CRIPTOGRAFIA Y TEORIA DE NUMEROS

GUILLERMO MORALES-LUNA

FERNANDO VAZQUEZ GARCIA

Sección de Computación, Depto. Ing. Eléc.
CINVESTAV-IPN, México, D.F.

RESUMEN

Se presentan sistemas numéricos para el encriptamiento de mensajes. Primeramente los llamados de llave secreta y después los de llave pública. En los primeros, los procedimientos de codificación y decodificación son fácilmente realizados conociendo sus llaves respectivas. En los segundos, la decodificación de mensajes permanece como un problema difícil aún conociendo la llave utilizada en la codificación. Algunos de estos segundos métodos se basan en las dificultades existentes para factorizar enteros grandes y para decidir si un entero grande es o no un primo. Presentamos métodos para tratar estos últimos problemas.

0. Introducción

Hasta hace diez años, el problema de factorización de números enteros era de competencia exclusiva de especialistas; no se tenía aplicación alguna de interés generalizado sobre este problema hasta 1978, año en que R. L. Rivest, A. Shamir y L. Adleman (RSA) del MIT, publicaron su sistema criptográfico, cuya seguridad esta basada en la enorme dificultad que se presenta al querer factorizar grandes números .

Pero, ¿ cómo se ha llegado a una aplicación inmediata y de gran importancia en la teoría de números ?. Antes de dar contestación a esta pregunta, considérese los siguientes aspectos de la información.

La información puede valorarse de muy diversas maneras. Por ejemplo, podría tener un costo financiero, político, militar y comercial. Un agente extraño podría interceptarla para modificarla e influir así en la toma de decisiones y lograr con ello ciertos objetivos. Esto puede suceder cuando la información no esta físicamente asegurada, es decir, cuando se le transmite por medios de comunicación de acceso compartido que se les considera como inseguros, entre los que se cuentan las líneas telefónicas (que pueden ser intervenidas), la transmisión por radio (microondas, que pueden ser interferidas), etc.. Por lo anterior, el valor de la información puede llegar a ser muy grande.

Es posible que aún por canales considerados como inseguros, la información que sea transmitida resulte incomprensible para una persona que no tenga derecho a recibirla. Se le puede dar un tratamiento de protección, para que únicamente las personas autorizadas puedan recobrar la información original. Lo anterior puede visualizarse como

M = mensaje

Los métodos para lograr lo anterior se enmarcan en lo que se ha llamado *CRIPTOGRAFIA*, la que a grandes rasgos consiste del estudio de sistemas matemáticos para resolver dos clases de problemas de seguridad: *privacidad* y *autentificación*. Un sistema de privacidad debe evitar la extracción de información, a personas no autorizadas, en mensajes trasmitidos por canales considerados como inseguros. Un sistema de autentificación debe prever la transmisión no autorizada de mensajes por medio de canales públicos y que lleguen a personas autorizadas que puedan considerarlo como válido.

Por todas estas consideraciones, la criptografía ha sido de gran importancia en las comunicaciones militares, diplomáticas y recientemente en el ámbito comercial.

Se ha mencionado que el problema de factorización tiene aplicaciones a la criptografía. Este problema es el siguiente:

Dado N, determinar su *representación canónica*, es decir, su representación como producto de números primos.

El problema en sí, consta de dos partes: la primera es determinar si N es un número primo, y luego, si no lo fuera, se trata de determinar a todos sus factores primos. En la actualidad, se cuenta con varios algoritmos para dar solución a este problema. Entre los más importantes se puede citar a los de *búsqueda directa*, al de *diferencia de cuadrados*, a los *métodos de John Pollard*, al de *H. C. Williams* y al algoritmo (que es hasta hoy el más rápido) *de Richard Schroppel*, el cual aún no ha sido publicado. Este último puede factorizar un número N en aproximadamente

$$C \ln N^{\sqrt{\ln(N)/\ln(\ln(N))}}$$

pasos. La siguiente tabla muestra el número de operaciones y el tiempo[1] que emplearía tal algoritmo para factorizar un número compuesto N.

digitos de N	número de operaciones	tiempo
50	1.4×10^{10}	3.9 hrs.
75	9.0×10^{12}	104 dias
100	2.3×10^{15}	74 años
200	1.2×10^{23}	3.8×10^{4} "
300	1.5×10^{29}	4.9×10^{15} "
500	1.3×10^{39}	4.2×10^{25} "

En el primer capítulo presentamos métodos de encriptamiento de llave secreta, en el segundo presentamos métodos de encriptamiento de llave pública, y en el tercero trataremos el problema de factorización de enteros así como aplicaciones de las técnicas vistas.

