LibreIM

Seguridad y Protección de Sistemas Informáticos
Grado en Ingeniería Informática
Universidad de Granada

21 de enero de 2020

Sugerencias para el examen

Fijamos a continuación la notación que seguiremos en este documento.

- Denotamos por ω al primer ordinal infinito, cuya cardinalidad coincide con la de $\mathbb{N}$. Podemos entenderlo como una forma de «ordenar» los números naturales, es decir, $\omega = \{0, 1, 2, \dots\}$. Dado $n \in \omega^* := \omega - \{0\}$, se tiene que $n = \{0, 1, \dots, n-1\}$, que identificamos cuando sea preciso con el número natural n. El orden viene dado por la pertenencia, es decir, $n \leq m$ si y solo si $n \in m$. Es un orden total por construcción.

- $\mathscr{A}$ será un alfabeto: un conjunto finito no vacío de símbolos. El conjunto de expresiones o palabras que podemos formar por yuxtaposición de símbolos de $\mathscr{A}$ lo denotaremos como $\exp(\mathscr{A})$, es decir,

$$\exp(\mathscr{A}) = \bigcup_{k \in \omega} \mathscr{A}^k,$$

  donde $\mathscr{A}^0 = \{\varepsilon\}$ (la palabra vacía) y $\mathscr{A}^k = \mathscr{A}^{k-1}\mathscr{A}$ para $k > 0$. Reservaremos la notación $\exp(\mathscr{A})^*$ para indicar que no se admite la palabra vacía.

- Dado $s \in \exp(\mathscr{A})$, existirá $k \in \omega$ tal que $s \in \mathscr{A}^k$, es decir, $s = s_1 s_2 \cdots s_k$ con $s_i \in \mathscr{A}$ para todo i. Denotamos $s = \langle s_j \rangle_{j \in k}$, y entonces k será la *longitud* de s, denotada por $\mathit{len}(s)$. Cuando la longitud de la palabra no sea relevante, escribiremos simplemente $s = \langle s_j \rangle_j$.

- En ocasiones será necesario poner en correspondencia los símbolos del alfabeto con números naturales. Si $\mathscr{A}$ es un alfabeto de $n \in \omega^*$ símbolos, consideraremos fijada una función biyectiva $f : \mathscr{A} \to n$. De esta forma, si $s \in \exp(\mathscr{A})$ nos referiremos a $\langle f(s_j) \rangle_j$ como $f(s)$.

- Denotaremos el cuerpo finito de n elementos como $\mathrm{GF}(n)$, y el grupo multiplicativo asociado como $\mathrm{GF}(n)^*$.

- Un *criptosistema* será una terna $\langle \mathscr{A}, E, D \rangle$, donde $\mathscr{A}$ es un alfabeto y E (resp. D) es una aplicación de $\exp(\mathscr{A})$ en $\exp(\mathscr{A})$, de forma que $D \circ E$ sea la identidad en $\exp(\mathscr{A})$. Nos referiremos a los elementos de $\exp(\mathscr{A})$ como *texto plano* cuando los veamos como argumentos de E (función de cifrado) y como *texto cifrado* cuando los veamos como argumentos de D (función de descifrado). Es usual que las funciones de cifrado y descifrado dependan de un parámetro llamado *clave*, que en general no tiene que coincidir para las dos.

Cuestión 1. *Cifrado de Vigenère.*

Respuesta. Dado $n \in \omega^*$ y $\mathscr{A}$ un alfabeto de n símbolos, el cifrado de Vigenère es un criptosistema polialfabético que consiste en una clave $\alpha \in \exp(\mathscr{A})^*$ y sendas funciones E_α y D_α para cifrar y descifrar respectivamente. Podemos definir la función $E_\alpha : \exp(\mathscr{A})^* \to \exp(\mathscr{A})^*$ como

$$E_\alpha(s) = \langle f^{-1}((f(s_j) + f((\alpha^{len(s)})_j)) \bmod n)\rangle_j,$$

donde $s = \langle s_j\rangle_j$ y la expresión $\alpha^{len(s)}$ es la yuxtaposición de α consigo misma y eventual truncamiento final hasta conseguir una palabra con la misma longitud que s. Hacemos esto para asegurar la existencia de una letra en cada posición $j \in len(s)$ de α. Por ejemplo, si α = HOLA y s = ESTOPA, entonces $\alpha^{len(s)}$ = HOLAHO. De modo análogo se define $D_\alpha : \exp(\mathscr{A})^* \to \exp(\mathscr{A})^*$ como:

$$D_\alpha(s) = \langle f^{-1}((f(s_j) - f((\alpha^{len(s)})_j)) \bmod n)\rangle_j.$$

