Criptología de clave pública

Félix Delgado - Ana Núñez [*]

Curso 2017-18

1. Introducción

En el primer tema ya comentamos esencialmente en qué consiste un sistema criptográfico de clave pública, o asimétrico. El modelo fue creado por W. Diffie y M. E. Hellman en 1975 y publicado en 1976 en el artículo titulado *New Directions in Cryptography*. La criptografía de clave pública no nace con la idea de sustituir a los métodos de cifrado con clave privada, ni es ese su interés primordial hoy día, sino que pretende resolver con ella algunos de los grandes problemas que presenta el uso masivo de cifrado de clave privada en redes vulnerables, como lo son por ejemplo las electrónicas. Los dos problemas principales son:

Intercambio de claves: El cifrado de clave privada requiere que dos usuarios concierten una determinada clave y que esta solo sea conocida por ellos. Ahora bien, no lo pueden hacer a través del canal, por ser este vulnerable; y en un sistema electrónico el intercambio presencial no tiene sentido.

Autentificación e integridad: Otro de los problemas que surgen es el de la autentificación, tanto del interlocutor como del mensaje. Es decir debemos garantizar que nuestro interlocutor es quien dice ser, de la misma manera que debemos tener la garantía de que un determinado mensaje (o clave) es legítimo y no ha sido sustituido o alterado por un tercero.

Un sistema criptográfico de clave pública no presenta problemas de intercambio de claves, ya que cada usuario puede, si dispone de suficiente capacidad de cálculo, generar sus propias claves y su clave privada no sale nunca del ámbito privado del usuario que la genera. Veremos después cómo se pueden resolver los problemas de autentificación e integridad mediante el uso de la criptografía de clave pública.

La razón por la cual los sistemas de clave pública no se suelen usar en los intercambios masivos de información es la rapidez. Por ejemplo, el AES (que es de clave privada) es centenares de veces más rápido que el sistema RSA (de clave pública).

[*]Estas notas se han escrito a partir del material elaborado por el Prof. Delgado a lo largo de varios cursos de docencia de la materia.

Los sistemas de clave pública se suelen usar en sistemas mixtos, por ejemplo, para intercambiar claves y autentificar usuarios previamente a transmitir la información con un sistema de clave privada.

En el artículo ya citado, Diffie y Hellman establecen una serie de condiciones que deben ser satisfechas por un criptosistema de clave pública. Dichas condiciones reciben el nombre de ***condiciones de Diffie y Hellman*** y son las siguientes:

1. El cálculo de las llaves, pública y privada, debe ser computacionalmente sencillo.
2. El proceso de cifrado debe ser computacionalmente sencillo.
3. El proceso de descifrado, conociendo la llave secreta, debe ser también computacionalmente sencillo.
4. La obtención de la llave secreta, a partir de la pública, debe ser un problema computacionalmente imposible.
5. La obtención del mensaje en claro, conociendo el mensaje cifrado y la llave pública, debe asimismo ser computacionalmente imposible.

2. El sistema criptográfico RSA

El sistema criptográfico RSA fue propuesto por Rivest, Shamir y Adleman (de ahí su nombre) en 1978 y es el primer criptosistema de clave pública construido. A fecha de hoy sigue siendo el más utilizado. Se apoya en que la factorización de números enteros se considera un problema intratable.

Pasemos ahora a describir el sistema:

2.1. Generación de una clave RSA

1. El usuario elige aleatoria e independientemente una pareja de números primos p, q, y calcula $n = pq$.
2. También elige arbitrariamente un entero e tal que $0 < e < \varphi(n) = (p-1)(q-1)$ y $\mathsf{mcd}\,(e, \varphi(n)) = 1$ (recordemos que la función de Euler $\varphi(n)$ es el número de enteros positivos menores que n y primos con n).
3. Calcula el inverso modular de e módulo $\varphi(n)$, $d \equiv e^{-1} \mod \varphi(n)$.
4. La clave pública generada es el par (n, e), y la privada es d.

2.2. Cifrado

Hay dos formas de ver el cifrado. La primera es entender el conjunto de mensajes como $\mathbb{Z}/n\mathbb{Z}$. En este caso, si se quiere enviar M con $0 \leq M < n$ a un usuario de clave pública (n, e), se cifra M por

$$C = M^e \text{ mód } n.$$

Existe una forma mejor, que consiste en entender el RSA como un cifrado por bloques, y que explicamos a continuación.

- Si (n, e) es la clave pública y el vocabulario es $\Sigma = \mathbb{Z}/N\mathbb{Z}$, el tamaño de los bloques, k, es el mayor entero tal que $N^k \leq n$ (es decir, $N^k \leq n < N^{k+1}$).
- Un bloque de longitud k será de la forma $M = M_1 \dots M_k$ con $M_i \in \mathbb{Z}/N\mathbb{Z}$, es decir, $0 \leq M_i < N$, y se puede interpretar como la escritura en base N del entero $m = M_1 \cdot N^{k-1} + M_2 \cdot N^{k-2} + \cdots + M_k$. La elección de k hace que $m < N^k \leq n$.
- Aplicamos ahora a m el sistema de cifrado anterior, calculando $c = m^e$ mód n. Tendremos $c < n < N^{k+1}$, por lo que la longitud de la expresión de c en base N pueden ser hasta $k + 1$. Escribimos $c = C_1 \cdot N^k + C_2 \cdot N^{k-1} + \cdots + C_{k+1}$.
- El bloque M (de longitud k) se cifra finalmente como el bloque (de longitud $k + 1$) $C = C_1 C_2 \dots C_{k+1}$.

El hecho de que el tamaño del bloque cifrado sea mayor que el del claro hace que este método se implante generalmente en modo ECB, es decir, dado un mensaje $M = M_1 M_2 \dots$ con vocabulario $\Sigma = \mathbb{Z}/N\mathbb{Z}$:

- Se calcula k tal que $N^k \leq n < N^{k+1}$.
- Se divide M en bloques de longitud k (usando algún procedimiento que complete el último de ser necesario), $M = B_1 \dots B_r$.
- Se cifra por el procedimiento anterior cada bloque B_i, $C_i = c(B_i)$ (bloques de longitud $k + 1$).
- El cifrado de M es $c(M) = C_1 C_2 \dots C_r$. Obsérvese que el número de símbolos ha aumentado en uno por bloque.

Se podría también aplicar una versión ligeramente modificada del modo CBC. Lo que en ningún caso tendría sentido sería aplicar los modos CFB u OFB, pues en ellos para el descifrado se utiliza el método básico de cifrado (y no el de descifrado), y este método es en este caso público.

2.3. Descifrado

Está basado en el siguiente resultado, que es consecuencia del pequeño teorema de Fermat, del teorema de los restos y del hecho de que $ed \equiv 1$ mód $\varphi(n)$:

Teorema 1. *Si* $0 \leq m < n = pq$, *entonces* $m^{ed} \equiv m$ mód n.

Por lo tanto, si se ha seguido el método ECB, para recuperar el texto claro una vez conocido el cifrado, C se hace lo siguiente:

1. Se calcula k tal que $N^k < n \leq N^{k+1}$.
2. Se divide C en bloques de longitud $k+1$, $C = C_1 \dots C_r$.
3. Se descifra cada bloque $C_i = C_1^i \dots C_{k+1}^i$ transformándolo en el entero $c_i = C_1^i \cdot N^k + C_2^i \cdot N^{k-1} + \dots + C_{k+1}^i$, calculando $b_i = c_i^d$ mód n y escribiendo b_i en base N con longitud k, $b_i = B_1^i \cdot N^{k-1} + B_2^i \cdot N^{k-2} + \dots + B_k^i$. El descifrado de C_i es $B_i = B_1^i B_2^i \dots B_k^i$.
4. El mensaje descifrado es $M = B_1 \dots B_r$.