1. Procedimientos de encriptamiento con llave secreta

En este capítulo presentaremos algunos métodos de encriptamiento determinados por procedimientos específicos que utilizan unos parámetros llamados *llaves*. Más aún, las llaves los determinan y ellas mismas determinan a las *llaves inversas*, las cuales a su vez determinan a los procedimientos de decodificación, motivo por el cual es necesario mantener en secreto a las llaves utilizadas.

1.1. Sea *Alfabeto* = $\langle a_1, \ldots, a_n \rangle$ un alfabeto de n símbolos. El *diccionario* de *palabras* sobre este alfabeto es

[1] La estimación del tiempo está basada en que, cada operación requiere de un microsegundo.

$Alfabeto^* = \bigcup_{k\geq 0} Alfabeto^k$ dónde

$Alfabeto^0 = \langle\ nil\ \rangle$ es la mónada que consta de la palabra *vacía* y

$Alfabeto^{k+1} = Alfabeto \times Alfabeto^k$ es el conjunto de palabras sobre el alfabeto cuya longitud es k+1 , $k \geq 0$.

Sea *Llaves* $\neq \emptyset$ un conjunto no vacío. Un *procedimiento de encriptamiento* es una función

$$ENCRIP: Llaves \times Alfabeto^* \longrightarrow Alfabeto^*$$

tal que $\forall$ Ll $\in$ *Llave* : $m \longmapsto ENCRIP\langle Ll , m \rangle = c$ es una función inyectiva, fácilmente computable y aunque su inversa sea fácilmente computable también, ésta debe ser dificilmente localizada. Si $ENCRIP\langle Ll , m \rangle = c$ diremos que c es el *código* del *mensaje* m mediante la *llave* Ll según ENCRIP.

Vemos pues que un procedimiento de encriptamiento determina una única forma de encriptar siempre que haya quedado fija una llave.

Veamos algunos ejemplos.

1.2. Sea $n = card\langle Alfabeto \rangle$ la cardinalidad del alfabeto. Sea $S_n = \langle\ \sigma: [\![1,n]\!] \longrightarrow [\![1,n]\!] \mid \sigma$ es inyectiva $\rangle$ el conjunto de permutaciones de los primeros n números naturales y sea *Llaves* $= S_n$ este mismo conjunto.

Supongamos al alfabeto enumerado, $Alfabeto = \langle\ a_1 , \ldots , a_n \rangle$. Entonces cada permutación $\sigma \in S_n$ puede identificarse naturalmente con la permutación $Alfabeto \longrightarrow Alfabeto$, $a_i \longrightarrow a_{\sigma(i)}$, la cual puede extenderse a la función biyectiva

$$\sigma^* : Alfabeto^* \longrightarrow Alfabeto^*$$

definida recursivamente por las reglas

$\sigma^*\langle\ nil\ \rangle = nil$

$\sigma^*\langle\ a\ x\ \rangle = \sigma\langle\ a\ \rangle\ \sigma^*\langle\ x\ \rangle$, $\forall\ a \in Alfabeto$, $\forall\ x \in Alfabeto^*$.

Resulta claro que

- $\forall\, k \geq 0 \quad \sigma^*\big|_{Alfabeto^k}$ es una permutación de $Alfabeto^k$ en sí mismo.

- La función de extensión $\sigma \longrightarrow \sigma^*$ es inyectiva.

- $\forall\, \sigma \in S_n \qquad (\sigma^*)^{-1} = (\sigma^{-1})^*$.

Esta construcción nos da un primer procedimiento de encriptamiento ENCRIP: $Llaves \times Alfabeto^* \longrightarrow Alfabeto^*$ haciendo

$$\forall\, \sigma \in S_n \quad \forall\, m \in Alfabeto^* \qquad ENCRIP(\sigma, m) = \sigma^*(m) \ .$$

Debido a la tercera propiedad enlistada anteriormente tenemos que para una llave fija $\sigma \in S_n$, la llave inversa es precisamente σ^{-1}, por tanto, el procedimiento de desencriptamiento no es más difícil que el de encriptamiento siempreque se conozca su llave σ^{-1}, o equivalentemente su inversa σ. Sin embargo, en el caso de desconocer la llave σ, se presentan $n! = card(S_n)$ alternativas para decodificar.

1.3. Como un segundo procedimiento de encriptamiento de llave secreta consideremos el siguiente:

Sea $m > 1$ un número entero. Supongamos esta vez mensajes cuya longitud sea un múltiplo de m.