Se comprueba que la terna $\langle \mathscr{A}, E_\alpha, D_\alpha\rangle$ es un criptosistema, es decir, $D_\alpha \circ E_\alpha = 1_{\exp(\mathscr{A})^*}$. En efecto, si $s = \langle s_j\rangle_j \in \exp(\mathscr{A})^*$, se tiene que:

$$\begin{aligned}
D_\alpha(E_\alpha(s)) &= D_\alpha(\langle f^{-1}((f(s_j) + f((\alpha^{len(s)})_j)) \bmod n)\rangle_j) \\
&= \langle f^{-1}((f(f^{-1}((f(s_j) + f((\alpha^{len(s)})_j)) \bmod n)) - f((\alpha^{len(s)})_j)) \bmod n)\rangle_j \\
&= \langle f^{-1}(((f(s_j) + f((\alpha^{len(s)})_j)) \bmod n) - f((\alpha^{len(s)})_j)) \bmod n)\rangle_j \\
&= \langle f^{-1}(f(s_j) \bmod n)\rangle_j \\
&= \langle f^{-1}(f(s_j))\rangle_j \\
&= \langle s_j\rangle_j \\
&= s.
\end{aligned}$$

Por último, enunciamos un resultado que nos dice que en realidad ambas funciones son la misma salvo un cambio de clave.

Teorema. *Si $\alpha = \langle \alpha_j\rangle_j$ es una clave de Vigenère, definiendo $\alpha' = \langle(-\alpha_j) \bmod n\rangle_j$ tenemos que $D_\alpha = E_{\alpha'}$.*

La principal debilidad del cifrado Vigenère es la naturaleza repetitiva de su clave. Si un criptoanalista adivina correctamente la longitud de la clave, el texto cifrado puede tratarse como cifrados entrelazados de César, que pueden romperse fácilmente de forma individual.

Cuestión 2. *Explicar la transformación `SubBytes()` que es parte del algoritmo simétrico de cifrado AES.*

Respuesta. El algoritmo de cifrado de AES realiza en cada ronda una serie de transformaciones a nivel de *bytes*. La primera de estas transformaciones es `SubBytes()`,

que consiste en sustituir cada *byte* del estado de forma no lineal e independiente según una tabla de sustitución llamada S-box.

La tabla de sustitución es una matriz cuadrada de orden 16 (no simétrica), que contiene en cada posición (x, y) un *byte* en hexadecimal que representa la transformación del *byte* s_{xy} del estado. La construcción de la tabla se realiza de la siguiente forma, considerando xy como la representación en hexadecimal de un *byte*:

1. Si $xy = 00$, se mantiene igual. Si $xy \neq 00$, se calcula el inverso de xy en $\mathrm{GF}(2^8)$, que será otro *byte* $b = b_7 \cdots b_0$.

2. El bit i-ésimo del *byte* b, b_i, se transformará en b'_i haciendo

$$b'_i = b_i + b_{(i+4) \bmod 8} + b_{(i+5) \bmod 8} + b_{(i+6) \bmod 8} + b_{(i+7) \bmod 8} + c_i, \quad (1)$$

   donde c_i es el i-ésimo bit del *byte* $\{63\}$ ó $\{01100011\}$ para todo $0 \leq i < 8$, entendido como $c = c_7 \cdots c_0$. El *byte* $b' = b'_7 \cdots b'_0$ resultante ocupará la posición (x, y) de la tabla, expresado en hexadecimal. Notamos que el símbolo + representa la suma en $\mathrm{GF}(2)$.

