UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELECTRICA
ESTRUCTURAS ABSTRACTAS DE DATOS Y ALGORITMOS
PARA INGENIERÍA
REPORTE LIBRERÍA CRIPTOGRÁFICA EN C++
GEYKE93

ESCUELA DE INGENIERIA ELECTRICA
UNIVERSIDAD DE COSTA RICA

GABRIEL JIMÉNEZ LEIVA B23466
CARLA VEGA JARQUÍN B06763

1 Introducción

La aparición de la informática y el uso masivo de las comunicaciones digitales han producido un número creciente de problemas de seguridad. Las transacciones que se realizan a través de la red pueden ser interceptadas. La seguridad de esta información debe garantizarse. Este desafío lleva al estudio de los algoritmos, protocolos y sistemas que se utilizan para proteger la información y dotar de seguridad a las comunicaciones y a las entidades que se comunican. Los criptógrafos investigan, desarrollan y aprovechan técnicas matemáticas que les sirven como herramientas para conseguir sus objetivos. Los estudios y métodos de la criptología, criptografía y criptoanálisis datan desde los tiempos de Julio César, León Battista Alberti ("padre de la criptografía") en el Siglo XV. Esta tuvo un gran desarrollo en el período entreguerras. Existen muchos métodos, desde los clásicos de transposición, ROT13, etc; hasta los algoritmos más utilizados y conocidos que se mencionan más adelante. GEYKE93 busca ser una librería "bastante completa" al incorporar un método sobresaliente por cada campo. Estos son, RSA en criptografía asimétrica, AES en criptografía simétrica y MD5, SHA-2 como contribuciones a hash functions (funciones de relleno, one-way, etc).

2 Desarrollo

2.1 RSA

En criptografía, RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente. Este sistema de encriptación asimétrica junto al cifrado de clave pública fue creado por Ronald Linn Rivest, nacido en 1947 en Schenectady, (Nueva York) quien es criptógrafo y profesor de ciencias de la computación en el departamento de ingeniería eléctrica y ciencias de la computación del MIT en colaboración de Len Adleman y Adi Shamir. Este algoritmo RSA los hizo ganar el Premio Turing en el año 2002.

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10^{200}, y se prevé que su tamaño crezca con el aumento de la capacidad de cálculo de los ordenadores. Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada. Se cree que RSA será seguro mientras no se conozcan formas rápidas de descomponer un número grande en producto de primos. La computación cuántica podría proveer de una solución a este problema de factorización.

El algoritmo consta de tres pasos: generación de claves, cifrado y descifrado.

- Generación de claves

  1. Cada usuario elige dos números primos distintos p y q .
  2. Se calcula n=pq
  3. Se calcula $\varphi(n) = (p-1)(q-1)$, donde φ es la función φ de Euler.
  4. Se escoge un entero positivo e menor que $\varphi(n)$, que sea coprimo con $\varphi(n)$.

5. Se determina un d (mediante aritmética modular) que satisfaga la congruencia $d = e^{-1} \mod \varphi(n)$, es decir, que d sea el multiplicador modular inverso de $emod\varphi(n)$ La clave pública es (n,e), esto es, el módulo y el exponente de cifrado. La clave privada es (n,d), esto es, el módulo y el exponente de descifrado, que debe mantenerse en secreto.

- Cifrado $c = m^e$
- Descifrado $m = c^d$

Seguridad

RSA hace uso del rellenado aleatorio dentro del valor de m antes del cifrado para evitar ataques. Esta técnica asegura que m no caerá en el rango de textos sin cifrar inseguros, y que dado un mensaje, una vez que este rellenado, cifrará uno de los números grandes de los posibles textos cifrados. La última característica es la incrementación del diccionario haciendo este intratable a la hora de realizar un ataque. Algunos esquemas de rellenado (padding scheme) RSA son:

- *RSA-OAEP* (Optimal Asymetric Encryption Padding) o su versión moficada *RSA-OAEP+*.
- *RSA-SAEP+* (Simplified Asymmetric Encryption Padding)
- *RSA-REACT*
- *RSA-PSS* (Probabilistic Signature Scheme).

El descifrado completo de un texto cifrado con RSA es computacionalmente intratable, no se ha encontrado un algoritmo eficiente todavía para ambos problemas. Proveyendo la seguridad contra el descifrado parcial podría requerir la adición de una seguridad padding scheme.

Autenticación con RSA

RSA puede también ser usado para autenticar un mensaje. Se debe observar que la seguridad de los padding-schemes como RSA-PSS son esenciales tanto para la seguridad de la firma como para el cifrado de mensajes, y que nunca se debería usar la misma clave para propósitos de cifrado y de autentificación.

