Criptografía de clave pública.

Introducción.

La criptografía de clave pública nace oficialmente con la presentación del trabajo de Diffie y Hellman en 1976. Sin embargo, la desclasificación de ciertos documentos del CESG (Communications-Electronics Security Group), la organización británica encargada de la seguridad en las comunicaciones en el Reino Unido, ha permitido saber que a principio de 1970, James Ellis había propuesto un sistema de clave pública [ELL70] y que en sendos documentos de esta organización del año 1973, se describían dos métodos de cifrado y de intercambio de claves que son prácticamente el RSA[COC73] y el protocolo de intercambio de claves de Diffie-Hellman[WIL74].

Lo importante no es quien descubrió el método, ya que todos lo hicieron de forma independiente, sino la gran cantidad de información de que disponen los organismos gubernamentales y los grandes avances en este tema que es probable posean, pero que solo ve la luz pública en casos como el comentado anteriormente. El caso anterior no es único, con la presentación oficial del criptoanálisis diferencial se supo que el equipo de diseño del DES conocía este tipo de ataque e hicieron resistente al DES contra el mismo, sin embargo, los detalles del mismo y su misma existencia estaban clasificados por el gobierno de los Estados Unidos.

Es probable que organismos como la NSA que disponen del mejor material, tanto técnico como humano, conozcan métodos que la comunidad científica todavía tardará años en descubrir y que sistemas que se consideran totalmente seguros, sean fácilmente forzables por alguna de estas técnicas.

Centrándonos en el tema, la criptografía de clave pública nace por la necesidad de encontrar un método que permita el intercambio eficiente de claves, el gran problema de los sistemas de clave secreta. Los algoritmos de cifrado de clave pública utilizan una clave diferente para el cifrado y para el descifrado y se basan en la utilización de funciones *trampa*, en las cuales un proceso es sencillo de realizar, pero su inverso es computacionalmente muy difícil sin el conocimiento de una información complementaria que se mantiene secreta. Para la implementación de las funciones trampa se suelen utilizar problemas de tipo NP-completo, principalmente de la teoría de los números. Básicamente se basan en alguno de los siguientes problemas:

1. *Factorización*. Factorizar un número en sus componentes primos es uno de los problemas más antiguos de las matemáticas. Prácticamente todos los grandes matemáticos han buscado una forma de resolver el problema de la factorización de una forma rápida, y todos han fracasado hasta el momento. El mejor algoritmo existente en la actualidad es la *criba de campos numéricos* que ha permitido la factorización de números de hasta 150 dígitos. Este método presenta dos fases claramente diferenciadas. En la primera, el proceso de criba propiamente dicho, el proceso puede ser distribuido entre muchos ordenadores. El segundo paso, de resolución de la matriz, necesita del uso de grandes supercomputadores.

2. *Logaritmo discreto*. El problema se basa en encontrar, dados dos enteros positivos *g*, *N* y para un determinado *p* primo, un *n* tal que $N = g^n \pmod p$. Los números *p* y *n* deben ser lo suficientemente grandes, del orden de 300 dígitos. Este problema se considera de una complejidad similar al de la factorización y en muchos casos las ideas utilizadas para la factorización suelen aplicarse a este problema.
3. *Problema de la suma de subconjuntos(mochila)*. Se trata de dado un conjunto de números enteros *M* calcular un subconjunto de ellos cuya suma dé como resultado un numero *N*. El problema es computacionalmente difícil, pero la mayoría de las implementaciones que se han hecho basadas en este problema han sido rotas.
4. *Retículas*. En este caso se utiliza el problema de encontrar el menor vector en una retícula. Definimos una retícula como el conjunto de elementos de la forma $r_1 b_1 + r_2 b_2 + ... + r_m b_m$ con los r_i números enteros y una base de vectores $b_i = \langle v_1, .., v_n \rangle$ con $i = 1, .., m$. El algoritmo más utilizado para resolver el problema es el LLL de Lenstra, Lenstra y Lovasz que encuentra una solución aproximada en tiempo polinomial, y en muchos casos ésta coincide con la mejor. Este algoritmo ha sido utilizado también para romper criptosistemas basados en el problema de la mochila.