2.4. Seguridad y eficiencia del RSA

Para asegurarnos que el sistema descrito realmente es eficaz es imprescindible antes verificar que satisface las condiciones de Diffie y Hellman.

Necesitamos asegurar que el problema de encontrar la clave secreta a partir de la pública es computacionalmente imposible. Esto de hecho no se ha demostrado, aunque sí el que es un problema computacionalmente equivalente al de factorizar n y al de calcular $\varphi(n)$. Por lo tanto, la seguridad del método reposa en la certidumbre de que, en general, es difícil factorizar un número que es producto de dos primos. Evidentemente esto no es verdad en general, los primos en cuestión deben satisfacer algunas condiciones suplementarias. En general los métodos de factorización eficaces existentes imponen condiciones sobre el tipo de primos que se admiten. Las parejas de primos que evitan los métodos de factorización conocidos se llamarán ***primos seguros***, aunque este concepto es variable dado que nuevos métodos de factorización impondrán nuevas condiciones. Algunas de las propiedades de un primo seguro son evidentes, por ejemplo, han de ser "grandes"; a día de hoy se estima que los primos seguros han de tener un mínimo de 1024 bits (más de 300 dígitos decimales).

Analizaremos los algoritmos de factorización y también el problema de encontrar primos seguros, así como otras posibles debilidades del RSA posteriormente.

Por el momento, lo que podemos afirmar es que, conocidos p y q, el cálculo de las claves pública y secreta es computacionalmente fácil. La elección de e está sometida a dos condiciones: que no sea demasiado grande, para que d sí lo sea y haga el descifrado más difícil, y que no sea excesivamente pequeño, pues en este caso existen

ciertos ataques que funcionan. De hecho, existe un recomendación de que sea al menos $2^{16}+1$, que es un número primo. El cálculo de d se hace como ya explicamos mediante el algoritmo de Euclides extendido.

Y los procesos de cifrado y descifrado consisten en una potenciación modular, que también es un algoritmo eficaz. De hecho, el proceso más costoso es el de descifrado, dado que d tiene un número de bits cercano al de n, pero, a partir de $n = pq$ y usando el teorema de los restos se puede dar otro procedimiento más rápido. Aun así, los procesos de cifrado y descifrado RSA son mucho más lentos que, por ejemplo, los del DES.

3. El sistema de Rabin

Es beneficioso que la seguridad de un criptosistema se base en la dificultad de un problema matemático que sea de interés también fuera de la criptografía, pues de esa manera es más probable que los avances que se realicen en el campo sean públicos y no queden en posesión de determinadas organizaciones. En el caso del RSA, su seguridad se basa en la dificultad de factorizar enteros, aunque no se sabe si se puede romper el RSA sin resolver la factorización.

El sistema de Rabin (1979) está de nuevo basado en la dificultad de la factorización, y de hecho se puede probar que un ataque computacionalmente eficaz contra este sistema proporcionaría un método eficaz de factorización.

3.1. Generación de las claves

El usuario elige aleatoria e independientemente una pareja de números primos p, q tales que $p \equiv q \equiv 3$ mód 4 , y calcula $n = pq$. La clave pública es n, y la privada es el par (p, q).

La condición $p \equiv q \equiv 3$ mód 4 hace el cálculo de las raíces cuadradas módulo n particularmente sencillo, según vimos. Esto hace el descifrado más eficiente, pero el sistema también funciona sin esta condición.

3.2. Cifrado

Como en el RSA, se puede entender el conjunto de mensajes como $\mathbb{Z}/n\mathbb{Z}$. En este caso, M con $0 \leq M < n$ se cifra, para ser enviado al usuario de clave pública n, por

$$C = M^2 \text{ mód } n.$$

Y este método básico se puede extender a un método de cifrado por bloques, lo que se hace de la misma forma que en el RSA.

3.3. Descifrado

Se calcula el mensaje claro M a partir del cifrado C extrayendo su raíz cuadrada módulo n, tal como se explicó en su momento. El problema es que se obtienen cuatro raíces cuadradas módulo n de C, y solo una de ellas es el mensaje M.

Justamente, una de las razones por las que este sistema no es demasiado útil es porque, en general, no hay forma de distinguir cuál de las cuatro soluciones es el mensaje m. Posiblemente solo haya una raíz que nos proporcione un texto inteligible, pero esto no es una vía demasiado satisfactoria si lo que se está cifrando es, por ejemplo, una sucesión de números y no un texto con sentido.

Para eliminar este inconveniente se utilizan diversos sistemas, muchos de los cuales consisten en añadir redundancias que permitan distinguir el descifrado correcto, o en elegir los textos cifrados con ciertas características.

3.4. Seguridad y eficiencia

Como ya se ha dicho, la seguridad del sistema de Rabin se basa en la dificultad de la factorización. De hecho se puede probar que un ataque computacionalmente eficaz contra este sistema proporcionaría un método eficaz de factorización.

En cuanto a la eficiencia, el cifrado solo requiere el cálculo de un cuadrado, luego es más rápido que el cifrado RSA. Sin embargo el descifrado, siendo polinomial, no es más eficiente que el del RSA.

4. Intercambio de claves de Diffie-Hellamn

Vamos a describir el protocolo de Diffie y Hellman para el intercambio de claves en canales inseguros. No es un criptosistema, pero sí es la base para el criptosistema de ElGamal que describiremos después.

La idea es la siguiente: Alicia y Benito quieren usar un sistema de cifrado de clave privada, para lo cual debe ponerse de acuerdo en una clave secreta. El protocolo de Diffie y Hellman permite a Alicia y Benito usar el canal inseguro para el intercambio de la clave. Puesto que el canal es inseguro, la información intercambiada puede ser obtenida por alguien, pero esa información no es suficiente para reconstruir la clave secreta.

La seguridad de este protocolo se basa en el *problema del logaritmo discreto* que vamos a describir.

4.1. El problema del logaritmo discreto

Es conocido que la condición para que un elemento a tenga inverso módulo un entero n es que $\mathsf{mcd}\,(a, n) = 1$. Se dice en este caso que a mód n es *unidad* en $\mathbb{Z}/n\mathbb{Z}$.

Es evidente que 1 mód n es unidad, y también que el producto de dos unidades es unidad (esto es cierto en cualquier anillo), luego el conjunto de unidades de $\mathbb{Z}/n\mathbb{Z}$ es, con la operación de multiplicación, un grupo conmutativo. Se denota por $(\mathbb{Z}/n\mathbb{Z})^*$.

Si p es un número primo, entonces $(\mathbb{Z}/p\mathbb{Z})^*$ está formado por todos los elementos distintos de 0 de $\mathbb{Z}/p\mathbb{Z}$. Y además, en este caso, el grupo tiene una estructura especial:

Teorema 2. *Si p es primo, $(\mathbb{Z}/p\mathbb{Z})^*$ es un grupo* cíclico de orden $p-1$. *Es decir, existe $g \in (\mathbb{Z}/p\mathbb{Z})^*$ tal que $(\mathbb{Z}/p\mathbb{Z})^* = \{1, g, g^2, g^{p-2}\}$ y además $a^{p-1} = 1$ para todo $a \in (\mathbb{Z}/p\mathbb{Z})^*$.*

Un elemento g como el descrito en el teorema se llama *raíz primitiva módulo* p. El teorema dice que dado cualquier $A \in \{1, 2, \ldots, p-1\}$ existe un exponente $a \in \{0, 1, \ldots, p-2\}$ tal que $A \equiv g^a$ mód p. El exponente a es el *logaritmo discreto de A en la base g*, y no se conoce ningún algoritmo eficiente que lo calcule (aunque no está probado que no exista). La dificultad de este cálculo es la base de la seguridad del protocolo de Diffie-Hellman.

