes, y por tanto garantizar la autoría de ese contenido. En la práctica, por eficiencia en lugar de aplicar la firma digital al mensaje completo se realiza sólo al resumen generado. Como el atacante no es capaz de encontrar diferentes mensajes que den el mismo resumen, entonces firmar el resumen es tan válido como firmar el documento entero.

BERNARDO

Ahora veo la utilidad de que las funciones hash sean resistentes a segunda pre-imagen. Esto facilita firmas en documentos de tamaño arbitrario.

ALICIA

Sí, es una de sus utilidades. No obstante, incluso si la función es resistente a segunda pre-imagen, hay otro ataque posible. Supongamos que contratas a un amigo para hacer un trabajo y firmáis un documento en el que dice que le pagarás 5.000 Euros. Misteriosamente, al mes siguiente tu amigo vuelve con el contrato firmado por ti en el que se indica que le pagarás 50.000 Euros por ese mismo trabajo. Ahora la pregunta: ¿cómo crees que fue esto posible?

BERNARDO

Pues no lo sé,... sólo sería posible si el contrato original y el modificado tienen el mismo resumen. Pero, ¿no se suponía que un atacante no puede hacer esto si la función hash implementa las medidas oportunas?

ALICIA

Bernardo, pensemos en el caso que tu amigo ha generado los dos contratos, antes de que los firmaras, y que ambos tienen el mismo resumen. Lo que tenemos que evitar es que sea computacionalmente abordable calcular dos mensajes que generen el mismo resumen (es decir, una colisión) para la misma función hash.

Por tanto, las funciones hash deben ser resistentes a colisiones. Es una propiedad criptográfica fundamental en muchas aplicaciones, por ejemplo, en la creación de esquemas de firma digital.

ESCENA 4. CONSIDERACIONES PRÁCTICAS: PROCEDIMIENTOS, CRIPTOANÁLISIS Y CONSECUENCIAS

BERNARDO

Bien Alicia, con lo que me has explicado ahora tengo la duda de cómo se las ingenian los matemáticos y criptógrafos para construir estas funciones unidireccionales.

ALICIA

Esa es una pregunta excelente, y como toda buena pregunta no es de fácil respuesta. En la práctica se reutiliza mucho del conocimiento existente en la construcción de algoritmos criptográficos, y en particular de la construcción de algoritmos de cifrado en bloque. Por ejemplo, aquí también se persiguen que entradas muy parecidas a nivel de bits generen salidas completamente diferentes y no correladas. Por ejemplo, en cifradores de bloque de n vueltas modificar un solo bit puede hacer que en una pasada del algoritmo cambien el 50% de los bits resultantes.

Sin embargo, hay una gran diferencia entre funciones hash (unidireccionales) y funciones de cifrado. El objetivo de las primeras no es proteger un secreto o la confidencialidad. Además lo habitual es que las funciones hash no necesiten del uso de claves secretas. Esta es una de las características que hacen que su diseño sea más difícil. No obstante, sí comparten la filosofía de que los algoritmos actuales son públicos y por tanto el atacante intentará vencer al algoritmo por este hecho.

BERNARDO

Buff... no parece nada sencillo construir una función hash segura.

ALICIA

De hecho, el diseño de funciones hash se ha ido renovando gracias a las propuestas de ataque de criptoanalistas. Por ejemplo, los algoritmos estándar de funciones hash desarrollados y utilizados desde la década de los 90, como MD5 o SHA-1, hoy en día o están rotos o significativamente amenazados.

BERNARDO

¿Qué quieres decir con que están rotos?

ALICIA

Pues básicamente que cuando se descubre una pareja de mensajes que generan el mismo resumen, la función hash ya no es resistente a colisiones. Esto le sucedió al algoritmo MD5 en 2004. Incluso esto es peor cuando se demuestran métodos para producir muchas colisiones de este tipo.

BERNARDO

¿Y este descubrimiento tuvo implicaciones prácticas serias?

ALICIA

Sí. Dado que una utilidad importante de las funciones hash está en las firmas digitales, encontrar colisiones puede facilitar falsificar firmas digitales. Es decir, dos mensajes distintos que tengan un mismo resumen, tienen además la misma firma y un atacante puede reemplazar un mensaje por el otro. Un ejemplo de este ataque fue demostrado en 2008 por un grupo de investigadores en Amsterdam los cuales falsificaron un certificado raíz mediante el uso de un ataque de colisión contra el algoritmo MD5.

BERNARDO

¿Se falsificó un certificado raíz? ¿Quieres decir que ellos podrían utilizarlo en cualquier Web?

ALICIA

En efecto. Los certificados digitales actúan como tarjetas de identificación para las Webs y este ataque permitió falsificar muchos de estos certificados.

BERNARDO

¿Y todo esto porque se demostró que MD5 no es resistente a colisiones?

ALICIA

Pues sí, para que veas que esto es un asunto serio.

BERNARDO

De este tipo de ataques, ¿qué podemos hacer para protegernos?

ALICIA

Tranquilidad Bernardo, hay varias líneas de defensa. Algunas son administrativas y permiten a Autoridades de Certificación hacer que estos ataques sean más difíciles. Por ejemplo, introducir números de serie aleatorios a los certificados digitales hace los ataques menos prácticos a corto plazo.

Por otro lado, utilizar funciones hash más seguras, como la familia SHA2, es sin duda la mejor solución de momento, aunque esto requiere más tiempo. Mientras tanto, los criptógrafos ya están analizando nuevas funciones hash mediante una competición internacional en curso que

está a punto de anunciar un nuevo estándar mundial, como sucedió hace más de diez años con el cambio del estándar de cifrado simétrico DES por el AES.

Otras formas de mejorar la seguridad, especialmente en firmas digitales, consiste en trabajar en procedimientos que no dependan de la resistencia a colisión. Por ejemplo, el uso de variables aleatorias en el proceso de generación del resumen puede producir esquemas de firma digital que no dependan de la resistencia a colisión.

BERNARDO

Me alegra saber que los criptógrafos no se aburren…

ALICIA

Ellos siempre tienen trabajo… y no duermen muy bien. Bueno yo creo que por hoy es más que suficiente. En la página Web de intypedia tienes información adicional de esta lección. ¡Hasta luego!

BERNARDO

¡Adiós!

Guión adaptado al formato intypedia del documento desarrollado por el Dr. Hugo Krawczyk, de IBM Estados Unidos.

Madrid, España, mayo de 2012

http://www.intypedia.com

http://twitter.com/intypedia