A Continuación se presentan alguno de los métodos más importantes, que nos permitirán tener una idea de cómo se implementan y en que basan su seguridad.

Métodos de la mochila (knapsack).

Este conjunto de métodos se basan en la dificultad de resolución del problema denominado de los subconjuntos aunque comúnmente se les denomine de la mochila (knapsack), en el resto del texto se les denominará así ya que es el nombre más aceptado. El problema se plantea de la siguiente manera, dado un conjunto de elementos, determinar si existe un subconjunto de ellos cuya suma dé como resultado un cierto valor. Existe una cierta similitud entre este problema y el de la factorización (determinar de un conjunto de elementos finito si existe un subconjunto tal que su producto sea exactamente el valor buscado). Ambos problemas han sido la base de dos de los algoritmos criptográficos más importantes, en el caso del problema de la suma del método de Merkle-Hellman y en el caso del producto del RSA.

Los métodos de mochila han sido considerados en los últimos tiempos como métodos fácilmente “forzables” debido principalmente a los resultados de Shamir, Lagarias y Odlyzko y Brickell. Todos estos ataques rompen el esquema de Merkle-Hellman, siendo los dos últimos más potentes al permitir romper cualquier sistema basado en un conjunto supercreciente de elementos [PAT87], sin embargo, existen variaciones sobre el método que no han sido rotas todavía tal como se apunta en [DUR98], el principal entre ellos es el de Chor-Rivest que presentaremos más adelante y que puede consultarse entre otras en las referencias [CHO84][MEN97][POM90][PAT87]. Pasamos a estudiar pues el método de Merkle-Hellman y la ampliación del mismo para su utilización en campos de Galois tal como viene definida en [COO84] y [PAT87].

Algoritmo de Merkle-Hellman.

El problema de los subconjuntos, tal como se ha definido anteriormente, es un problema de tipo NP-completo. Sea $P = \{p_1, p_2, ..., p_n\}$ el conjunto de pesos, en total existen

2^n posibles combinaciones de los elementos que lo forman. El tiempo necesario para determinar si una combinación es solución es del orden de *n*, sin embargo el tiempo necesario para encontrarla es de $O(2^{n/2})$ y el espacio requerido de $O(2^{n/4})$.
Más formalmente podemos definir el problema de la siguiente manera. Sea $P = \{p_1, p_2, ..., p_n\}$ el conjunto de pesos y *N* un número determinado, se trata de decidir si existe un subconjunto $S = \{p_{i_1}, p_{i_2}, ..., p_{i_m}\}$ de *P* de forma que $N = p_{i_1} + p_{i_2} + ... + p_{im}$. Por seguir la misma notación que la gran mayoría de los textos de criptografía, utilizaremos la notación de mochila para el conjunto de los p_i y utilizaremos indistintamente los términos problema de la mochila y de los subconjuntos, aunque el problema de la mochila sea un conjunto de problemas mucho más amplio que el que estamos considerando.

Para conjuntos arbitrarios de pesos el problema es muy difícil de resolver, sin embargo, existen casos en los que el problema de los subconjuntos es de fácil solución. Se trata de aquellos casos en los que el conjunto *P* es un conjunto supercreciente.