Consideremos ahora, como conjunto de llaves a S_m. La idea para encriptar consiste en dividir el mensaje en bloques de longitud m y aplicar en cada bloque la permutación correspondiente a la llave. Precisamente, sea enum: $\mathbb{N} \times [\![\, 0 \,,\, m-1 \,]\!] \longrightarrow \mathbb{N}$ la función $(i, j) \longmapsto enum(i, j) = i\, m + j$ que "representa a cada entero en ase m con dos dígitos". Sea

$$ENCRIP:\ S_m \times (Alfabeto^m)^* \longrightarrow (Alfabeto^m)^*$$

definida mediante la relación

$$\forall\, \sigma \in S_m\ \forall\, x = x_0 \ldots x_{k-1} \ , \text{ con } x_i = (x_{enum(i,j)})_{j=0,\ldots,m-1} \ ,$$

$$ENCRIP(\sigma, x) = y \quad \text{donde} \quad y = (x_{enum(i,\sigma(j))})_{0 \leq i < k,\, 0 \leq j < m} \ .$$

Resulta claro que dado $\sigma \in S_m$ la inversa de la función

$x \longmapsto$ ENCRIP(σ , x) es la función $y \longmapsto$ ENCRIP(σ^{-1} , y) .

Como en el caso anterior tenemos que conocida σ, la codificación y la decodificación de mensajes y códigos son igualmente sencillos, en tanto que si se ignora σ, para un código dado habría que considerar $m! = \text{card}(S_m)$ posibilidades de decodificación.

1.4. Veamos un último procedimiento elemental de encriptamiento.

Identifiquemos al alfabeto *Alfabeto* con el anillo $\mathbb{Z}_n$ de los enteros módulo n, donde n = card(*Alfabeto*). Recordamos que, para $m > 0$, $\mathbb{Z}_n^m$ posee una estructura de módulo sobre $\mathbb{Z}_n$. Luego las transformaciones lineales $\mathbb{Z}_n^m \longrightarrow \mathbb{Z}_n^m$ se pueden identificar, respecto a la base canónica de $\mathbb{Z}^m$, con el anillo de las matrices $\mathbb{Z}_n^{m\times m}$. El grupo de unidades de este último anillo es el que nos dará las llaves en este procedimiento de encriptamiento.

Recordamos que una matriz $U \in \mathbb{Z}_n^{m\times m}$ es *unitaria* o *unidad* si $\exists\, V \in \mathbb{Z}_n^{m\times m}$: $U\,V = \mathbb{I} = V\,U$, donde $\mathbb{I}$ es la matriz identidad de $\mathbb{Z}_n^{m\times m}$. Sea pues *Llaves* = $\{ U \in \mathbb{Z}_n^{m\times m} \mid U \text{ es unitaria} \}$. Consideremos como antes mensajes cuya longitud es un múltiplo de m. Esta vez, la idea del encriptamiento consiste en sustituir cada bloque de m caracteres por su imagen bajo la transformación lineal deteremínada por una llave dada.

Definimos ENCRIP: *Llaves* $\times (\mathbb{Z}_n^m)^* \longrightarrow (\mathbb{Z}_n^m)^*$ como
$\forall\, U \in$ *Llaves* $\forall\, x = x_0 \ldots x_{k-1}$, con $x_i = (x_{\text{enum}(i,j)})_{j=0,\ldots,m-1}$,

$$\text{ENCRIP}(U, x) = (U\,x_i)_{i=0,\ldots,k-1}$$

Resulta claro que dado $U \in$ *Llaves* la inversa de la función $x \longmapsto$ ENCRIP(U , x) es la función $y \longmapsto$ ENCRIP(U^{-1} , y) , por lo que la codificación y la decodificación de mensajes y códigos son igualmente sencillos. Sin embargo, desconociendo U, tendríamos que para un código dado habría tantas posibilidades de decodificación como elementos tenga el grupo de unidades de $\mathbb{Z}_n^{m\times m}$

Observamos que una matriz $U \in Z_n^{mxm}$ es unitaria cuando y sólo cuando sus columnas forman una base de Z_n^m. Las m! permutaciones de sus columnas nos darán otras tantas matrices distintas que representan a la misma base de Z_n^m. Así pues tendremos m! p llaves, en donde p es el número de bases, vistas como conjuntos de vectores, distintas de Z_n^m sobre Z_n.

2. Procedimientos de encriptamiento con llave pública

En el capítulo anterior hemos visto procedimientos de encriptamiento tales que conocida la llave utilizada para codificar, el cálculo de la llave inversa es directo y ésta da, con el mismo procedimiento de encriptamiento, la manera de decodificar códigos.

En este capítulo veremos algunos procedimientos tales que, bien que sean conocidos y que se hace pública una llave particular para codificación, los procedimientos de decodificación se mantienen ocultos, o más precisamente, el localizarlos requiere una gran cantidad de tiempo.

En los procedimientos de *llave pública*, un receptor de mensajes hace públicas las maneras en las que hay que codificar los mensajes que se le quieran enviar. Aún cuando los procedimientos, las llaves y los códigos generados sean conocidos, únicamente el receptor es capaz de decodificar a los códigos para recobrar los mensajes.