4.2. Intercambio de claves de Diffie-Hellman

El protocolo de Diffie y Hellman para intercambiar claves entre dos usuarios a través de un canal inseguro funciona como sigue.

Supongamos que Alicia y Benito desean ponerse de acuerdo en una clave común que usarán posteriormente en un sistema criptográfico de clave privada, y que solo pueden usar para comunicarse un canal inseguro. Supondremos que dicha clave va a ser un elemento de un cuerpo del tipo $\mathbb{Z}/p\mathbb{Z}$, aunque se puede generalizar a cualquier cuerpo finito. El método es el siguiente:

1. Conciertan un número primo p "grande" y una raíz primitiva módulo p, g. Ambos pueden ser públicos, por lo que Alicia y Benito usan el canal inseguro para ponerse de acuerdo en estos números.

2. Alicia elige aleatoriamente un entero $a \in \{0, 1, \ldots, p-2\}$ (que mantiene en secreto), calcula $A = g^a$ mód p y lo transmite a Benito. Benito elige $b \in \{0, 1, \ldots, p-2\}$ y transmite $B = g^b$ mód p a Alicia.

3. Alicia y Benito toman como clave secreta $K = g^{ab}$ mód p. Para ello, Alicia eleva el número recibido de Benito, A, a su clave secreta, a, y Benito eleva el recibido de Alicia, B, a su clave secreta b.

Un atacante puede conocer los números p, g, A y B, y sin embargo no dispone de medios para calcular los logaritmos discretos a de A ni b de B. Lo que quiere es encontrar la clave K a partir de p, g, A, B. Esto se conoce como *el problema de Diffie-Hellman*. Es claro que si se encontrase un algoritmo eficiente para calcular

el logaritmo discreto, se resolvería este problema. Por ahora, no se ha probado que ambos problemas sean equivalentes.

Existe otro posible ataque a este protocolo, y es el hecho de que Alicia no sabe si el número A que supuestamente recibe de Benito viene en realidad de él, y lo mismo ocurre con Benito. Así, un atacante podría interceptar este intercambio y enviar a Alicia y Bob sus propias claves con lo que podría descifrar los mensajes que interceptase. Para evitar este ataque se utiliza los sistemas de firma digital, que explicaremos más adelante.

Basados en el problema del logaritmo discreto hay varios sistemas de clave pública (ElGamal, Massey-Omura) y de firma digital, notablemente el sistema DSS (Digital Standard Signature) propuesto por el NIST (National Institute of Standards and Technology, USA) como sistema de firma digital estándar.

5. El sistema criptográfico de ElGamal

Es un sistema muy relacionado con el protocolo de intercambio de claves de Diffie-Hellman.

5.1. Generación de las claves

El usuario elige un número primo p y una raíz primitiva módulo p, g. Luego elige aleatoriamente un exponente $a \in \{0, \ldots, p-2\}$ y calcula $A = g^a \bmod p$. Su clave pública es (p, g, A) y la privada es a.

5.2. Cifrado

Se puede entender el conjunto de mensajes como $\mathbb{Z}/p\mathbb{Z}$. Dado M con $0 \leq M < p$, si un usuario quiere enviar M al usuario de clave pública (p, g, A), en primer lugar elige aleatoriamente $b \in \{0, 1, \ldots, p-2\}$, calcula $B = g^b \bmod p$ y $C = A^b M \bmod p$, y cifra M por el par (B, C).

5.3. Descifrado

Si el usuario con clave pública (p, g, A) y privada a recibe (B, C), entonces calcula en primer lugar $K = B^a \bmod p$. Puesto que $K = g^{ab} \bmod p = A^b \bmod p$, M se recupera como $M = C/K \bmod p$.

Para evitar tener que dividir módulo p, se puede observar que también $M = B^{p-1-a}C \bmod p$, puesto que $B^{p-1-a}C \equiv g^{b(p-1-a)}A^b M \equiv g^{b(p-1-a)}g^{ab}M \equiv (g^{p-1})^b M \equiv M \bmod p$.

5.4. Eficiencia y seguridad

El descifrado, al igual que en el RSA, requiere de una exponenciación modular, aunque en este caso el Teorema de los Restos no mejora este cálculo.

Para el cifrado es necesario hacer dos potencias, $B = g^b$ mód p y A^b mód p. Sin embargo, ambas son independientes del mensaje que se quiere cifrar y se pueden por tanto hacer previamente, de forma que para cifrar solo es necesaria una multiplicación modular, por lo que el sistema es mucho más rápido que el cifrado RSA. Sin embargo, los valores precalculados deben ser secretos y deben ser almacenados de forma segura.

La seguridad está de nuevo basada en la dificultad de encontrar el logaritmo discreto. Y es fácil ver que romper el criptosistema de ElGamal equivale de hecho a resolver el problema de Diffie-Hellman.

Los sistemas criptográficos basados en el logaritmo discreto son seguros hasta el momento, aunque conviene ser precavidos, pues hay métodos que, en condiciones particulares, son eficientes para su cálculo. Para evitar la aplicación de algunos algoritmos de cálculo del logaritmo discreto, el número p debe ser al menos de longitud binaria 768, y evitar los primos de determinadas formas. Parece que la mejor es una elección aleatoria entre los primos de determinada longitud.

Por otra parte, el exponente b elegido para el cifrado se debe elegir nuevo y de forma aleatoria para cada mensaje, lo cual evita ataques de tipo estadístico.

6. Primalidad

Todos los sistemas comentados precisan para la generación de claves de la elección previa de números primos "grandes". Por lo tanto, para que el cálculo de tales claves sea computacionalmente sencillo es necesario contar con criterios eficaces de primalidad. El hecho crucial que hace por ejemplo del sistema RSA un sistema eficiente de clave pública, es precisamente que el problema de saber si un número es o no primo es mucho más sencillo que encontrar una factorización del mismo.

Dado que los métodos de factorización conocidos no son computacionalmente buenos, no podemos pensar en probar que un número es primo intentando encontrar una factorización del candidato. Hay dos tipos de test usados para determinar si un número es o no primo.

Los *test deterministas*, cuya salida indica sin lugar a dudas si un número es o no primo. Recientemente Agrawal, Kayal y Saxena han probado que la primalidad es un problema de complejidad polinómica exhibiendo un algoritmo determinista de complejidad $\mathcal{O}((\log n)^{12})$. Bajo ciertas condiciones, que son ciertas para valores muy altos del número n y que se conjeturan ciertas en general, la complejidad desciende hasta $\mathcal{O}((\log n)^6)$. En la práctica criptográfica dicho resultado no es demasiado útil, ya que (como veremos a continuación) existen algoritmos probabilísticos muy

rápidos que certifican la primalidad de un número entero con probabilidad tan alta como se desee, pero resuelve una de las conjeturas más importantes abiertas en este campo. De hecho Manindra Agrawal recibió el premio del Instituto Clay en Octubre de 2002 por este resultado.

Los *test probabilísticos* funcionan de la siguiente forma. Si aplicados a un número n el algoritmo determina que n no es primo, entonces con toda seguridad n no es primo. Pero si el algoritmo determina que sí lo es, entonces lo que ocurre es que existe una cierta probabilidad, determinada por el test, de que esto sea cierto, aunque podría no serlo. Estos test se pasan el número de veces que se desee con vistas a hacer la probabilidad de error tan pequeña como se quiera.

Muchos test probabilísticos se basan en el concepto de *número pseudoprimo*. Hay varias definiciones distintas de pseudoprimalidad. Todas ellas se hacen basándose en alguna propiedad que los números primos deben necesariamente cumplir, aunque quizá no solo ellos.

Describiremos a continuación algunos de los algoritmos de primalidad más habituales.