Sea *P* un conjunto de pesos tal como lo hemos definido anteriormente, se dice que *P* es un conjunto supercreciente si se cumple que para todo $i \leq n$:

$$a_i > a_1 + a_2 + ... + a_{i-1}$$

En el caso de un conjunto supercreciente, la solución a la pregunta de qué elementos lo forman es sencilla, consiste en la resta sucesiva de los elementos a_i menores que el número resultante de las restas anteriores restando siempre de mayor a menor.
Basándose en la propiedad anterior Merkle y Hellman diseñaron el método de cifrado que lleva su nombre. Básicamente consiste en convertir un problema de la mochila fácil basado en un conjunto supercreciente, en un problema difícil. Para ello se realiza una cierta transformación sobre el conjunto original. Los pasos a seguir para cifrar con este método son:

1. Escoger un conjunto *P* de pesos con la propiedad de ser supercreciente.
2. Escoger un número primo *p*, de tal manera que $p > 2.a_n$.
3. Escoger un número *m* en el intervalo $[1, a_n - 1]$, relativamente primo con *p*.
4. Encontrar el inverso de *m* mod *p*.
5. Crear el conjunto $T = \{t_1, t_2, ..., t_n\}$ de pesos transformados mediante la operación $t_i \equiv p_i.m \pmod p$.

Para enviar un mensaje con este método, el emisor divide el mensaje en grupos de *n* bits y calcula $S = b_1.t_1 + b_2.t_2 + ... + b_n.t_n$, donde $b_1, b_2, ..., b_n$ son los bits del mensaje original, y se transmite *S*. El receptor calcula

$$m^{-1}.S = m^{-1}.b_1.t_1 + ... + m^{-1}.b_n.t_n = b_1.p_1 + ... + b_n.p_n$$

a partir de este momento, el receptor puede calcular fácilmente las b_i al tratarse de un problema fácil. Los autores sugieren la utilización de valores de *n* iguales a 100 y que los pesos p_i se escojan en el intervalo $[2^{100+i-1}, 2^{100+i+1}]$, con $p > 2^{201}$ y *m* escogido aleatoriamente en el intervalo $[1, p-1]$.

Utilización de Campos de Galois en el algoritmo de Merkle-Hellman.

En 1.984 Cooper y Patterson proponen una generalización del método de Merkle-Hellman para su utilización aplicando Campos de Galois en un artículo de la revista Cryptologia[COO84]. Lo interesante de este artículo no es el método en sí, que es una simple modificación del de Merkle-Hellman, y por lo tanto susceptible de ser atacado de la misma manera, sino la solución dada a uno de los principales problemas del método anterior, la longitud de clave. Los pasos a seguir para cifrar son:

1) Seleccionar una mochila fácil en el anillo de polinomios sobre los enteros. Todos los elementos de la mochila, que denotaremos $a_i(x)$ son del mismo grado, *k*.
2) Calcular un número primo $p > 2.a_n$.
3) Calcular un polinomio irreducible *I(x)* de grado *k+1*.
4) Escoger un polinomio, $m(x) \leq k$.
5) Generar la mochila fuerte aplicando: $a_i'(x) \equiv a_i(x).m(x) \bmod(p, I(x))$.

Como vemos el método hasta ahora es similar al de Merkle-Hellman, siendo éste un caso particular del que estamos presentando para un valor de $k = 0$. El emisor en este caso cifra los bits del mensaje realizando la siguiente transformación:

$$S'(x) = b_1 a'_1(x) + ... + b_n a'_n(x)$$

Este polinomio *S'(x)* es el que se enviará. El destinatario del mensaje recibirá el polinomio anterior y lo multiplicará por $m^{-1}(x) = m(x) \bmod(p, I(x))$, con lo que obtendrá el mensaje original.