2.1. Procedimiento de Merkle-Hellman: Recordemos el célebre **Problema de la alforja (Knapsack):** Dados $a = (a_1,\ldots,a_n) \in \mathbb{N}^n$ y $S \in \mathbb{N}$ decidir si existe un conjunto de índices $X \subset [\![1 , n]\!]$, y en su caso encontrarlo, tal que $\sum_{i \in X} a_i = S$. Equivalentemente se trata de encontrar un vector $x = (x_1,\ldots,x_n) \in \langle 0 , 1 \rangle^n$, con entradas 0 o 1, tal que $\langle a \mid x \rangle = S$, donde $\langle \mid \rangle$ denota al producto interno usual de vectores de dimensión n.

La fama de este problema se debe a su *completitud-NP*: Dada una posible solución, mediante una suma de a lo más n sumandos, comprobamos si en efecto es una solución; en tanto que para localizar una solución habria que revisar hasta 2^n posibilidades. Además cualquier otro problema, que posea un procedimiento de comprobación de soluciones en tiempo polinomial respecto al tamaño de las instancias presentadas, se puede reducir a este problema de la alforja.

Un caso sencillo de este problema es el que corresponde a sucesiones $a = (a_1,\ldots,a_n) \in \mathbb{N}^n$ que son de *crecimiento acumulativo*. Es decir, a aquellos en los que cualquiera de sus entradas es mayor que la suma de las anteriores, en simbolos

$$\forall\, k \in [\![\, 1\ ,\ n-1\ ]\!] \qquad\qquad a_{k+1} > \sum_{i=1}^{k} a_i$$

En este caso el problema es sencillo, pues dado el valor de una suma $S \in \mathbb{N}$, consecutivamente le iremos sustrayendo las entradas mayores, en el vector de crecimiento acumulativo que estén por debajo de los resultados de las sustracciones. Si algún resultado se anulara entonces tendríamos una solución, dada precisamente por los índices de las entradas que fueron sustraídas. En caso contrario no habrá solución.

Más precisamente, el algoritmo de solución es el siguiente:

Algoritmo AlfCreAcu (Alforja con Crecimiento Acumulativo)

Entradas:
- $a = (a_1,\ldots,a_n)$: Vector con crecimiento acumulativo
- $S \in \mathbb{N}$: Valor de una suma

Salida :
- $x \in \{0,1\}^n$: $S = \langle x|a \rangle$; si tal x existe
- *'No exite solución'* ; en otro caso

begin

```
    T := S  ;
    x := 0  ( hacemos x igual al vector nulo )  ;
    while  ∃ j : a_j ≤ T  do
    begin
        i := max( j | a_j ≤ T )  ;
        x[ i ] := 1  ;
        T := T - a_i
    end
    if  T = 0  then   return  x
               else   'No existe solución'
end
```

El problema de la alforja define un procedimiento de encriptamiento dado de la siguiente manera:

Consideremos *Alfabeto* = ⟨ 0 , 1 ⟩ y, como conjunto de mensajes, a $Alfabeto^n$, es decir, los mensajes serán las palabras de longitud n que constan de símbolos 0 , 1. Las llaves en este caso serán los vectores cuyas entradas son enteros positivos, $Llaves = (\mathbb{N} \setminus \{0\})^n$. La función de encriptamiento es entonces

$$\text{ENCRIP}: Llaves \times Alfabeto^n \longrightarrow \mathbb{N}$$
$$\langle a , x \rangle \longmapsto \text{ENCRIP}\langle a , x \rangle = \langle a \mid x \rangle$$

Para llaves arbitrarias, el decodificar presenta instancias del problema general de la alforja y consecuentemente es una tarea difícil aún para el receptor que hiciera pública la llave a. Por otro lado, si la llave a es de crecimiento acumulativo entonces cualquier persona que la conozca y que conozca el código generado podrá decodificarlo fácilmente.

La idea de este procedimiento de encriptamiento de llave pública consiste en elegir una llave que casi sea de crecimiento acumulativo pero que no lo parezca. El receptor de mensajes elige una llave de crecimiento acumulativo, la traduce a una que no lo sea y la hace pública para recibir mensajes codificados como arriba se mostró. Cuando él recibe un código, la correspondiente instancia del problema de la alforja con la llave pública se traduce a una instancia del problema de la alforja correspondiente

a la llave de crecimiento acumulativo elegida inicialmente y es posible entonces decodificarlo fácilmente. Por supuesto, la conversión de la llave pública a la de crecimiento acumulativo sólo es conocida por el receptor de mensajes.