Mencionamos también que es posible expresar el efecto de (1) mediante una transformación afín aplicada a cada *byte* del estado (visto como vector de bits). Dicha transformación es invertible, y a la transformación resultante de aplicar la inversa seguida de tomar el inverso multiplicativo en $\mathrm{GF}(2^8)$ la llamaremos transformación `InvSubBytes()`. Se usará para revertir los efectos de `SubBytes()` a la hora de descifrar.

Nota. Sabemos que $\mathrm{GF}(2^8) \cong \mathbb{Z}_2[x]/m(x)$, donde $m(x) = x^8 + x^4 + x^3 + x + 1$. En consecuencia, podemos expresar cada elemento de $\mathrm{GF}(2^8)$ de alguna de las siguientes formas:

1. Como el único representante de su clase de grado inferior a 8.

2. Por la representación binaria correspondiente a la lista de los coeficientes del representante.

3. Por la representación hexadecimal correspondiente a la representación binaria mencionada anteriormente.

Cuestión 3. *Limitaciones de los sistemas simétricos de cifrado en la comunicación y cómo la criptografía de clave pública los ha resuelto.*

Respuesta. En los criptosistemas de cifrado simétrico, emisor y receptor acuerdan una misma clave k que marca como funcionarán las funciones de cifrado y descifrado E_k y D_k. Este esquema presenta las siguientes limitaciones:

1. Es necesario que se produzca un intercambio inicial de la clave antes de poder establecer la comunicación cifrada. Este es uno de los mayores problemas de la criptografía clásica, pues no siempre es posible garantizar un canal seguro de comunicación para dicho intercambio.

2. Cuando consideramos una red de usuarios que quieren comunicarse entre sí, cada par de usuarios debe tener una clave distinta para maximizar la seguridad. Esto hace que tengamos un ratio de crecimiento del número de claves cuadrático en relación al tamaño de la red, por lo que no es un sistema escalable. Por ejemplo, con 10 usuarios necesitamos 45 claves diferentes, con 100 usuarios necesitamos 4950 claves, y en general para n usuarios necesitamos $n(n-1)/2$ claves distintas. Es por esto que no es fácil añadir un usuario a la red, pues es preciso actualizar toda la base de datos para incluirlo. El gran número de claves provoca también un problema de seguridad, al ser necesario que cada usuario guarde sus claves, y genera problemas logísticos cuando llegue el momento de renovar las claves de todos los usuarios de la red.

3. Estos sistemas también adolecen de falta de crédito, ya que cualquier persona que obtenga la clave compartida por dos usuarios puede formar parte de la comunicación. Si Alice y Bob comparten una clave secreta y se da que Carol la descubre, esta puede interceptar los mensajes de Alice a Bob y modificarlos sin que Bob lo descubra, o suplantar la identidad de Alice mandando mensajes a Bob. También es posible que Bob envíe mensajes falsos y después se encubra en esta última posibilidad para eximirse de las culpas.

La llegada de los criptosistemas de clave pública permite solucionar estos problemas. En este nuevo esquema cada usuario tiene una clave de cifrado pública, que comparte con el resto de usuarios, y una clave de descifrado privada, que solo él o ella conoce.

1. Ya no es necesario que se produzca un intercambio inicial de claves, pues las claves de cifrado de cada usuario son conocidas por todos los demás (el conocimiento de la misma no permite el descifrado).

2. Solo se mantienen dos claves por cada usuario del círculo, lo que elimina los problemas de creación y mantenimiento de la base de datos. Más aún, en la base de datos solo se guarda la clave pública de cada usuario, reduciendo considerablemente el tamaño de esta.

3. Es posible implementar un sistema de comunicación que garantice la *autenticación* de usuarios, la *integridad* de mensajes y la *no repudiabilidad* mediante un protocolo de firma de mensajes. De esta forma, el receptor podrá garantizar que el mensaje no ha sido modificado y que solo puede provenir de la persona que en teoría lo ha enviado. Además, la firma depende del contenido del mensaje, por lo que nadie puede usarla con otro documento.