Lo verdaderamente interesante de esta modificación del método original es la posibilidad de eliminar el problema del tamaño de las claves y la utilización de los campos de Galois en el método, que serán la base del sistema de Chor-Rivest. Si consideramos una tabla de 100 elementos, para conseguir un conjunto supercreciente el mayor elemento de la tabla será del orden de 2^{100}. Podemos reducir el tamaño de las claves de la siguiente manera:

1) Decidir el número de bits que se cifrarán en cada paso del algoritmo. Denotamos *N* a este número.
2) Dividir el número *N* en un conjunto de *s* enteros $\{r_1, r_2, .., r_s\}$ tales que $N = r_1 + r_2 + .. + r_s$.
3) Para cada $i = 1, 2, ..., s$ generar secuencias r_i de mochilas fáciles como en el paso anterior. Todos los conjuntos se generarán independientemente de los demás.
4) Generar una matriz *A* de rango *N* x *s* y colocar cada uno de los conjuntos generados en cada una de las columnas de la matriz. Los elementos $a_{i,j}$ de la matriz *A* son los elementos del polinomio $a_i(x) = a_{i,k}x^k + a_{i,k-1}x^{k-1} + ... + a_{i,1}x + a_{i,0}$.

Aunque el sistema es eficiente, no es en absoluto seguro, como reconoce uno de sus autores en [PAT87].

Algoritmo de Chor-Rivest.

Se trata de un algoritmo tipo mochila que no es susceptible de ataque por los mismos métodos utilizados para romper los sistemas anteriores. Desarrollado en 1985 por Chor y Rivest, actualmente se sigue considerando un algoritmo muy seguro. Su principal inconveniente es que si bien el cifrado es una operación rápida, la cantidad de esfuerzo computacional y tiempo requerido para realizar el descifrado hacen que no sea un serio rival con respecto a otros sistemas de clave pública. La principal diferencia con el método de Merkle-Hellman es que no utiliza una mochila supercreciente sino de tipo general y al igual que en el método anterior se utiliza la aritmética en Campos de Galois en lugar de la aritmética modular. Sea *m* el mensaje a cifrar y *c* su equivalente cifrado, el proceso de generación de claves, cifrado y descifrado es como sigue:

GENERACION DE CLAVES

1) Seleccionar un número *p* primo y un número *h* lo suficientemente grandes de forma que $p > h$ y que $q = p^h - 1$ tenga solo factores pequeños. Se recomienda que el tamaño de *p* sea de unos 200 dígitos y el de *h* unos 25. Con estos parámetros podemos generar un campo F_q en el cual el problema del algoritmo discreto es posible.
2) Seleccionar un polinomio mónico irreducible *f(x)* de grado menor o igual a *h*. Los elementos de F_q se representarán como polinomios en $Z_p[x]$.
3) Seleccionar un polinomio primitivo *g(x)* del campo F_q.
4) Calcular los logaritmos discretos $a_i = \log_{g(x)}(x + i)$ con i = 0,.., $p - 1$.
5) Calcular una permutación π del conjunto de elementos {0,1,..*p*-1}.
6) Seleccionar un entero *d* tal que $0 \le d \le p^h - 2$.
7) Calcular $c_i = (a_{\pi(i)} + d) \bmod (p^h - 1)$ con $0 \le i \le p - 1$.
8) La clave secreta es (*f(x),g(x),*π*,d*) y la pública $((c_0, c_1, .., c_{p-1}), p, h)$.

CIFRADO

1) Representar *m* como una serie de bits y dividirla en subseries de longitud

$$\left\lfloor \log \binom{p}{h} \right\rfloor .$$

2) Transformar las subseries generadas anteriormente en el vector binario

$M = (M_0, M_1, .., M_{p-1})$ mediante el siguiente proceso:

i. $l \leftarrow h$.

ii. Para i desde 1 a *p* hacer

$$\text{Si } m \geq \binom{p-i}{l} \text{ entonces hacer} \begin{cases} M_{i-1} \leftarrow 1 \\ m \leftarrow m - \binom{p-i}{l} \\ l \leftarrow l-1 \end{cases}$$

Sino hacer $M_{i-1} \leftarrow 0$

Fin Si.

Fin Para.

iii. $c = \sum_{i=0}^{p-1} M_i c_i \bmod(p^h - 1)$.

DESCIFRADO

1) Calcular $r = (c - hd) \bmod(p^h - 1)$.