La construcción de la llave pública es pues la siguiente:

Elijamos una sucesión $a = (a_1,\ldots,a_n) \in \mathbb{N}^n$ de crecimiento acumulativo y dos números primos p , q tales que $p > 2a_n$ < q. Sea $b \in \mathbb{N}^n$ el vector tal que $b_i = p\, a_i \bmod q \quad \forall\, i = 1,\ldots,n$. b es una llave que no es de crecimiento acumulativo y por tanto se podrá hacer pública. El receptor de mensajes anunciará: "Cualquier persona que desee enviarme un mensaje x en secreto ha de enviarme el código $S = \langle\, b \mid x \,\rangle$, donde $b \in \mathbb{N}^n$ es el vector $b = \ldots$ ". El algoritmo de decodificación que utilizará el receptor es el siguiente:

Entrada: $T \in \mathbb{N}$: Código del mensaje con la llave pública b.

Salida : $x \in \{0,1\}^n$: Mensaje correspondiente a la llave T.

```
begin
     S := p^-1 T mod q          ( p^-1 es el inverso
                                  multiplicativo de p en Z*_q ) ;
     x := AlfCreAcu( S , a )
end
```

Vemos que bien que la llave b se hace pública, los primos p y q y el mismo vector a, que no son necesarios para la codificación de mensajes, han de mantenerse en secreto pues ellos dan la clave para decodificar.

Ejemplo: Sea $a = (1,2,3,6,12,24) \in \mathbb{N}^6$. a es un vector de crecimiento acumulativo. Sean $p = 47$ y $q = 53$. Se tiene que $p^{-1} = 44$, pues $47 \times 44 = 2068 = 39 \times 53 + 1 = 1 \bmod 53$. La llave pública es entonces

$b = (47 \times 1,\ 47 \times 2,\ 47 \times 3,\ 47 \times 6,\ 47 \times 12,\ 47 \times 24) \bmod 53 =$

$= (\ 47\ ,\ 41\ ,\ 35\ ,\ 17\ ,\ 34\ ,\ 15\ )$.

Ahora, si recibiéramos el código T = 116, calculamos $S = p^{-1} T \bmod q = 44 \times 116 \bmod 53 = 16$. Al resolver el problema de la alforja para S = 16 con el vector a elegido inicialmente obtenemos como solución x = (1 , 0 , 1 , 0 , 1 , 0) y vemos que en efecto

$$16 = S = a_1 + a_3 + a_5 = 1 + 3 + 12 \qquad y$$
$$116 = T = b_1 + b_3 + b_5 = 47 + 35 + 34 \qquad \blacksquare$$

2.2. Procedimiento de Diffie - Hellman

Sea $p \in \mathbb{N}$ un primo suficientemente grande. En este método identificaremos al conjunto de mensajes con el grupo de unidades de $\mathbb{Z}_p$, $\mathbb{Z}_p^* = \{ x \in \mathbb{Z}_p \mid \exists\, y \in \mathbb{Z}_p : x\,y = 1 \}$. Si tuviéramos mensajes dados por palabras de caracteres ASCII entonces podríamos dividirlos en bloques de caracteres de longitud $\leq \log p$). Cada uno de los cuales puede naturalmente identificarse con un elemento de $\mathbb{Z}_p^*$.

Por el Pequeño Teorema de Fermat tenemos que

$$\forall\, x \in \mathbb{Z}_p^* \qquad x^{p-1} = 1 \mod p$$

y consecuentemente

$$\forall\, x \in \mathbb{Z}_p^* \quad \forall\, m \in \mathbb{N} : \quad m = 1 \mod (p-1) \quad \Rightarrow \quad x^m = x \mod p$$

Luego, si $c_1 , d_1 , c_2 , d_2 \in \mathbb{Z}_p^*$ son tales que $c_1\, d_1 = 1 \mod (p-1)$ y $c_2\, d_2 = 1 \mod (p-1)$ entonces

$$\forall\, x \in \mathbb{Z}_p \qquad x^{c_1 d_1 c_2 d_2} = x \mod p$$

El procedimiento de encriptamiento consiste, esta vez, en que un receptor de mensajes hace público el número p y le pide a cualquier persona que quiera enviarle un mensaje x codificado que elija dos números c_2 , d_2 tales que $c_2\, d_2 = 1 \mod (p-1)$. El envío del mensaje codificado lo ha de hacer mediante dos transmisiones y una recepción de valores: La persona que quiere enviar el mensaje $x \in \mathbb{Z}_p^*$ transmite primero el valor $y = x^{c_2}$, recibirá en cambio un valor $z \in \mathbb{Z}_p^*$ transmitido por el receptor y

transmite luego el valor $w = z^{d_2}$. Al conocer w el receptor puede recobrar el valor de x.