Cuestión 4. *Explicar los fundamentos de la criptografía de clave pública y las líneas fundamentales de la firma a través de la misma.*

Respuesta. Los sistemas criptográficos de clave pública fueron inventados para el manejo de claves secretas. En estos sistemas, cada usuario usa dos claves: una clave de cifrado k_e (pública) y otra clave de descifrado k_d (privada), que determinan las funciones $E = E_{k_e}$ y $D = D_{k_d}$, respectivamente. Para que el sistema funcione correctamente, debe cumplirse que $D(E(m)) = m$ para todo mensaje m.

Cualquiera que desee mandar un mensaje confidencial a otra persona lo cifrará empleando la función de cifrado asociada a la clave pública de esta persona, que es conocida, confiando en que solo el poseedor de la clave privada asociada podrá descifrarlo. Nótese que ni siquiera el emisor sería capaz de descifrar el mensaje, y que no ha hecho falta realizar un intercambio previo de claves.

Para que el sistema sea práctico es necesario que se puedan calcular $E(m)$ y $D(E(m))$ rápidamente, pero por seguridad debería ser imposible determinar D a partir de E en un tiempo razonable. Esto es lógico, pues en otro caso la confidencialidad de D estaría comprometida al ser E pública.

Teóricamente estos sistemas podrían construirse a partir de una *función unidireccional trampa*. Esta debe ser una función f que sea *fácil* de calcular, pero que su inversa f^{-1} sea *difícil* de calcular sin una información adicional conocida como *trampa*. En el contexto criptográfico dicha trampa estará asociada a la clave privada, de forma que solo el receptor tenga la información necesaria para determinar el algoritmo de descifrado a partir de f. Subrayamos que la existencia de funciones unidireccionales no ha sido demostrada, y de hecho de existir alguna se tendría necesariamente que $P \neq NP$ [1].

Un ejemplo de función cuya unidireccionalidad ha sido conjeturada es $(p, q) \mapsto pq$, donde p y q son primos. El problema de descomponer un número en sus factores primos se conoce como problema de factorización, y es considerado extremadamente difícil a día de hoy. Esta función es la que se utiliza para implementar el criptosistema de clave pública RSA, donde la trampa es, por ejemplo, el conocimiento de la *función phi de Euler* $\Phi(pq)$.

A partir de estos sistemas es posible implementar un protocolo de firma. Por ejemplo, si suponemos que las funciones E y D son biyectivas e inversas la una de la otra, la firma funcionaría del siguiente modo. Sean $\{E_a, D_a\}$ las funciones de cifrado y descifrado de Alice y $\{E_b, D_b\}$ las correspondientes funciones de Bob. Si Alice quiere enviar a Bob un mensaje cifrado y firmado, se procede como sigue:

1. Alice cifra el mensaje con su clave privada, es decir, calcula $D_a(m)$.

2. Alice cifra de nuevo el mensaje, pero esta vez utilizando la clave pública de Bob. Envía a Bob la cadena $E_b(D_a(m))$.

[1] Alan L. Selman. *A survey of one-way functions in complexity theory*. Mathematical systems theory, 25(3):203–221, 1992.

3. Bob recupera $D_a(m)$ aplicando su función privada de descifrado, D_b.

4. Finalmente Bob recupera el texto plano descifrando el mensaje con la clave pública de Alice: $E_a(D_a(m)) = m$. Es aquí donde se emplea la hipótesis extra de que $E \circ D = 1$.

De esta forma se sigue garantizando la confidencialidad, pues solo Bob puede descifrar la primera capa del mensaje mediante su clave privada. Además, puede estar seguro de que el mensaje proviene de Alice (o, al menos, de alguien que posea su clave privada), pues el procedimiento solo tiene éxito si la segunda capa fue cifrada con la clave privada de Alice.