2) Calcular $u(x) = g(x)^r \bmod f(x)$.

3) Calcular $s(x) = u(x) + f(x)$ polinomio de grado h sobre Z_p.

4) Factorizar $s(x)$ en factores lineales sobre Z_p. Es decir calcular

$$s(x) = \prod_{j=1}^{h} (x + t_j) \text{ con } t_j \in Z_p \ .$$

5) Los componentes de M que sean 1 tienen índices $\pi^{-1}(t_j) \text{ con } 1 \leq j \leq h$, el resto de los componentes son cero.

6) Para recuperar el mensaje desde M hacemos:

i) $m \leftarrow 0, l \leftarrow h$

ii) Para i desde 1 a p hacer

$$\text{Si } M_{i-1} = 1 \text{ hacer } \begin{cases} m \leftarrow m + \binom{p-i}{l} \\ l \leftarrow l-1 \end{cases}.$$

La comprobación de que el proceso de cifrado y descifrado son inversos es como sigue:

$$u(x) = g(x)^r \bmod f(x) \equiv g(x)^{c-hd} \equiv g(x)^{(\sum_{i=0}^{p-1} M_i c_i)-hd} \bmod f(x) \equiv g(x)^{(\sum_{i=0}^{p-1} M_i (a_{\pi(i)}+d))-hd}$$

$$\equiv g(x)^{\sum_{i=0}^{p-1} M_i a_{\pi(i)} \bmod f(x)} \bmod f(x) \equiv \prod_{i=0}^{p-1} \left[g(x)^{a_{\pi(i)}} \right]^{M_i} \bmod f(x) \equiv \prod_{i=0}^{p-1} (x+\pi(i))^{M_i} \bmod f(x).$$

Ya que $s(x)$ y $\prod_{i=0}^{p-1} (x+\pi(i))^{M_i}$ son polinomios mónicos de grado *h* y son congruentes modulo $f(x)$ tenemos que $s(x) = u(x) + f(x) = \prod_{i=0}^{p-1} (x+\pi(i))^{M_i}$, con lo que aplicando π^{-1} obtenemos las coordenadas de *M* que son 1.

Otros sistemas de clave pública.

El problema de la mochila es solo uno de los múltiples ejemplos de funciones trampa en las cuales es fácil ejecutar la función pero difícil invertir el proceso a no ser que se conozca información complementaria. Se han desarrollado multitud de sistemas basados en problemas matemáticos de tipo NP-completo, el problema de la factorización, el del logaritmo discreto y el del cálculo de raíces discretas. Sin embargo, todos estos sistemas presentan un problema común, el de la velocidad de cifrado, lo que en aplicaciones comerciales los inhabilita para su utilización como sistemas de cifrado de grandes volúmenes de información en línea. Son muy adecuados para la generación de claves y para la firma digital, y en combinación con un algoritmo de clave secreta son quizás la mejor elección para un sistema de cifrado seguro. A continuación se presentan varios de estos algoritmos, de los cuales el más ampliamente utilizado es el primero, el RSA.

Algoritmo R.S.A.

Este método se remonta al año 1978 en que fue publicado por sus autores en el *Communications of the ACM* con el titulo "A method for obtaining digital signatures and public key cryptosystems"[RSA78]. Su nombre proviene de las iniciales de sus autores R. L. **R**ivest, A. **S**hamir y L. **A**dleman. El método se basa en la dificultad de factorizar un número en sus componentes primos. Se trata de un método de clave pública muy seguro y con unos fundamentos matemáticos sólidos. Particularmente se basa en la aplicación de los teoremas de Fermat y la generalización de Euler, que dice que dados *a* y *n* tales que el $\text{mcd}(a,n) = 1$, se cumple que $a^{\Phi(n)} \bmod n = 1$, siendo $\Phi(n)$ la función de Euler que da el número de elementos menores que *n* que son relativamente primos con él. En particular si *n* es primo, $\Phi(n) = n-1$. Como contrapartida, este algoritmo tiene el inconveniente de ser lento comparado con algoritmos de cifrado en flujo e incluso con algoritmos de cifrado en bloque como el DES, lo que impide su utilización como algoritmo de cifrado de grandes volúmenes de información. Esta desventaja ya era apuntada por sus autores en la presentación de su trabajo. Su principal campo de aplicación es como método para cifrar claves de sesión en algoritmos simétricos y de autentificación de los participantes en una comunicación.