En efecto, así como cada enviador de mensajes elige c_2 , d_2 tales que $c_2\ d_2 = 1 \mod (p-1)$ también el receptor elige c_1 , d_1 tales que $c_1\ d_1 = 1 \mod (p-1)$. Al hacer $y = x^{c_2}$, $z = y^{c_1}$ y $w = z^{d_2}$ tendremos que

$$w^{d_1} = (((x^{c_2})^{c_1})^{d_2})^{d_1} = x^{c_2\ d_2\ c_1\ d_1} = x \mod p$$

En principio los valores públicos de p , y , z , w determinan los valores de c_1 y d_2 , los que a su vez determinan los de d_1 y c_2, los que, junto con w, determinan el valor de x. Sin embargo el cálculo de c_1 y d_2 a partir de p , y , z , w es una tarea harto complicada. Es aquí donde radica la seguridad de este procedimiento de encriptamiento con llave pública p.

Sea $b \in \mathbb{Z}_p^*$. El *logaritmo discreto en base* b de un elemento $x \in \mathbb{Z}_p^*$ es el elemento $\log_b x = m \in \mathbb{Z}_{p-1}$ tal que $x = b^m$. Por el Pequeño Teorema de Fermat y por el hecho de que $\mathbb{Z}_p^*$ es cíclico en cuanto a su estructura multiplicativa, tendremos que el logaritmo discreto de x en base b es único, visto como un elemento de $\mathbb{Z}_{p-1}$.

Si tuviéramos un mecanismo para el cálculo de logaritmos discretos, el procedimiento de encriptamiento arriba descrito podría romperse observando que, con la notación introducida:

$$x = w^{[\log_y z]^{-1}_{\mathbb{Z}_{p-1}}} \mod p$$

donde, naturalmente, $[\ ]^{-1}_{\mathbb{Z}_{p-1}}$ es la función *inverso multiplicativo* de $\mathbb{Z}_{p-1}^*$.

El cálculo de logaritmos discretos es muy complicado. Esto a diferencia del de los logaritmos reales, el cual está dado por el desarrollo en serie de Taylor de la función logaritmo. La complicación en el cálculo de los logaritmos discretos es debida a que, esencialmente, tendrá que realizarse de manera exhaustiva: Dados x y b, hay que calcular consecutivamente las potencias b^m y

detenerse cuando se haya topado con el valor de x.

El algoritmo de Silver, de complejidad $O(\sqrt{p})$, que presentamos a continuación, puede considerarse como un algoritmo "rápido" para calcular logaritmos discretos. La idea en él consiste en dividir primeramente a Z_p^* en $[\sqrt{p}] + 1$ listas de longitud hasta $[\sqrt{p}] + 1$, llevando el recuento de cada una de ellas por sus primeros elementos. Luego, al formar otra de $[\sqrt{p}] + 1$ elementos consecutivos, la intersección de esta segunda lista con la formada por los primeros elementos de las anteriores, nos dará el logaritmo buscado. Precisamente:

Entrada: Un primo $p \in \mathbb{N}$, una base $b \in Z_p^*$ y un elemento $x \in Z_p^*$.
Salida : El logaritmo discreto $m = \log_b x$.

begin
    $k := [\sqrt{p}] + 1$;
    for i =1 **to** k-1 **do**
    begin
        ListConsec[i] := $x\, b^{i}$;
        ListBloque[i] := b^{ik}
    end ;
    escoge i,j∈[0,k-1] tales que ListConsec[i] = ListBloque[j] ;
    m := (jk - 1) mod (p-1)
end

2.3. Esquema de Rivest-Shamir-Adleman (RSA)

Se ha visto la infactibilidad de factorizar números de más de 200 digitos. La seguridad del esquema RSA está basada en esta dificultad.

Los mensajes toman forma de números en el rango [0,n-1]; donde n es el producto de dos números primos p y q de no menos de 100 digitos cada uno, de esta forma se asegura que n será de a lo menos 200 dígitos. El número n se hace público.

Sea $r = (p-1)\cdot(q-1)$, debe determinarse un entero d mayor al

máximo de p y q, tal que (d,r)=1, en seguida se calcula el inverso multiplicativo e de d módulo r.

El mensaje M deberá ser partido en bloques, cada uno de ellos será un entero entre 0 y n-1. El mensaje encriptado C a partir de M, se calcula como $C = M^e \bmod n$ (1)

Para recuperar M a partir de C , se calcula $M = C^d \bmod n$ (2)

Puesto que se hace uso de exponenciaciones, es conveniente realizarlas mediante el siguiente algoritmo, conocido como *"exponenciación por cuadrados repetidos y multiplicación"* :

Algoritmo POTENCIAS
Entrada: Dos enteros e y C.
Salida : La potencia $D = C^e \bmod n$.
begin
escribamos $e = (e_k\ e_{k-1}\ \ldots\ e_1 e_0)_2$ (representación binaria de e) ;
D := 1 ; F := C ;
for i = 1 **to** k **do**
begin
if $e_i = 1$ **then** D := D F mod n ;
$F := F^2$
end
end

Ejemplo: Sean p = 661 y q = 691, así n = p·q = 456751, por tanto r = 455400 . Sea d = 709. Se tiene m.c.d.(d , r) = 1 y por tanto e = 83589.