Cuestión 5. *Enumerar resumidamente las precauciones más destacables a tomar al generar un círculo de comunicación basado en RSA.*

Respuesta. Para adherirse a un círculo RSA cada usuario elige dos números primos p y q y dos exponentes e y d de forma que $(e, \Phi(n)) = 1$ y $ed \equiv 1 \pmod{\Phi(n)}$, donde $n = pq$. Se ha conjeturado que la seguridad de este criptosistema reside en la dificultad de descomponer el número n en sus factores primos. Los usuarios incluirán en un archivo público la pareja $\langle n, e\rangle$ y guardarán celosamente cualquiera de las entradas de la 4-tupla $\langle p, q, \Phi(n), d\rangle$. Además, deben tomar las siguientes precauciones:

1. El número n debe superar (a comienzos del siglo XXI) los 308 dígitos para que no sea factible su factorización. Algunas implementaciones actuales de RSA emplean módulos de 617 dígitos (2048 bits).

2. Los primos p y q deben ser ambos elevados para que sea difícil aplicar algoritmos de factorización como la criba general del cuerpo de números. Además, jamás deben ser elegidos de entre una lista conocida ni ser próximos el uno al otro; de lo contrario podría emplearse eficientemente el método de factorización de Fermat.

3. El valor de $(p-1, q-1)$ no debe ser elevado en exceso, y nunca debe ocurrir que $p-1 \mid q-1$. Si $(p-1, q-1)$ es muy elevado, $[p-1, q-1]$ sería pequeño en comparación con $\Phi(n)$ y factible de ser encontrado por fuerza bruta, lo que facilitaría la obtención de d. Es indeseable también que los factores primos de $\Phi(n)$ sean pequeños.

4. Para solucionar los problemas del apartado anterior, es deseable que p y q sean *primos seguros*. Además, tanto $p-1$ como $q-1$ deben tener factores primos elevados para evitar ataques de cifrados iterados.

5. El exponente d debe ser elevado, y esta es la razón de comenzar eligiéndolo para luego determinar e. Así se evita que pueda ser encontrado mediante procedimientos de prueba y error.

6. Dos usuarios nunca deben elegir el mismo módulo como parte de su clave pública. Si un tercer usuario cifra un mismo mensaje para ellos, este puede ser leído si los exponentes públicos de ambos usuarios son primos relativos.

7. No es recomendable tener un valor pequeño de e y, en caso de tenerlo, jamás debe ser elegido por varios integrantes de un mismo círculo RSA. De lo contrario, si se enviara un mismo mensaje cifrado a estos integrantes se podría obtener el texto plano aplicando convenientemente el *teorema chino del resto.*

8. Cuidarse de mensajes inocultables. Existen mensajes para determinadas claves cuyo cifrado es igual al mensaje original.

Cuestión 6. *Protocolo de intercambio de claves según el esquema de Diffie-Hellman y explicación de su supuesta fortaleza.*

Respuesta. El protocolo de intercambio de claves de Diffie-Hellman permite que ambas partes de una comunicación se pongan de acuerdo en una clave secreta sin tener que compartirla explícitamente a través de un canal potencialmente inseguro. Dicha clave podrá ser usada posteriormente, por ejemplo, como clave de un sistema de cifrado simétrico. El funcionamiento para dos usuarios i y j, que puede ser extendido a un número arbitrario de participantes, es el siguiente:

1. Ambos usuarios se ponen de acuerdo en un primo elevado n y eligen un generador g del grupo cíclico $\mathrm{GF}(n)^*$. La pareja $\langle n, g\rangle$ no tiene por qué ser secreta.

2. Cada usuario genera un número aleatorio elevado x_i de forma independiente. Dicho número actuará como *clave privada* y se guarda en secreto.

3. Cada usuario hace público el número $y_i = g^{x_i} \bmod n$, que actuará como *clave pública.*

4. Ahora ambos usuarios pueden utilizar la clave $K = g^{x_i x_j} \bmod n$ para comunicarse de forma segura. El usuario i obtiene la clave K a partir del número público y_j, realizando las operaciones

$$\begin{aligned} K &= y_j^{x_i} \bmod n \\ &= (g^{x_j})^{x_i} \bmod n \\ &= g^{x_j x_i} \bmod n \\ &= g^{x_i x_j} \bmod n. \end{aligned}$$

   El usuario j obtiene la clave de forma análoga, calculando en su caso

$$K = y_i^{x_j} \bmod n.$$

Si una tercera persona intercepta los mensajes intercambiados en este protocolo con el fin de averiguar la clave K, tendría que obtenerla partir de g, n, y_i e y_j. En definitiva, estaría ante una situación como la siguiente:

> Si $G = \langle g \rangle$ es un grupo y se conocen los valores g^a y g^b, encuentra el valor de g^{ab}.