Supongamos que queremos cifrar un mensaje *M* de longitud *l* en bloques de longitud *k*, cumpliéndose siempre que *k* < *l*. Los pasos a seguir son:

1. Codificar los símbolos del alfabeto mediante números. Esta representación puede ser en decimal o en binario.
2. Cada usuario participante en la transmisión escoge un par de números primos *p* y *q* de un orden de magnitud de 100 dígitos o superior (en [WAY96] se recomienda aumentar considerablemente esta cantidad, ya que en 1.994 se logro la descomposición factorial de un número de 129 dígitos), de tal manera que su producto *n* = *p.q* cumpla siempre que $N^k < n < N^l$.
3. Calcular la función de Euler $\Phi(n) = (p-1)(q-1)$.
4. Escoger un número *e* relativamente primo con $\Phi(n)$ tal que $1 < e < \Phi(n)$. Al ser el mcd(*e*, $\Phi(n)$) = 1 se cumple que existe un entero *d* tal que $e.d = 1 \bmod \Phi(n)$.

Tomamos pues el par (*n*, *e*) como claves públicas, y (*n*, *d*) como claves privadas. Notemos que el conocimiento de *e* no nos da ninguna información sobre *d*, ya que no sabemos cual es el valor de $\Phi(n)$ y para calcularlo necesitaríamos conocer los valores de *p* y *q*.

En este momento podemos escoger las funciones de cifrado f_c y descifrado f_d, que serán respectivamente:

$$f_c = M^e \bmod n$$

$$f_d = C^d \bmod n$$

Podemos comprobar que efectivamente estos pasos son correctos, tal como presentamos a continuación:

$$f_d(f_c(M)) = (f_c(M))^d = (M^e)^d \bmod n = M^{ed} \bmod n$$

$$f_c(f_d(M)) = (f_e(M))^e = (M^d)^e \bmod n = M^{ed} \bmod n$$

pero al ser $e.d \equiv 1 \bmod n$, tenemos que $M^{ed} \bmod n \equiv M \bmod n$.

La búsqueda de los números primos necesarios para el método puede hacerse utilizando cualquiera de los métodos probabilísticos, tales como el test de Solovay y Strassen, el de Lehmann y Peralta o el de Miller y Rabin. Una descripción detallada de estos métodos puede encontrarse en las referencias [STI95] y [PAT87] y en el apéndice de Teoría de los números.

La obtención de *e* a partir de *d* y $\Phi(n)$ es directa mediante la aplicación del algoritmo extendido de Euclides tal como lo presenta Knuth en [KNU69].

Veamos un ejemplo. En primer lugar debemos definir nuestro alfabeto, que reduciremos a las letras asignándoles un valor numérico, en este caso su posición relativa dentro del alfabeto. En un caso real se utilizaría el ASCII.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Escogemos dos números primos *p* = 17 y *q* = 13. Consecuentemente tenemos que $n = p.q = 17.13 = 221$ y que $\Phi(n) = (p-1).(q-1) = 192$. Escogemos un número $e = 35$ y

calculamos su inverso multiplicativo modulo 192 que nos da como resultado $d = 11$. Ahora ya podemos cifrar el mensaje. Supongamos que queremos cifrar RSA. Su valor numérico sería 19, 20, 01. Ciframos los números obtenidos:

$$f_c = M^e \bmod n = 19^{35} \bmod 221 = 76$$
$$f_c = M^e \bmod n = 20^{35} \bmod 221 = 197$$
$$f_c = M^e \bmod n = 1^{35} \bmod 221 = 1$$