6.1. Test de Lucas y test de Fermat

El pequeño teorema de Fermat es la base para un primer criterio determinista de primalidad, aunque poco efectivo. Sabemos que si p es un número primo entonces $a^{p-1} \equiv 1 \text{ mód } n$ para todo entero a primo con p. Y se puede probar, con unos mínimos conocimientos de teoría de grupos, el siguiente resultado:

Teorema 3. *n es primo si y sólo si existe un entero a primo con n tal que para todo h con $0 < h < p-1$ se tiene que $a^h \not\equiv 1 \text{ mód } n$.*

Así, para decidir si n es o no primo con arreglo a este criterio, se comprueba si la condición anterior se cumple para algún a. Este método, determinista, se conoce como *test de Lucas*.

La efectividad de dicho test es escasa. La condición $a^h \not\equiv 1 \text{ mód } n$ se puede sustituir por $a^{\frac{n-1}{p}} \not\equiv 1 \text{ mód } n$ para todo primo p divisor de $n-1$, pero esta reducción requiere conocer la factorización de $n-1$, por lo que solo es útil cuando n tiene alguna forma especial. Aún en este caso, por ejemplo para saber si un número de Fermat $F_n = 2^{2^n} + 1$ es o no compuesto deberíamos, en principio, probar con todos los posibles números a.

Ahora bien, el pequeño teorema de Fermat nos da la primera definición de pseudoprimo:

Definición. *1. Se dice que un número n es pseudoprimo respecto de una base a tal que $\mathsf{mcd}\,(a,n) = 1$ si $a^{n-1} \equiv 1 \text{ mód } n$.*

2. Se dice que n es pseudoprimo si lo es con respecto a toda base a tal que $\mathsf{mcd}\,(a,n) = 1$.

3. Un número de Carmichael es un número que es pseudoprimo pero no primo.

La pregunta obvia es si existen números de Carmichael. Se puede probar que n es de Carmichael si y solo si la descomposición en producto de primos de n es de la forma $n = p_1 \cdots p_r$ con $p_i \neq p_j$ si $i \neq j$ (n es *libre de cuadrados*) y $r > 1$, y además $p_i - 1$ divide a $n - 1$ para $1 \leq i \leq r$.

En 1992, Alfords, Granville y Pomerance probaron la existencia de infinitos números de Carmichael. Los primeros son $561 = 3 \cdot 11 \cdot 17$, $1105 = 5 \cdot 13 \cdot 17$, $1729 = 7 \cdot 13 \cdot 19$. Existen tablas con los números de Carmichael de hasta 20 cifras.

Un hecho curioso es que los números de Carmichael se podrían utilizar en un sistema RSA sin mayor problema (ya que cumplen la propiedad básica usada), excepto por el pequeño detalle de que son muy sencillos de factorizar, como veremos después.

El test de Fermat sería el siguiente:

- La entrada es un número n, del que queremos saber si es primo o no.
- Se elige de forma aleatoria a primo con n.
- La salida es NO si $a^{n-1} \not\equiv 1$ mód n y SI en caso contrario.

Es evidente que si la salida del test es NO entonces n es con total certeza compuesto. Se puede probar que si n no es un número de Carmichael y no es primo, entonces n es pseudoprimo respecto de como mucho el 50 % de las bases a, por lo que en este caso, si la salida del test es SI, existe una probabilidad menor que $1/2$ de que el número sea compuesto. Si se quiere disminuir la probabilidad de error a ϵ, no hay más que pasar el test k veces con $\frac{1}{2^k} < \epsilon$.

Sin embargo, persiste el problema de saber previamente si n es de Carmichael o no.

6.2. Test de Solovay-Strassen

Si $p > 2$ es un número primo y a es primo con p, del pequeño teorema de Fermat se tiene que $(a^{\frac{p-1}{2}})^2 = 1$ mód p, por lo que $a^{\frac{p-1}{2}} = \pm 1$ mód p. La propiedad de los números primos que vamos a explotar ahora es la siguiente:

Proposición 4. *Si p es un número primo impar y a es primo con p entonces $a^{\frac{p-1}{2}} \equiv 1$ mód p si a es un cuadrado módulo p y $a^{\frac{p-1}{2}} \equiv -1$ mód p si no lo es.*

Esta propiedad se puede escribir en función del símbolo de Legendre que vamos a definir a continuación.

6.2.1. Símbolos de Legendre y Jacobi

Definición. *Dados un número primo p y un entero a, el* símbolo de Legendre de a respecto de p *se define como*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & si\ a = 0 \\ 1 & si\ a \neq 0\ es\ un\ cuadrado\ módulo\ p \\ -1 & si\ a \neq 0\ no\ es\ un\ cuadrado\ módulo\ p \end{cases}$$

Dados un número $n = p_1^{r_1} \cdots p_t^{r_t}$ y un entero a, el símbolo de Jacobi de a respecto de n *es*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{r_1} \cdots \left(\frac{a}{p_t}\right)^{r_t}.$$

Algunas propiedades que proporcionan métodos de cálculo de estos símbolos:
Propiedades: Si n es impar positivo

1. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.
2. $\left(\frac{a+in}{n}\right) = \left(\frac{a}{n}\right)$.
3. $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$.
4. *Reciprocidad:* si también a es impar positivo se tiene $\left(\frac{a}{n}\right) = \left(\frac{n}{a}\right)(-1)^{\frac{(a-1)(n-1)}{4}}$.

La proposición anterior se puede reescribir:

Proposición 5. *Si p es un número primo impar y a es primo con p, $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}}$ mód p.*

Esta propiedad da lugar a la siguiente definición:

Definición. *1. Se dice que un número n es pseudoprimo de Euler respecto de una base a tal que $\mathsf{mcd}\,(a,n) = 1$, si $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}}$ mód n*

2. Se dice que n es pseudoprimo de Euler si lo es con respecto a toda base a tal que $\mathsf{mcd}\,(a,n) = 1$.

Es claro que si n es pseudoprimo respecto de a, entonces es pseudoprimo de Euler con respecto de a. Además, a diferencia de lo que ocurría con el criterio de Fermat, en este caso se tiene que de hecho n es primo si y sólo si $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right)$

para todo a primo con n, es decir, si y solo si es pseudoprimo de Euler. Este criterio recibe el nombre de criterio de Euler, y aunque en principio daría lugar a un test determinista, dado que sería necesaria la comprobación para todas las bases a con $\mathsf{mcd}\,(a,n)=1$, tampoco sería eficiente.

Sin embargo, se puede demostrar que si n es un número compuesto impar, al menos el 50 % de las bases a no cumplen la condición $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right)$ mód n. Por tanto, si elegimos k bases a de forma aleatoria con la condición $\mathsf{mcd}\,(a,n)=1$ y comprobamos la condición $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right)$ mód n para cada una de ellas tendremos que:

1. Si alguno de los test falla entonces con seguridad n es compuesto.
2. Si n supera los k tests entonces la probabilidad de que n sea compuesto es a lo sumo de $1/2^k$.

Disponemos por tanto de un test probabilístico realmente eficiente ya que, usando la potenciación modular, la complejidad de k pasadas del test es del orden de $O(k(\log n)^3)$.

6.3. Test de Rabin-Miller

El test de Solovay-Strassen quedó en desuso ante la aparición del concepro de pseudoprimo fuerte. La idea subyacente de que la ecuación X^2-1 tiene sólo las soluciones ± 1 módulo un primo se puede refinar en el siguiente sentido.