Este problema se denomina *problema de Diffie-Hellman*. Para algunos primos, resolver este problema presenta una dificultad equivalente a resolver el *problema del logaritmo discreto* para y_i ó y_j. Por ejemplo, el atacante podría intentar calcular

$$K = y_i^{\left(log_g y_j\right)} \text{ mód } n.$$

Este último problema puede enunciarse en general como:

> Sea G un grupo y $g \in G$. Dado $a \in \langle g \rangle$, encuentra x tal que $g^x = a$.

Hasta ahora no se ha encontrado ningún algoritmo que resuelva el problema del logaritmo discreto en tiempo polinómico en el tamaño de la entrada, y es de hecho considerado intratable computacionalmente. El problema de Diffie-Hellman es en consecuencia intratable para ciertos grupos, y en ello radica finalmente la seguridad del protocolo homónimo. Para maximizar la seguridad es recomendable que n sea un *primo seguro* de 512 bits o incluso de 1024 bits.

Este protocolo de intercambio de claves es susceptible de recibir un ataque por persona interpuesta (*man-in-the-middle*). Una tercera persona malintencionada podría interceptar los valores públicos y_i e y_j y sustituirlos por otros y_i' e y_j' de su elección. De esta forma, si el atacante pudiera interceptar toda la comunicación entre las dos partes, podría sustituir sus propios mensajes. Este tipo de ataque puede ser frustrado añadiendo al protocolo un esquema de autenticación e identificación.

Cuestión 7. *El criptosistema de ElGamal.*

Respuesta. El criptosistema de ElGamal es un criptosistema de clave pública propuesto por Taher ElGamal en 1985. Está basado en el protocolo de intercambio de claves de Diffie-Hellman.

Preliminares

En primer lugar, se debe elegir un primo p elevado y un elemento $g \in \text{GF}(p)^*$. Es preferible, aunque no absolutamente necesario, que g sea un elemento primitivo de $\text{GF}(p)$, es decir, un generador del grupo cíclico $\text{GF}(p)^*$. Suponemos que las unidades de texto plano estarán expresadas en números de $\text{GF}(p)$. Ahora, cada usuario i elige aleatoriamente un valor elevado x_i, que será su clave privada. La clave pública será $y_i = g^{x_i} \text{ mód } p$.

Cifrando el mensaje

Supongamos que Alice quiere compartir con Bob un mensaje $0 \leq m < p$. Para ello, recoge la clave pública de Bob (y_B), elige aleatoriamente un número x con $0 < x < p$, y calcula $K = y_B^x \bmod p$. Finalmente envía a Bob la pareja $c = \langle g^x \bmod p, mK \bmod p \rangle$.

Descifrando el mensaje

Cuando Bob recibe el mensaje de Alice obtiene una pareja de números $\langle c_1, c_2 \rangle$ menores que p. Para descifrarlo hace lo siguiente:

1. Calcula el valor de K a partir de su clave privada, sin más que observar que $c_1^{x_B} = g^{x x_B} \bmod p = y_B^x \bmod p = K$.

2. Ahora calcula el inverso de K módulo p y computa el valor de m haciendo $c_2 K^{-1} \bmod p = mKK^{-1} \bmod p = m$, lo que le permite recuperar el texto plano.

Vulnerabilidades

No es recomendable elegir la misma clave x para cifrar más de un bloque del mensaje. Esto se debe a que si un atacante supiera el contenido de un primer mensaje m_1, podría calcular el contenido de las porciones sucesivas. Sean

$$c_{11} \equiv g^x \bmod p, \quad c_{21} \equiv m_1 K \bmod p,$$
$$c_{12} \equiv g^x \bmod p, \quad c_{22} \equiv m_2 K \bmod p.$$

Entonces, es fácil calcular $m_2 m_1^{-1} \bmod p = c_{22} c_{21}^{-1} \bmod p$, de donde se deriva inmediatamente m_2 (multiplicando por m_1).