Con n = 456751, que tiene seis dígitos, el mensaje M puede ser partido en bloques de tres caracteres. Convengamos en que cada carácter alfabético se representa por un número de la siguiente forma: blanco = 10 , A = 11 , B = 12 , . . , Z = 36.

Encriptemos el mensaje M = "AQUI NADIE VIVIRA PARA SIEMPRE". Partido queda como 112731 191024 111419 151032 192219 281110

261128 111029 191523 262815. Ahora, la representación binaria e es e = 63589 = 1111100001100101 , así, empleando el algoritmo anteriormente descrito para encriptamiento de mensajes, M resulta

$$M_1 = 112731 \longrightarrow C_1 = 380598$$
$$M_2 = 191024 \longrightarrow C_2 = 133250$$
$$M_3 = 111419 \longrightarrow C_3 = 456675$$
$$M_4 = 151032 \longrightarrow C_4 = 257835$$
$$M_5 = 192219 \longrightarrow C_5 = 26730$$
$$M_6 = 281110 \longrightarrow C_6 = 151001$$
$$M_7 = 261128 \longrightarrow C_7 = 438504$$
$$M_8 = 111029 \longrightarrow C_8 = 291568$$
$$M_9 = 191523 \longrightarrow C_9 = 203190$$
$$M_{10} = 262815 \longrightarrow C_{10} = 332129$$

Por tanto C = 380598 133250 456675 257835 26730 151001 438504 291568 203190 332129.

Para el desencriptamiento lo único que cambia entre (1) y (2) es M por C y d por e; de esta forma es posible utilizar el mismo algoritmo de encriptamiento para desecriptar C, empleando ahora d en lugar de e.

3. Métodos de Factorización de Enteros y Generación de Primos

3.1. Método de Pollard

A mediados de los años 70's, John Pollard dió a conocer un método que ha significado un gran avance en el problema de factorización, el llamado *método "p-1"*, que puede, en ocasiones, determinar un factor primo p de N muy rápidamente. El método es eficiente siempre que p-1 es el producto de pequeños factores primos, es decir cuando $p - 1 = \prod_{i=1}^{k} q_i^{\alpha_i}$, donde $\forall\, i=1,\ldots,k$ q_i es un número primo pequeño y $\alpha_i \in \mathbb{N}$.

La idea de este método consiste en extraer información de un elemento a del grupo multiplicativo Z_N^*. Supóngase que p es un factor primo de N, por el Teorema Pequeño de Fermat, tenemos que $a^{p-1} \equiv 1 \bmod p$, ahora si $m.c.d.(Q, p) = 1$ y $p-1 \mid Q$ entonces $a^Q \equiv 1 \bmod p$. En particular consideremos $Q = \prod_{i=1}^{m} Q_i$ donde cada $Q_i = q_i^{\beta_i}$ es tal que $Q_i \leq B$ pero $q_i Q_i > B$, siendo B una cota superior no muy grande del conjunto $\{ q_i^{\alpha_i} \mid i = 1, \ldots, m \}$. Ahora, el cálculo de p se efectúa como sigue: Tómese un a tal que $(a, N) = 1$, hagamos primeramente $R_0 = a$ y posteriormente $R_{i+1} = R_i^{Q_{i+1}} \bmod N$, $i=0, \ldots, m-1$. Calculamos $A = (R_{m-1}, N)$. Por lo general ocurre que A es primo y hacemos $p = A$. Si A es un número compuesto, éste divide exactamente a N, y por tanto, se aplica nuevamente el método de Pollard, sustituyendo a N por A. En caso de que $A = 1$, se procede a ejecutar el segundo paso del que consiste el método de Pollard. En él es posible determinar un factor primo p de N, si es que p-1 es el producto de pequeños primos y un primo de mayor tamaño, es decir, si $p-1 = s \prod_{i=1}^{k} q_i^{\alpha_i}$ donde s es un número primo. Con argumentos similares a los anteriores, puede verse que en este caso $p \mid (a^s - 1, N)$. En cuanto al segundo paso, se hace uso de una segunda cota superior C tal que $B < s \leq C$, donde B es la cota del primer paso. Ahora bien si N es un entero muy grande entonces C también lo será. Esto implica que se deben conocer los primos s_j, $j = 1, \ldots, k$, que estén dentro del intervalo $]B, C]$. Hagamos $2d_j = s_{j+1} - s_j$, $j = 1, \ldots, k$. Ahora definamos, inicialmente $b_1 = R_m^{s_1} \bmod N$, donde R_m es el último módulo calculado en el paso 1, y consecutivamente $b_{j+1} \equiv R_m^{2d_j} b_j \bmod N$. Una vez obtenidos los b_j, se calcula $A_t = m.c.d.(\prod_{i=0}^{c} (b_{t+i} - 1), N)$, $t = 1, c+1, 2c+1, \ldots, [C/c]c+1$, donde c es una constante entera mayor o igual a 1, que indica el número de multiplicandos a entrar en cada cálculo. A_t es bien un factor primo de N o bien $A_t = 1$ o $A_t = N$ en cuyos dos últimos casos habrá de volver intentarse factorizar a N en un nuevo intervalo $]B, C]$.