Proposición 6. *Sea p primo impar y escribamos $p-1=2^e d$ con d impar. Sea a un entero prio con p. Entonces la sucesión a^d mód p, a^{2d} mód $p, a^{2^2 d}$ mód $p, \ldots, a^{2^{e-1}d}$ mód p tiene la peculiaridad de que*

- *o bien está formada toda ella por unos (lo cual equivale a que $a^d \equiv 1$ mód p),*
- *o bien existe j, con $0 \leq j < e$, de manera que $a^{2^j d} = -1$ (en este caso los siguientes son todos iguales a 1).*

Definición. *1. Se dice que un número impar n es pseudoprimo fuerte respecto de una base a tal que $\mathsf{mcd}\,(a,n)=1$ si, escribiendo $n-1=2^e d$ con d impar, o todos los elementos la sucesión a^d mód n, a^{2d} mód $n, a^{2^2 d}$ mód $n, \ldots, a^{2^{e-1}d}$ mód n son 1 o uno de ellos es -1.*

2. Se dice que n es pseudoprimo fuerte si lo es con respecto a toda base a tal que $\mathsf{mcd}\,(a,n)=1$.

Se puede comprobar que si n es pseudoprimo fuerte respecto de a entonces es pseudoprimo de Euler respecto de la base a, es decir, $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right)$. Por tanto,

también tendremos que n es pseudoprimo fuerte si y sólo si es primo, con lo que disponemos de un nuevo test de primalidad, llamado de Rabin-Miller. Como test determinista tampoco es un test eficiente, ya que sería necesario comprobarlo para todas las bases.

Pero se puede demostrar que si n es un entero impar compuesto, el número de bases a para las que n es pseudoprimo fuerte es a lo sumo $\varphi(n)/4$. Por lo tanto, eligiendo k bases al azar, si n es un entero impar que pasa el test para todas ellas (es decir, si es pseudoprimo fuerte para todas ellas), n es compuesto con probabilidad menor que $1/4^k$.

6.4. Otros algoritmos

Existen muchos más test de primalidad que los descritos, aunque para los objetivos de este curso el test de Rabin-Miller es más que suficiente. En particular, merece la pena mencionar la existencia de tests de primalidad específicos para cierto tipo de enteros.

Por ejemplo para los números de Mersenne (los de la forma 2^p-1, p primo) existen algoritmos eficientes como el test de Lucas-Lehmer que es la versión computacional del siguiente enunciado:

Teorema 7. *Sea $M_p = 2^p - 1$, p primo impar. Sea $r_1 = 4$ y, para $k \geq 2$, sea $r_k = r_{k-1}^2 - 2 \mod M_p$. Entonces M_p es primo si y sólo si $r_{p-1} \equiv 0 \mod M_p$.*

Los números de Mersenne suelen proporcionar los primos "récord" y se utilizan muy frecuentemente como pruebas de rapidez de cómputo de nuevos sistemas. En la página web: http://www.utm.edu/research/primes/ se puede encontrar información actualizada sobre los primos mayores conocidos y, en particular, sobre los primos de Mersenne conocidos. La página http://www.mersenne.org/ es la página "oficiosa" dedicada a los primos de Mersenne.

En cuanto a los test de propósito general, los basados en curvas elípticas juegan un papel destacado. Golwasser y Killian en 1986 proporcionaron el primero y más conocido, que posteriormente fue mejorado por Atkin y Morain mediante el que se conoce hoy día como test ECPP (Elliptic curve primality proving).

7. Factorización

En esta sección vamos a analizar algunos algoritmos de factorización y a extraer conclusiones sobre algunas condiciones sobre los primos p y q para que dichos algoritmos no sean eficientes si se aplican a $n = pq$. Esto garantizará la seguridad de los sistemas RSA y de Rabin.

Para obtener la factorización de n vamos a conformarnos con algoritmos que produzcan una una descomposición $n = a \cdot b$ no trivial (es decir, a y b distintos

de 1). La razón es que una vez hecho esto, podemos aplicar el mismo algoritmo de forma recursiva hasta que recuperemos todos los factores primos. Puesto que es mucho más eficiente comprobar si un número es primo que tratar de factorizarlo, se recomienda aplicar primero un test de primalidad para asegurarse de que el número n no es primo.

Existe por supuesto la posibilidad de que n tenga un único factor, elevado a una potencia superior a 1. Afortunadamente, existen métodos capaces de verificar si n es una potencia perfecta a^k, con $k > 1$, por lo que todos los algoritmos que comentaremos en esta sección partirán de la suposición de que n tiene al menos dos factores primos diferentes.

El algoritmo más ingenuo para tratar de factorizar un número n es probar a dividirlo por todos los números enteros positivos comprendidos entre 2 y n. Evidentemente, este método es del todo inaceptable en cuanto n alcanza valores elevados, y ha sido ampliamente mejorado por otras técnicas que, sin llegar a ser realmente eficientes, son mucho más rápidas que la fuerza bruta. En esta sección haremos un breve repaso a algunos de los métodos más interesantes aparecidos hasta la fecha.

7.1. Método de Fermat

Un primer método bastante elemental consistiría en hacer divisiones sucesivas bien por números primos pequeños bien por números primos "cercanos" a $\sqrt{n}$. De hecho no es necesario conocer la primalidad de cada uno de los números cercanos a $\sqrt{n}$, basta calcular el máximo común divisor del candidato y de n. Como consecuencia, una primera precaución es que los primos p y q deben ser ***grandes*** y no deben ser ***cercanos***.

El siguiente método, más sofisticado que el anterior, se debe a Fermat y se aplica a enteros n impares (naturalmente, comprobar la divisibilidad por 2 es trivial).

Supongamos que tenemos dos enteros x e y de tal forma que $x^2 - y^2 = n$. En este caso $n = (x+y)(x-y)$ y, salvo que $x - y = 1$, tenemos una factorización de n. Por tanto, podemos proceder como sigue:

Algoritmo (Fermat)

Inicio: Se hace $x = \lceil \sqrt{n} \rceil$.

Iteración: Si $x^2 - n$ no es un cuadrado perfecto, $x = x + 1$.

Final y salida: Se termina si $x^2 - n$ es un cuadrado perfecto, $x^2 - n = y^2$, entonces devolvemos $x + y$, $x - y$. Obsérvese que en el peor de los casos se llega a $x = \dfrac{n+1}{2}$, $y = \dfrac{n-1}{2}$, correspondiente a la factorización $n = n \cdot 1$. En este caso, se deduce que n es primo.

El algoritmo anterior tiene orden de complejidad exponencial, $O(n)$. Pero en el caso $n = pq$ donde p y q son dos primos cercanos, y como consecuencia cercanos a $\sqrt{n}$, es sin embargo muy rápido. Por tanto, tanto en el RSA como en el sistema de Rabin es conveniente que p y q no sean de la misma longitud. Si tomamos esta precaución el método de Fermat es impracticable debido al alto número de iteraciones que precisaría. En la práctica se debe usar limitando el número de iteraciones, para evitar tener que parar el proceso por medios más drásticos.

7.1.1. Cribas

La antigua idea de utilizar cribas, al modo de la criba de Eratóstenes, para cuestiones de factorización se puede utilizar en el contexto actual, produciendo mayor eficiencia en los algoritmos. Existen varias formas eficaces en el uso de las mismas. Veamos como se pueden utilizar en el método de Fermat que acabamos de describir.

Si queremos comprobar si $x^2 - n$ es o no un cuadrado, una forma de eliminar una cantidad considerable de casos es primero reducir módulo un entero m (que debe ser pequeño) y comprobar si $x^2 - n$ es un cuadrado módulo m o no lo es. Se supone que se ha computado previamente una tabla con los residuos cuadráticos módulo m, con lo que esta última comprobación es rápida. El nombre de criba viene de este hecho: cribamos los x posibles previamente a comprobar si son o no un cuadrado.

En particular, la reducción es considerable si tomamos $p_1, \ldots, p_r$ primos y cribamos sucesivamente mediante ellos, ya que en cada una de las etapas disminuye a la mitad aproximadamente la cantidad de números x que se deben comprobar. Por tanto, para comprobar una lista que a priori consta de M enteros x sólo debemos hacer la comprobación final con $M/2^r$ de ellos.