La seguridad de este criptosistema reside en la dificultad de cálculo del logaritmo discreto, por lo que es imprescindible tomar las precauciones necesarias para que no sea fácil calcular dicho logaritmo. A saber, necesitamos que $p - 1$ contenga al menos un factor primo elevado. Para ello podemos por ejemplo elegir p como un *primo seguro*, es decir, de la forma $p = 2r + 1$ con r primo.

Cuestión 8. *El algoritmo de firma estándar (DSA).*

Respuesta. En primer lugar es necesario introducir el concepto de *función hash*. Se trata de una función H que transforma un *input* de longitud arbitraria en una lista de longitud fija de unos cuantos bits, usualmente entre 128 y 512. Debe ser una función para la que sea sencillo calcular el valor de $H(x)$ a partir de x pero computacionalmente intratable calcular x a partir de $H(x)$, y también que sea muy difícil encontrar x y x' distintos tales que $H(x) = H(x')$. La aplicación a un mismo

mensaje debería producir el mismo *hash*, pero cualquier manipulación de los datos debería dar lugar a un *hash* completamente diferente.

El algoritmo de firma estándar (DSA) fue propuesto en 1991, y sigue un esquema basado en el problema del logaritmo discreto. Supongamos que Alice quiere enviar a Bob un mensaje firmado, y consideremos fijada una función hash H.

Para generar las claves de firma, Alice realiza los siguientes pasos:

1. Elige un primo q de aproximadamente 160 bits, utilizando para ello un generador aleatorio de números y uno o varios tests de primalidad.
2. Elige un nuevo primo p congruente con 1 módulo q, con aproximadamente 512 bits.
3. Elige g_0 con $1 < g_0 \ll p-1$ y tal que $g_0^{p-1/q} \not\equiv 1 \mod p$. Llamamos g al valor $g_0^{p-1/q} \mod p$.
4. Elige un entero x aleatoriamente cumpliendo $0 < x < q$, que será la clave secreta.
5. La clave pública será $y = g^x \mod p$.

Para firmar un mensaje m, Alice procede como sigue:

1. Aplica el hash al mensaje y obtiene el valor $h = H(m)$, truncándolo si fuera necesario para que $0 < h < q$.
2. Genera aleatoriamente un entero k tal que $0 < k < q$.
3. Calcula $r = (g^k \mod p) \mod q$. En el poco probable caso de que $r = 0$, lo vuelve a intentar con un k distinto.
4. Calcula $k^{-1} \mod q$ y $s = (k^{-1}(h + xr)) \mod q$. En el poco probable caso de que $s = 0$, lo vuelve a intentar con un k distinto.

La firma del mensaje sería entonces la pareja $\langle r, s \rangle$, que se envía a Bob junto al valor h. Si Bob quiere verificar la firma, debe hacer lo siguiente:

1. En primer lugar, comprueba que $0 < r,\ s < q$. En otro caso, la firma no es válida.
2. Calcula $w = s^{-1} \mod q$.
3. Calcula $u_1 = hw \mod q$.
4. Calcula $u_2 = rw \mod q$.
5. Calcula $v = ((g^{u_1} y^{u_2}) \mod p) \mod q$.

La firma queda verificada únicamente si $v = r$. Mediante una aplicación sencilla del *teorema pequeño de Fermat* se comprueba que con los valores generados siguiendo el procedimiento descrito se cumple efectivamente $v = r$.

Cuestión 9. *Rasgos esenciales de SSH: cifrado, funcionamiento, negociación de cifrado para la sesión y autenticación del acceso del usuario al servidor*

Respuesta. SSH (del inglés *Secure SHell*) es un protocolo seguro utilizado principalmente para conectarse a un servidor de forma remota. Tras establecer conexión, todos los comandos escritos en la terminal local (*cliente*) se envían al servidor remoto (*host*) y se ejecutan allí.