2.2 AES

Advanced Encryption Standard (AES), también conocido como Rijndael (pronunciado "Rain Doll" en inglés), es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. El AES fue anunciado por el Instituto Nacional de Estándares y Tecnología (NIST) como FIPS PUB 197 de los Estados Unidos (FIPS 197) el 26 de noviembre de 2001 después de un proceso de estandarización que duró 5 años. Se transformó en un estándar efectivo el 26 de mayo de 2002. Desde 2006, el AES es uno de los algoritmos más populares usados en criptografía simétrica. El cifrado fue desarrollado por dos criptólogos belgas, Joan Daemen y Vincent Rijmen, ambos estudiantes de la Katholieke Universiteit Leuven, y enviado al proceso de selección AES bajo el nombre "Rijndael".

Algoritmo

Key Expansion: AES encripta por rondas, el numero de rondas depende de la longitud de la llave, y cada ronda utiliza una llave diferente. Estas llaves son generadas a partir de la inicial, es decir la brindada por el usuario o establecida dentro del alogirtmo, a esta generacion de llaves le vamos a llamar Key Expansion. Suponiendo que se ordenan los bytes de la llave en una matriz cuadrada, la key expansion funciona asi:

- Se toma la ultima columna de la llave de ronda anterior y se mueve el bit de mas arriba para la ultima casilla
- Se pasa cada byte de esta columna por una matriz de substitucion

- Aplica 'xor' (operacion binaria OR (v)exclusivo) a la columna con una "constante de ronda", diferente para cada ronda
- Finalmente aplica 'xor' a la columna con la primera columna de la llave de ronda anterior
- Para el resto de columnas, aplica 'xor' a la columna anterior con la misma columna de la llave de ronda anterior. Ejemplo: para generar la columna 3 en la ronda 4, hace 'xor' a la columna 2 ronda 4 con la columna 3 ronda 3.

Rondas: Supongamos que estamos trabajando con una llave de 128 bits. El mensaje a encriptar se rompe en bloques de 16 bytes, la llave es tambien un bloque de 16 bytes, estos bloques los vamos a ver como matrices 4x4. La primera ronda solamente hace 'xor' a cada byte del bloque del mensaje con los de la llave. Rondas intermedias:

- Substitucion de bytes: pasa cada byte del bloque por una matriz de substitucion que lo transforma en un byte diferente
- "Shift rows": hace una permutacion en los bytes del bloque, cambiandolos de lugar
- Mezcla de columnas: agarra cada columna y mezcla los bits de ella
- Al final, aplica la siguiente llave de ronda con un xor
- En la ultima ronda no aplica la parte de mezcla de columnas

El descifrado funciona de manera inversa, la ultima ronda se convierte en la primera, y así sucesivamente. Los pasos de cada ronda se aplican inversamente.

2.3 Función hash (o resumen)

A las funciones hash (adopción más o menos directa del término inglés hash function) también se les llama funciones picadillo, funciones resumen o funciones de digest (adopción más o menos directa del término inglés equivalente digest function). Una función hash H es una función computable mediante un algoritmo, $H : U \to M$
$x \to h(x)$, que tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M.

Al conjunto U se le llama dominio de la función hash. A un elemento de U se le llama preimagen o dependiendo del contexto clave o mensaje. Al conjunto M se le llama imagen de la función hash. A un elemento de M se le llama valor hash, código hash o simplemente hash.

Parámetros adicionales

La calidad de una función hash viene definida con base en la satisfacción de ciertas propiedades deseables en el contexto en el que se va a usar.

1. Bajo costo: Calcular el valor hash necesita poco costo (computacional, de memoria...).
2. Compresión: Una función hash comprime datos si puede mapear un dominio con datos de longitud muy grande a datos con longitud más pequeña.
3. Uniforme: Se dice que una función hash es uniforme cuando para una clave elegida aleatoriamente es igualmente probable tener un valor hash determinado, independientemente de cualquier otro elemento.

Para una función hash H uniforme del tipo $H : \{0,1\} m \to \{0,1\} n$, es decir:

- Las cadenas están construidas sobre un alfabeto de 2 símbolos (Alfabeto binario)

- El dominio es el conjunto de las cadenas de longitud m
- El rango es el conjunto de las cadenas de longitud n

Se puede decir que a cada resumen le corresponde 2m-n mensajes y que la probabilidad de que dos mensajes den como resultado la misma salida es 2-n. Para algoritmos de búsqueda, si todas las entradas son igualmente probables, se busca esta propiedad para minimizar el número de colisiones ya que cuantas más colisiones haya, será mayor el tiempo de ejecución de las búsquedas.