Podemos programar el algoritmo anterior de la siguiente forma:

Algoritmo (Fermat con cribas)

La entrada del algoritmo está formada por el entero n que se quiere factorizar, y otros dos enteros A, B. A va a limitar el número de comprobaciones y por tanto la complejidad del algoritmo (aunque su uso puede ocasionar que no se llegue a factorizar), y B es una cota para la elección de los primos.

Inicio: Se calculan r primos $p_1, \ldots, p_r < B$. Para $i = 1, \ldots, r$ se calcula el conjunto de cuadrados módulo p_i, $C[i]$ y el conjunto $D[i] := \{z + n \bmod p_i \mid z \in C[i]\}$. Se empieza con $x = \lceil \sqrt{n} \rceil$.

Iteración: Para $i = 1, \ldots, r$:

1. Si $x^2 \bmod p_i \notin D[i]$, se hace $x = x + 1$ y se va a la iteración siguiente.
2. Si $x^2 \bmod p_i = y^2 + n \in D[i]$, y $x^2 \neq y^2 + n$, se hace $x = x + 1$ y se va la iteración siguiente.

Final y salida: Se acaba si se encuentra y tal que $x^2 = y^2 + n$, en cuyo caso se devuelven $x + y$, $x - y$, o si se llega a $x > A + \lceil \sqrt{n} \rceil$, en cuyo caso no hay salida.

7.2. Método $p - 1$ de Pollard

El método $p - 1$ de Pollard es uno de los muchos que utilizan para su ejecución la construcción de una base de factores.

Definición. *Dado un número $B \in \mathbb{R}$, diremos que un número entero m es* B-uniforme *si todos los factores primos de m son menores o iguales que B.*

Supongamos dados n y B, y sean $q_1, q_2, \ldots, q_r$ todos los números primos menores o iguales que B. Para cada i, denotemos por e_i el máximo natural tal que $q_i^{e_i} \leq n$, y sea $m_B = \prod_{i=1}^{r} q_i^{e_i}$.

Supongamos que p es un factor primo de n tal que $p-1$ es B-uniforme. Entonces, es claro que m_B es un múltiplo de $p - 1$, por lo que del pequeño teorema de Fermat se deduce que si a es primo con n (y por tanto con p), entonces $a^{m_B} \equiv 1$ mód p, es decir, $a^{m_B} - 1$ es múltiplo de p. Como n también lo es, escribiendo $x = a^{m^B}$ mód n (resto de la división de a^{m_B} entre n), entonces también $x-1$ es múltiplo de p, así que p divide al $\mathsf{mcd}\,(x - 1, n)$ y por lo tanto $\mathsf{mcd}\,(x - 1, n)$ sería un factor de n.

El algoritmo consiste entonces en, dado n, fijar la cota B, construir m_B y calcular $\mathsf{mcd}\,(x - 1, n)$:

Algoritmo $p - 1$: Se parte de n y se escoge B.

1. Se escoge aleatoriamente $a \in \{2, \ldots, n - 1\}$.

2. Se calcula $d = \mathsf{mcd}\,(a, n)$. Si $d > 1$, entonces d es un factor de n y se acaba.

3. Se calcula $x = a^{m^B}$ mód n. Si $x = 1$, el algoritmo no da respuesta.

4. Se calcula $d = \mathsf{mcd}\,(x-1, n)$. Si $d > 1$, d es un factor de n y se termina. Si $d = 1$, el algoritmo no da respuesta.

La cota B se escoge no muy grande para limitar la complejidad del algoritmo. Si resulta que efectivamente existe un factor primo p de n tal que $p-1$ es B-uniforme, entonces el algoritmo tiene una probabilidad del 50 % de encontrar un valor de a que permita obtener un factor de n. Además, en este caso la complejidad es $O(B \log_B(n))$ operaciones de multiplicación modular.

En relación con la elección de los primos en el RSA o en Rabin, para evitar la factorización por este método es necesario evitar que $p - 1$ ó $q - 1$ tengan todos sus factores primos "pequeños". Como consecuencia, si queremos que los primos p y q

sean "seguros" debe ocurrir que $p-1$ y $q-1$ tengan algún factor primo grande. Una forma posible es que sean del tipo $p = ap' + 1$ y $q = bq' + 1$ para primos grandes p' y q'.

7.3. Cribas cuadráticas

Existen varios métodos, llamados *métodos cuadráticos*, que se basan en encontrar soluciones de la ecuación $x^2 \equiv y^2$ mód n. Si se tiene además $x \neq \pm y$ mód n, entonces n divide a $(x-y)(x+y)$, y n no divide a $x+y$ ni a $x-y$, por lo que n tiene factores comunes con ambos factores. Así, $d = \mathsf{mcd}\,(n, x-y)$ es un divisor no trivial de n.

La diferencia entre unos métodos y otros es la forma de buscar x e y. El método de las cribas cuadráticas (conocido abreviadamente como QS: quadratic sieve) fue introducido por Pomerance en 1982. Dicho método fue el más eficaz hasta comienzos de los 90. Permitió factorizar en un tiempo razonable enteros de unas 120 cifras, aumentando las exigencias de los primos a elegir en el RSA.

La idea del método de las cribas cuadráticas es la siguiente. Partimos de un entero n, compuesto y que sabemos que no es potencia de un primo. Fijamos un entero b y una base $B = \{p_1, \ldots, p_k\}$ formada por los primos $p \leq b$ tales que $\left(\dfrac{n}{p}\right) = 1$. Se añade $p_0 = -1$. Tomamos además $\pm a_i$ tales que $a_i^2 \equiv n \mod p_i$.

Nuestro objetivo es encontrar $x_1, \ldots, x_r$, con $r \leq k+1$, $[\sqrt{n}] - C \leq x_i \leq [\sqrt{n}] + C$ (C fija el *intervalo de cribado*), tales que $x_i^2 - n$ descompone totalmente en B, es decir, $x_i^2 - n = \prod_{j=0}^{k} p_j^{e_{i,j}}$, y $\prod_{i=1}^{r}(x_i^2 - n)$ es un cuadrado. Esto equivale a $e_{1,j} + \cdots + e_{r,j} = 0$ mód 2 para todo j. En este caso, como ocurría en la discusión que hicimos sobre las bases de factores, obtenemos que $\prod_{i=1}^{r} x_i^2$ es un cuadrado, y^2, módulo n que es precisamente lo que buscamos.

El método de cribado aparece precisamente a la hora de calcular los enteros $x_1, \ldots, x_r$. Por la propiedad que les pedimos, se debe tener $x_i^2 - n \equiv 0$ mód p_j si $e_{ij} > 0$, luego en este caso necesariamente $x_i = \pm a_j + kp_j$ para algún $h \geq 1$. Consideramos por tanto los x con $x \geq [\sqrt{n}]$, y para cada $i = 1, \ldots, k$ se eliminan de la lista todos aquellos que no son de la forma $a_i + hp_i$ o $-a_i + hp_i$ para algún i. El método de cribado se puede refinar para detectar las potencias de p_i que aparecen en la factorización de $x^2 - n$.

7.4. Otros métodos de factorización

El sucesor del método de cribas cuadráticas (en cuanto a eficiencia) fue el método de las cribas de cuerpos de números (NFS: number field sieves) desarrollado, a partir de ideas de Pollard, por varios autores (Lenstra, Lenstra Jr., Manasse, Odlyzko y Pomerance entre ellos). Con dicho método, que sigue siendo uno de los más eficaces, se factorizó en 1990 el noveno número de Fermat, $2^{2^9} + 1$.

Además de los mencionados merecen ser destacados los métodos de factorización basados en la estructura de grupo de las curvas elípticas sobre cuerpos finitos (Lenstra).