Cifrado

En el protocolo SSH se contempla el uso combinado de tres tecnologías de cifrado diferentes:

1. En primer lugar, se utilizan algoritmos de *cifrado simétrico* para cifrar toda la comunicación durante una sesión SSH, de forma que cada sesión tenga asociada una clave de cifrado distinta. El cliente y el servidor se ponen de acuerdo para utilizar como cifrado bidireccional el realizado por algún criptosistema soportado por ambos, como puede ser AES, CAST128, Blowfish, etc.

2. También es utilizado en varias ocasiones el *cifrado asimétrico*. Por ejemplo, veremos que se utiliza durante el proceso inicial de intercambio de claves para configurar el cifrado simétrico, y como base del método de autenticación de clientes mediante pares de claves SSH.

3. Por último, se emplea la técnica de *hashing criptográfico* (ver cuestión 8) como método de creación de una firma o resumen que garantice la integridad de los mensajes. Junto a cada paquete se envía un *código de autenticación de mensaje basado en hash*, que permite comprobar que el mensaje está intacto y no ha sido modificado. Estos códigos se calculan a partir de la clave de sesión, del número de secuencia del paquete y del contenido del mensaje.

Funcionamiento

Se emplea un modelo cliente-servidor para autenticar a dos partes y cifrar los datos entre ellas. La sesión SSH se establece en dos etapas separadas, consistiendo la primera en acordar y establecer el mecanismo de cifrado que protegerá la comunicación futura. La segunda etapa comprende la autenticación del usuario y el consecuente acceso al servidor en caso de éxito.

El *host* se encarga escuchar las peticiones en el puerto designado (por defecto el 22) y de autenticar al cliente y generar un entorno correcto si las credenciales son aceptadas. Por su parte, el cliente es responsable de iniciar la conexión TCP, negociar la conexión segura, verificar la identidad del host y finalmente proporcionar las credenciales para autenticarse.

Negociación de cifrado para la sesión

Cuando un cliente realiza una conexión TCP, el servidor responde con las versiones del protocolo que admite, y si el cliente puede coincidir con alguna de ellas la conexión continúa. El servidor proporciona también su clave pública para que el cliente pueda verificar su identidad.

Llegados a este punto, ambas partes negocian un algoritmo de código de autenticación de mensajes (MAC) y una clave de cifrado para la sesión. Para esto último se utiliza una versión del protocolo de intercambio de claves de Diffie-Hellman (ver cuestión 6) que involucra un generador de cifrado (usualmente AES). De esta forma, la clave generada nunca se transmite entre el cliente y el *host*. Este secreto compartido se utiliza como clave de cifrado simétrico, envolviendo toda la comunicación posterior en un túnel encriptado que no puede ser descifrado por agentes externos.

Autenticación de acceso al servidor

El método más simple para autenticar el acceso del usuario al servidor es la autenticación por contraseña, en la que el servidor simplemente solicita la contraseña de la cuenta con la que se intenta iniciar sesión. Aunque la contraseña está protegida al viajar por el túnel cifrado, las limitaciones en cuanto a longitud y complejidad de la misma hacen que este método no sea recomendado generalmente.

La alternativa más popular y recomendada es el uso de pares de claves SSH asimétricas. El cliente crea una pareja de claves pública y privada y, tras haber accedido por primera vez al servidor mediante algún otro método de autenticación, carga en este su clave pública (en un archivo llamado `certified_keys`). En accesos posteriores, el procedimiento de autenticación es el siguiente:

1. El cliente envía al servidor un identificador para el par de claves con el que desea autenticarse.

2. El servidor verifica el archivo de claves autorizadas para la cuenta con la que se intenta iniciar sesión y comprueba si alguna clave coincide con el identificador proporcionado.

3. En caso afirmativo, el servidor genera un *mensaje de desafío* (un número aleatorio de 256 bits) que envía al cliente cifrado con la clave pública en cuestión.

4. El cliente utiliza su clave privada para descifrar el mensaje, lo combina con la clave de sesión actual y calcula el *hash* SHA2 de este valor, que envía de nuevo al servidor.

5. El servidor combina de igual forma el desafío original con la clave de sesión y le calcula el *hash*. Si coincide con el que el cliente envió de vuelta, se demuestra que efectivamente este estaba en posesión de la clave privada y se autentica con éxito.