Una función hash criptográfica se define como una serie de operaciones sobre un mensaje de entrada de tamaño arbitrario, que producen una salida de tamaño fijo (hash o resumen criptográfico del mensaje) de forma que un cambio en el mensaje no pueda pasar inadvertido. Debe ser sencillo calcular un hash a partir de un mensaje, pero dado un hash debería ser computacionalmente inviable encontrar un mensaje que lo produzca. Además, dado un mensaje debería ser inviable encontrar un segundo mensaje con el mismo hash y, como ya he dicho, debería ser inviable modificar un mensaje sin que se modifique su hash.

Por tanto, las propiedades que una función hash debe cumplir son:

- Resistencia pre-imagen: dado un hash h, debe ser inviable encontrar un mensaje m tal que h=hash(m). De lo contrario, la función hash sería vulnerable a ataques de pre-imagen.
- Resistencia a segunda pre-imagen: dado un mensaje debería ser inviable encontrar un segundo mensaje, , que produzca el mismo hash. Es decir, dado debería ser difícil encontrar tal que . En caso contrario, se dice que la función es vulnerable a ataques de segunda pre-imagen.
- Resistencia a colisiones: Debería ser difícil encontrar dos mensajes con el mismo hash. Obviamente, dado una función hash con una salida de n bits, si probamos mensajes seguro que encontraremos dos con el mismo hash. La teoría acerca de los ataques de cumpleaños nos dice que para un hash de n bits deberíamos probar unos mensajes distintos para encontrar una colisión. Este número se conoce como la cota del cumpleaños (en inglés birthday bound).

Estructura típica de una función hash Una función hash consiste típicamente de una función de compresión que toma bloques de datos de un tamaño fijo como entrada y produce bloques de un tamaño fijo (el tamaño de salida de la función hash). Adicionalmente, la salida del bloque previo es mandada como entrada junto con el bloque siguiente, de forma que la salida depende de todos los bloques previos. Si no se hiciera así, el hash solo sería función del último bloque .

2.3.1 MD5

MD5 (Message-Digest Algorithm) es un algoritmo de reduccion (Hash Function) diseñado por Ronald Rivest en 1991 para sustituir al MD4. Este algoritmo no es completamente seguro, se han encontrado vulnerabilidades como colisiones de hash, y de hecho se recomienda utilizar otros algoritmos como los SHA. No obstante, el MD5 es ampliamente utilizado en la actualidad, entre algunas de sus aplicaciones estan la autenticacion de archivos, para comprobar que el archivo no ha sido modificado, tambien en algunos sistemas de UNIX se utiliza para verificar las claves de usuario.

Algoritmo:

Primero el algoritmo rellena el mensaje dado a una longitud congruente con 448 con modulo 512. Es decir que la longitud es 64 bits menos que un entero multiplo de 512. Este relleno consiste de 1 bit seguido de los ceros necesarios. La longitud original del mensaje se almacena en los ultimos 64 bits del relleno.

Se inicializa un buffer que consta de cuatro palabras (128 bits) constantes. Se definen cuatro funciones que utilizan operadores binarios para convertir una entrada de tres palabras en una salida de una palabra. Estas funciones

son: $F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$
$H(X,Y,Z) = (X \oplus Y \oplus Z)$
$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$

La transformacion del texto de entrada al hash final se da a traves de funciones que dentro del programa definimos como FF, GG, HH e II. Estas funciones utilizan a las funciones F, G, H e I, y otras como rotacion de bits. Tambien utilizan las constantes que definimos como A, B, C y D.

2.3.2 SHA2

Hash SHA-2 certificado SSL es un algoritmo criptográfico desarrollado por el Instituto Nacional de Estándares y Tecnología (NIST) y la Agencia de Seguridad Nacional (NSA). SHA2 certificados son más seguros que todos los algoritmos anteriores, y se requiere en determinadas aplicaciones, en lugar de certificados firmados con el SHA-1 que comienza la función hash de 01 de enero 2011. El principal obstáculo para el aumento de la proliferación de SHA-2 es que mientras que SHA-2 certificados son compatibles con la mayoría de los navegadores modernos actualizados, las plataformas, los clientes de correo y dispositivos móviles, algunos sistemas antiguos, como los que ejecutan Windows XP SP2 o más, son incapaces de apoyar SHA2 cifrado.

Construcción Merkle-Damgard: en esta estructura, las funciones hash más populares están basadas en esta construcción. Sin embargo, otras alternativas existen y muchas de las propuestas para el concurso SHA-3 están basadas en otras construcciones.