8. Análisis del sistema RSA

Terminaremos en esta sección el estudio de la solidez del RSA y de las condiciones de Diffie y Hellman analizando posibles ataques y sus consecuencias respecto a las condiciones que se deben tener en cuenta en la elección de las claves.

Por ahora, para evitar la posible factorización de n ya hemos visto la necesidad de que p y q sean grandes, y $p-1$ y $q-1$ no tengan factores pequeños.

8.1. La elección de la clave e

Claves débiles en RSA: En un sistema RSA siempre hay algunos mensajes que no cambian tras ser cifrados. Son aquellos M tales que $M \equiv M^e \pmod{n}$. Esta condición es equivalente, por el teorema chino de los restos, a que $M \equiv M^e \pmod{p}$ y $M \equiv M^e \pmod{q}$. Ahora bien, las soluciones de la ecuación $X \equiv X^e \pmod{p}$ son exactamente $1 + \mathsf{mcd}\,(e-1, p-1)$ y por lo tanto el número de mensajes M que coinciden con su cifrado son

$$(1 + \mathsf{mcd}\,(e-1, p-1))(1 + \mathsf{mcd}\,(e-1, q-1)) \,.$$

Puesto que $e-1$, $p-1$ y $q-1$ son pares, al menos hay 9 mensajes con esta condición. Por supuesto, se pueden calcular todos los mensajes invariantes para evitarlos, pero es mejor elegir e de manera que su número sea el menor posible y, en este caso, la probabilidad de que nos encontremos con uno de ellos es prácticamente cero.

Así pues, lo mejor es tomar e con las condiciones:

1. e es inversible módulo $\varphi(n)$.
2. $\mathsf{mcd}\,(e-1, p-1) = \mathsf{mcd}\,(e-1, p-1) = 2$.

Ataques de exponente bajo: Si el exponente e es demasiado bajo, existen algunas formas de romper el sistema.

La primera se produce por la posibilidad de que $M^e < n$, en cuyo caso $C = M^e$ y basta aplicar un logaritmo para calcular M.

La segunda se debe a la posibilidad de que el mismo mensaje se cifre con claves (n_i, e) para $1 \leq i \leq e$, siendo los n_i primos entre sí dos a dos. Esta es una situación que se puede dar con facilidad si e es pequeño. Por ejemplo, si un banco envía el mismo mensaje a todos sus clientes no sería raro encontrar la coincidencia de e de

los exponentes de la clave pública, y puesto que los n_i se forman como producto de dos primos escogidos de forma aleatoria, lo más fácil es que sean primos entre sí.

Por lo tanto se tienen los mensajes cifrados c_i con $c_i \equiv m^e \mod n_i$. Por el teorema de los restos podemos encontrar $c < n_1 \cdots n_e$ solución de $X \equiv m^e$ mód $n_1 \cdots n_e$. Como $m < n_i$, $m^e < n_1 \cdots n_e$, y evidentemente es también solución de la ecuación, por lo que la unicidad de la solución nos permite deducir $c = m^e$, y por lo tanto calcular m.

8.2. Elección de p y q

Cálculo de la clave de descifrado Un posible ataque es intentar calcular la clave de descifrado d a partir de (n, e). Además del ataque basado en la factorización (o equivalentemente, en el cálculo de $\varphi(n)$), hay que tener en cuenta que además de la clave d elegida puede haber más enteros r tales que $M^{er} \equiv M \mod n$ para todo $M < n$, y por lo tanto sirven para descifrar. Lógicamente conviene asegurarse de que estas posibles claves de descifrado sean las menos posibles, pues en caso contrario un ataque por fuerza bruta podría tener éxito.

La condición $M^{er-1} \equiv 1 \mod n$, se traduce módulo p y q en que $er - 1$ sea múltiplo de $p-1$ y $q-1$, por lo que lo es de su mínimo común múltiplo, γ. Si d_0 es el inverso multiplicativo de e módulo γ, el conjunto de las posibles claves privadas es $\{d_i = d_0 + i\gamma\}$ para $0 \leq i \leq [(\varphi(n) - d_0)/\gamma]$. Como consecuencia cuanto mayor sea γ, menor es el número de claves privadas. Así pues, lo mejor es elegir $p = 2p' + 1$, $q = 2q' + 1$ con p' y q' primos grandes. De esta forma $\gamma = 2p'q'$ y, dependiendo del tamaño de d_0, se tendrá que existen un máximo de dos claves privadas posibles.

Además esta condición sobre p y q también lleva consigo el que $p-1$ y $q-1$ no tengan todos sus factores pequeños, condición necesaria para evitar la factorización.

Pseudoprimalidad de n: Si se pudiese encontrar una base a con $\mathsf{mcd}\,(a, n) = 1$ tal que $n = 2^e d$ es pseudoprimo pero no es pseudoprimo fuerte respecto de a, entonces se podría factorizar n. Esto se debe a que en este caso encontramos un entero $j < e$ de manera que $b = a^{2^j d} \not\equiv \pm 1 \mod n$ pero $b^2 \equiv 1 \mod n$, es decir, una raíz cuadrada módulo n de 1 diferente de ± 1. Por tanto, $(b-1)(b+1)$ es un múltiplo de n y el máximo común divisor de $b-1$ y n proporciona un factor no trivial de n. En particular, los números de Carmichael se pueden factorizar fácilmente, ya que no son pseudoprimos fuertes para la mayoría de las bases pero son pseudoprimos para todas ellas.

Como consecuencia, un riesgo a evitar en la elección de los primos p y q del sistema RSA es que existan demasiadas bases a tales que n es pseudoprimo respecto de a pero no pseudoprimo fuerte. Como la proporción entre ambos tipos de bases es de dos a uno, lo mejor es que n sea pseudoprimo respecto del menor número posible de bases. Analizando el número de soluciones de la ecuación $x^{n-1} \equiv 1 \mod n$ se

puede comprobar que si $(p-1)/2$ y $(q-1)/2$ son primos entre sí, entonces hay cuatro bases a de manera que n es pseudoprimo respecto de a y solo dos, $a = \pm 1$, de manera que n es pseudoprimo fuerte respecto de ellas. Por lo tanto en este caso el sistema es seguro respecto de este tipo de ataques.

A veces, un primo impar p con la condición de que $(p-1)/2$ es primo se llama un primo seguro (por ejemplo en Maple, la función *safeprime* devuelve un primo con tales características). Obsérvese que nos volvemos a encontrar con que la condición $p = 2p' + 1$, $q = 2q' + 1$ con p' y q' primos grandes resuelve este problema.

Ataque de módulo común: Existen algunas situaciones en que se podría pensar en fijar p y q, y variar e y por lo tanto d.

Por ejemplo, si se decide usar un sistema criptográfico basado en el RSA de la siguiente forma: el sistema es el mismo que ya conocemos, pero ahora el entero n es común para todos los usuarios del mismo que, para evitar que los mensajes de uno sean accesibles a los demás, desconocen los factores primos p y q. Este sistema requiere por tanto la existencia de un administrador que distribuye las claves a todos los usuarios. En principio parece un sistema razonable en grupos de usuarios que habitualmente se comunican sólo con el administrador ya que facilita bastante los tiempos de computación del mismo.

Sin embargo este sistema es muy frágil, por ejemplo se podría realizar el siguiente ataque:

Si el texto en claro M se codifica utilizando dos exponentes distintos, los cifrados que se obtiene son $C_1 = M^{e_1} \mod n$ y $C_2 = M^{e_2} \mod n$. Si e_1 y e_2 fuesen primos entre sí (lo cual va a ocurrir en una gran mayoría de casos), mediante el algoritmo de Euclides encontraríamos a y b tales que $ae_1 + be_2 = 1$, y entonces $c_1^a c_2^b = m^{e_1 a} m^{e_2 b} = m$.