Nuestro mensaje sería 014196001. Vamos a hacer la operación inversa para determinar si nuestro sistema funciona bien. Calculamos el descifrado de los tres números:

$$f_d = C^d \bmod n = 76^{11} \bmod 221 = 19$$
$$f_d = C^d \bmod n = 197^{11} \bmod 221 = 20$$
$$f_d = C^d \bmod n = 1^{11} \bmod 221 = 1$$

Elección de los parámetros en el RSA.

La seguridad del sistema depende en gran manera de una elección adecuada de los parámetros que lo configuran. Como ya se ha indicado anteriormente, los números primos *p* y *q* deben tener una longitud de unos cien dígitos para proveer de una adecuada seguridad. Pero incluso la elección de números primos de esa magnitud no garantiza que el sistema vaya a ser seguro. Es conveniente que los números *p*, *q* y *e* cumplan una serie de características adicionales.

1. *p* y *q* no deben ser demasiado próximos. La razón es la siguiente, supongamos que *p* y *q* son suficientemente próximos y *p* > *q*. Tenemos pues que *n* = *p.q* y que $n = \frac{(p+q)^2}{4} - \frac{(p-q)^2}{4}$. Si denotamos como $x^2 = \frac{(p+q)^2}{4}$, $y^2 = \frac{(p-q)^2}{4}$, tenemos que la solución a la ecuación $n = x^2 - y^2$ viene dada cuando $x^2 - n$ es un cuadrado perfecto. En este caso, $p = x + y$, $q = x - y$. Por esta razón es conveniente que las longitudes de los números primos difieran en algunos bits[PAS98].

2. Los números *p* y *q* deben ser primos fuertes. Se dice que un número *p* es un primo fuerte si cumple las siguientes propiedades:
   a. $p-1$ tiene un factor primo grande que denominamos *r*.
   b. $p+1$ tiene un factor primo grande.
   c. $r-1$ tiene un factor primo grande.

3. El $\text{mcd}(p-1, q-1)$ debe ser pequeño, con lo cual su m.c.m. debe ser grande. Como $p-1$ y $q-1$ son pares tenemos que su producto es divisible por 4. Si el $\text{mcd}(p-1, q-1)$ fuera grande, su m.c.m sería pequeño. Entonces podría calcularse fácilmente un número $i = e^{-1} \bmod m$ que podría utilizarse para descifrar el mensaje.

Algoritmo de Rabin.

El algoritmo de Rabin fue propuesto por éste en 1979 y basa su seguridad en la dificultad de calcular raíces cuadradas modulo un número compuesto que es equivalente al problema de factorizar un número en sus componentes. El algoritmo de Rabin tiene la característica de ser inmune a los ataques con texto en claro escogido, sin embargo, es completamente inseguro ante los ataques con texto cifrado escogido[STI95]. Descripciones del algoritmo pueden encontrarse en el texto anterior, en [MEN97] y en [LUC99] entre otros.

Primeramente se calcula *n* como producto de dos números primos *p* y *q* escogidos aleatoriamente de forma que *p* y *q* sean congruentes con 3 modulo 4. El número *n* será la clave pública, mientras que los números *p* y *q* serán la clave privada. El cifrado del mensaje *M* se realiza mediante la aplicación al mismo de la siguiente formula

$$C = M^2 \bmod n$$

El proceso de descifrado es un poco más complicado y se basa en el método general para encontrar las raíces cuadradas de un número *n* con las características que hemos apuntado en el párrafo anterior.