MD5 y SHA-1 son mucho más populares, pero la estructura de MD5 fue rota hace tiempo, primero por el equipo de la dra. Wang y luego por un grupo de investigadores que incluía al dr. Benne de Weger. SHA-1 es muy similar a MD5 y se cree que padece el mismo tipo de problemas. La familia SHA-2 incluye varias funciones hash con distintos tamaños de salida: SHA-224, SHA-256, SHA-384 y SHA-512, donde el número define el número de bits de salida. SHA-256 y SHA-512 utilizan tamaños de palabra de 32 y 64 bits respectivamente, mientras que SHA-224 y SHA-384 son versiones truncadas de las primeras.

Para el SHA-256, básicamente, el mensaje de entrada se divide en bloques de 512 bits M y se le añade información adicional que incluye la longitud del mensaje original. Para cada uno de estos bloques se ejecuta un message schedule que produce 64 variables. Estas 64 variables son procesadas con una función de compresión.

Tras este procesado, el valor intermedio del hash es obtenido como la suma (módulo 32) de las variables y el valor obtenido en la iteración anterior. Este proceso se ejecuta para cada bloque del mensaje de entrada y al final se obtiene el resumen del mensaje.

En el 2012 se inició un concurso abierto mantenido por NIST para crear un nuevo estándar para funciones hash, SHA-3 que no viene a sustituir el SHA2 pues este no presenta amenaza alguna al día de hoy. El ganador fue Keccak diesñado por Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.

2.4 Librerías

Ver contenidos en carpeta include de geyke93.

1. Crypto++
2. Beecrypt
3. Hashlib

2.4.1 SSH

SSH (Secure SHell, en español: intérprete de órdenes segura) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la

computadora mediante un intérprete de comandos, y también puede redirigir el tráfico de X para poder ejecutar programas gráficos si tenemos un Servidor X (en sistemas Unix y Windows) corriendo. Además de la conexión a otros dispositivos, SSH nos permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a los dispositivos y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

Seguridad: SSH trabaja de forma similar a como se hace con telnet. La diferencia principal es que SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible, evitando que terceras personas puedan descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión; aunque es posible atacar este tipo de sistemas por medio de ataques de REPLAY y manipular así la información entre destinos.

3 Conclusiones

- Se debe reconocer el algoritmo necesario para realizar cierta tarea si no se quiere caer en error, ya que en este trabajo se observa que, por ejemplo, para la criptografía simétrica, no se tiene muy buena escabilidad en poblaciones muy numerosas; y en el caso de la criptografía asimétrica se resuelve ese problema pero puede ser más lento, largo (pues expande el texto) y hay problemas de distribución de la clase.
- Las claves asimétricas se usan para envolver las claves simétricas y protegerlas durante su tránsito, lo mismo que para cirgrar verificaciones (hashes) para crear firmas digitales. De ahí que las librerías trabajen ambos métodos.

References

[1] http://beecrypt.sourceforge.net/doxygen/c++/index.html

[2] http://sourceforge.net/projects/cryptopp/?source=recommended

[3] http://sourceforge.net/projects/hashlib2plus/?source=recommended

[4] http://www.aescrypt.com/

[5] http://es.wikipedia.org/wiki/Ronald_Rivest

[6] http://es.wikipedia.org/wiki/SSH

[7] http://es.wikipedia.org/wiki/RSA

[8] http://es.wikipedia.org/wiki/Funci%C3%B3n_hash

[9] http://www.openssh.org/

[10] mirror.math.ku.edu/tex.../pgfgantt/pgfgantt.pdf

[11] http://cryptopp.com/

[12] http://es.wikipedia.org/wiki/Historia_de_la_criptograf%C3%ADa

[13] http://es.wikipedia.org/wiki/Criptograf%C3%ADa

[14] http://elsofista.blogspot.com/2004/06/por-qu-nos-importan-los-nmeros-primos.html

[15] http://elrincondeliumeg.blogspot.com/2011/03/encriptacion-y-numeros-primos-por-elfio.html

[16] http://primes.utm.edu/mersenne/index.html

[17] http://www.partow.net/programming/hashfunctions/

[18] http://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch03_09.htm

[19] http://www.cs.rit.edu/~ib/Classes/CS482-705_Winter10-11/Slides/crypto_6b.pdf

[20] http://algoritmoshade.blogspot.com/2010/05/encriptacion-rsa.html

[21] http://www.youtube.com/watch?v=8rT1zRD5cjI

[22] http://www.codeproject.com/Questions/602310/ImplementingplusRSAplusCryptographyplusinplusC-2b

[23] http://maxkalavera.blogspot.com/2012/09/algoritmo-rsa.html

[24] http://es.wikipedia.org/wiki/Advanced_Encryption_Standard