Y de hecho, cualquiera de los usuarios con una razonable capacidad de cálculo podría encontrar la factorización de n, escribiendo $ed - 1 = 2^h r$ con r impar y considerando, para un entero a primo con n, la sucesión

$$a^r, a^{2r}, a^{2^2 r}, \ldots, a^{2^h r} \quad \mod n \ .$$

Un análisis de las distintas situaciones que pueden darse garantiza que al menos la mitad de las bases a proporcionan un entero $b = a^{2^j r}$, $j < h$, de manera que $b \not\equiv \pm 1$ mód n y $b^2 \equiv 1 \mod n$. Por lo tanto se podría factorizar n.

Así pues, se deben generar p y q diferentes para cada clave.

8.3. Claves seguras

Tanto los métodos de factorización como la vulnerabilidad a algunos ataques nos han proporcionado ya algunas condiciones sobre los primos p y q, en particular la de que $p-1$ y $q-1$ contengan algún factor primo grande.

Añadamos que también hay algoritmos de factorización relativamente eficientes si $p+1$ y $q+1$ tienen todos sus factores primos pequeños. Por lo tanto debemos elegir dos primos p y q, de manera que $p = ap'+1$ y $q = bq'+1$ para p' y q' primos grandes y de forma que $p+1$ y $q+1$ tengan al menos un factor primo grande.

Hay también métodos de factorización que hacen que los primos sean vulnerables si para cada factor primo r de $p-1$ (o de $q-1$) $r-1$ tiene todos sus factores primos pequeños.

Como consecuencia definiremos ***primo fuerte*** como un número primo p que satisface las siguientes condiciones:

- $p-1 = ar$, siendo r un primo grande y a un entero.

- $r-1 = bs$, siendo s un primo grande y b un entero.

- $p+1 = ct$, siendo t un primo grande y c un entero.

Diremos que una clave RSA $n = pq, e$ es una clave segura si la pareja de números primos (p, q) es una pareja de ***primos seguros***, $(p-1)/2$ y $(q-1)/2$ son primos entre sí y $\mathsf{mcd}\,(e-1, p-1) = \mathsf{mcd}\,(e-1, q-1) = 2$.

Generación de primos fuertes: Un método para generar primos fuertes ha sido descrito por Gordon. en [**?**]. El siguiente procedimiento de Maple produce un número primo de longitud aproximada el k que se pasa como parámetro.

```
> primoFuerte:=proc(k)
    local l,r,s,t,p0,p;
 l:=floor((k-2)/2);
 readlib(randomize)();
 rand(10^(l-1)..10^(l))(); s:=nextprime(%);
 rand(10^(l-2)..10^(l-1))(); t:=nextprime(%);
 r:=2*3*t+1;
 while not isprime(r) do
    r:=r+2*t;
 end do;
 p0:=(2*s&^(r-2) mod r)*s-1;
 if irem(p0,2)=0 then p0:=p0+r*s; end if;
 p:=p0+2*2*r*s; while not isprime(p) do
    p:=p+2*r*s; end do;
 end proc:
```

Es sencillo comprobar que el primo p de salida es congruente con 1 módulo el primo r y con -1 módulo el primo s. Además el primo r se ha construido de

manera que es congruente con 1 módulo el primo t. Los primos s y t son dos primos elegidos al azar con aproximadamente la mitad de los dígitos que indica el parámetro de entrada. En el algoritmo se han utilizado funciones propias de Maple, como `nextprime`, `isprime`,.. cualquiera de ellas es fácil de sustituir por tests de Rabin-Miller con un margen de confianza prefijado (para la función `nextprime` se recorren sucesivamente a, $a+2, a+4, \ldots$, hasta dar con un primo probable). Las operaciones que involucra el procedimiento son de complejidad polinómica, ya conocida, por lo que el algoritmo será de complejidad polinómica siempre que los diferentes bucles se detengan en un número pequeño de pasos. Analizaremos este extremo un poco más adelante.

Nota: En diciembre de 1998, Ronald Rivest y Robert Silverman publicaron un trabajo en el que quedaba demostrado que la aparición de técnicas más modernas de factorización, como la de Lenstra, basada en curvas elípticas, o la criba cuadrática, hacía que se ganase poco o nada con el empleo de este tipo de números primos.

Generación de claves seguras: Para generar ahora una clave segura solamente necesitamos buscar dos primos fuertes y la clave pública e. El siguiente programa tiene en cuenta los requisitos que se han ido detectando en cuanto a ellos.

```
> genClaveSegura:=proc(k)
    local p,q,n,phin,e,d,i,j,m;
 p:=primofuerte(k);
 readlib(randomize)();
 m:=k+rand(5..10)();
 i:=2;
 while i<>1 do
    q:=primofuerte(m);
    i:=igcd((p-1)/2,(q-1)/2);
 end do;
    n:=p*q;
    phin:=(p-1)*(q-1);
    i:=2; j:=3;
    while i<>1 or j<>2 do
      e:=rand(): i:=igcd(e,phin): j:=igcd(e-1,phin):
    end do;
 d:=1/e mod phin;
 return [[n,e], [d,p,q]];
 end proc:
```

Nota: En diciembre de 1998, Ronald Rivest y Robert Silverman publicaron un trabajo en el que quedaba demostrado que la aparición de técnicas más modernas

de factorización, como la de Lenstra, basada en curvas elípticas, o la criba cuadrática, hacía que se ganase poco o nada con el empleo de este tipo de números primos.

8.4. Complejidad

Una vez que disponemos de algoritmos eficientes para generar parejas de primos fuertes y claves seguras, solo nos resta comprobar que son computacionalmente eficientes.

Esta eficiencia reposa en dos teoremas: el teorema de densidad de números primos y el teorema de Dirichlet sobre primos en progresiones aritméticas. En ambos casos la demostración, que utiliza métodos analíticos, queda fuera del alcance de este curso

Teorema de los números primos: Sea $x \in \mathbb{R}$, $x > 0$. Denotamos por $\pi(x)$ el cardinal del conjunto de números primos menores o iguales que x. El teorema de los números primos dice que

$$\pi(x) \simeq \frac{x}{\ln x} \ .$$

Se puede precisar un poco más, ya que se tiene que

$$\frac{x}{\ln x}\left(1 + \frac{1}{2\ln x}\right) < \pi(x) < \frac{x}{\ln x}\left(1 + \frac{3}{2\ln x}\right)$$

para $x \geq 59$. A partir de este resultado se puede estimar que la probabilidad de que un número aleatorio comprendido entre B y $2B$ (para aproximar la longitud binaria que deseamos) sea primo es mayor o igual que $1/2\ln B$. Como consecuencia, usando el k-test de primalidad de Rabin-Miller y elecciones aleatorias de números n con $B < n \leq 2B$, podemos afirmar que el algoritmo correspondiente devuelve un número n que es primo con probabilidad al menos de $1 - (2\ln B)/4^k$. Además el número de operaciones esperadas de dicho algoritmo sera del orden de $\mathcal{O}(k(\log^2 B)M(\log B))$.

Teorema de Dirichlet: El teorema de Dirichlet mencionado establece que si a y d son enteros primos entre sí, entonces la progresión aritmética $a, a+d, a+2d, \ldots, a+kd, \ldots$ contiene infinitos números primos. Además se puede establecer su densidad: si $\pi(x, d, a)$ denota el cardinal del conjunto de primos de la sucesión que son menores que x se tiene que

$$\pi(x, d, a) \sim \frac{\pi(x)}{\phi(d)} \sim \frac{x}{\phi(d)\ln x} \ .$$

El resultado anterior garantiza que los algoritmos de búsqueda de números primos en progresiones de razón d relativamente pequeña son eficientes.