- Calcular mediante el algoritmo extendido de Euclides los enteros *a* y *b* tales que $ap + bq = 1 \bmod n$.
- Calcular $r = c^{(p+1)/4} \bmod p$.
- Calcular $s = c^{(q+1)/4} \bmod q$.
- Calcular los cuatro posibles mensajes solución que son:
    1. $M_1 = (aps + bqr) \bmod n$.
    2. $M_2 = -(aps + bqr) \bmod n$.
    3. $M_3 = (aps - bqr) \bmod n$.
    4. $M_4 = -(aps - bqr) \bmod n$.

El problema aquí está en decidir cual de los cuatro mensajes es el correcto, con lo cual deberemos mandar alguna información complementaria que nos permita decidir cual de ellos es el mensaje original *M*.

Algoritmo de ElGamal.

El algoritmo de ElGamal fue desarrollado por Taher ElGamal y presentado al público en 1.985 en el IEEE Transactions in Information Theory con el título "*A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*". Se trata de una variante del algoritmo de Rabin que sirve tanto para el cifrado como para la firma digital de mensajes. El algoritmo es sencillo, primeramente se escoge un número primo *p* con *p* grande y un número *n* tal que $1 < n < p$. A continuación cada usuario escoge un número k_s que será su clave secreta y calcula $k_p = n^{k_s} \bmod p$, que será su clave pública junto con *n* y *p*. El proceso de cifrado y descifrado es como sigue.

Sea *M* el mensaje a cifrar que el usuario *A* quiere enviar al usuario *B*. *A* escoge un número aleatorio *a* y recupera la clave pública de *B*, $k_{pB} = n^{k_{sB}} \bmod p$. Calcula $k_{pB}^a = n^{k_{sB}.a} \bmod p$ y $C = M.k_{pB}^a = M.n^{k_{sB}.a} \bmod p$ y envía a *B* el par $(n^a \bmod p, C)$.

El usuario B recibe el par y teniendo en cuenta que $C = M.n^{k_{sB}.a} \bmod p$, tenemos que $M = \dfrac{C}{n^{k_{sB}.a} \bmod p}$ solo tiene que calcular $n^{k_{sB}}.n^{a} \bmod p = n^{k_{sB}.a} \bmod p$ y utilizando el algoritmo extendido de Euclides calcular su inversa.

Criptografía de curvas elípticas.

Los criptosistemas de curva elíptica fueron propuestos por Neil Koblitz y Victor Miller como un método para cifrar la información que en lugar de utilizar números como base del alfabeto, utiliza puntos de una curva. Su principal atractivo reside en que necesita un tamaño menor de la clave para ofrecer una seguridad similar al del algoritmo RSA. Sin embargo algunos autores sostienen que el hecho de necesitar menos bits en la clave es una cuestión inherente al relativamente escaso conocimiento sobre este campo al tratarse de una tecnología relativamente nueva.

Los algoritmos de curva elíptica se basan en la dificultad de resolver el problema del logaritmo discreto, siendo el cálculo aún más difícil que en los cuerpos finitos.

Se define una curva elíptica como el conjunto de puntos representados por una ecuación cúbica de la forma

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

donde *a*, *b*, *c*, *d*, *e* son constantes reales que cumplen una serie de condiciones.

Existe un punto especial denominado generalmente punto cero o punto en el infinito y denotado por *O* que hace el papel de elemento identidad. Sea *P* un punto de la curva, se cumple siempre que $P + (-P) = O$.

Podemos definir las siguientes reglas de adición de puntos sobre una curva elíptica:

1. Se cumple siempre que $O = -O$ y que $P + O = O + P = P$.
2. El inverso de un punto se define de la siguiente manera. Sean $P = (x, y)$, $Q = (x, -y)$ se cumple que $P + Q + O = O$ y por los tanto que $P = -Q$.
3. Para sumar dos puntos *P* y *Q*, se traza una línea entre los dos puntos el resultado será el inverso del punto *R* en el cual la línea corta a la curva elíptica. Es decir se cumple que $P + Q + R = O$.

CONTINUARÁ