Este método fué programado en lenguaje LISP y como prueba a dicha implementación se consideraron entre otros, a los siguientes números enteros

1. $N = 2^{98} - 1$

Paso 1: Con a = 2 y B = 10 , se obtuvo A = 7, el que resultó ser un factor. Sustituimos N := N / 7. Después de varios intentos en este paso, no fué posible determinar otro factor primo de N. Procedimos entonces al

Paso 2: El último residuo calculado fué $R_m = 4$. Así c = 3 , B = 2 , C = 100 , y entonces $A_t = 2147483647$, que es un nuevo factor. Por tanto $N = 7 \cdot 2147483647 \cdot 658812288653553079$. ≣

2. $N = 2^{133} - 1$

Paso 1: Se dió a = 1 , B = 10. Se obtuvo el factor A = 127. Sustituimos N := N / 127.

Paso 2: $R_m = 4$, $c_1 = 3$, B = 2 , C = 100 . Así $A_t = 524287$. Por tanto $N = 127 \cdot 524287 \cdot 1635372208527253988514343257209 59$. ≣

3. $N = 2^{132} - 1$

Se logró factorizarlo haciendo uso solamente del paso 1, obteniendo $N = 3 \cdot 3 \cdot 5 \cdot 7 \cdot 13 \cdot 23 \cdot 67 \cdot 89 \cdot 397 \cdot 683 \cdot 2113 \cdot 20857 \cdot 312709 \cdot 599479 \cdot 4327489$. ≣

4. $N = 2^{220} - 1$

A este número N no fué posible factorizarlo por completo mediante los dos pasos, pues se tendría que emplear un intervalo] B , C] más grande o intentarlo con algún otro método.

$N = 3 \cdot 5 \cdot 5 \cdot 11 \cdot 11 \cdot 23 \cdot 31 \cdot 41 \cdot 89 \cdot 397 \cdot 683 \cdot 881 \cdot 2113 \cdot 2971 \cdot 3191 \cdot 201961 \cdot$
$\cdot 7384233730933633527846074574023 91$.

El último término es precisamente lo no factorizado. ≣

El método de Pollard requiere de aproximadamente $O(N^{(1/2)\alpha+\delta})$ operaciones para su próposito, siendo $0 < \alpha < 1$ y $\delta > 0$.

3.2. Algoritmos para la generacion de numeros primos

En el presente trabajo, para mostrar la utilidad de los números primos a la Criptografía, se ha hecho uso de dos algoritmos probabilisticos para generar grandes números con una fuerte certeza de ser primos, esto es, cuando los algortimos determinan que un número N es compuesto, la aseveración es siempre cierta, pero cuando determinan que el número es primo, se hace con una cierta probabilidad de error, que aunque es menor a 1/2 siempre existira la posibilidad de trabajar con un número compuesto considerándolo como un número primo.

El primer algoritmo que se presenta, fué ideado por R. SOLOVAY y V. STRASSEN en 1977, en él, el número N a ser probado no tiene forma especial alguna. Presentamos el algoritmo descrito en un seudo-código.

Algoritmo PROBABILISTICO de PRIMOS

Entrada: Un entero impar N mayor que la unidad

Salida : { 'Número compuesto' si N es compuesto
'Número primo' si N es probablemente primo

```
begin
    sean a_1,...,a_k números enteros, escogidos aleatoriamente
                en el intervalo [ 1 , N-1 ] ;
    for i = 1 to k do
    begin
        if m.c.d.( a_i , N ) = 1 then
        begin
            E = a_i^((N-1)/2) mod N  ;
            F = ( a_i / N )  ;
            if E = F then  número = "Número_primo";
                      else  número = "Número_compuesto"
        end
        else número = "Número_compuesto"
    end  ;
    imprime(número)
end
```

Aquí $\left(\frac{a_i}{N} \right)$ representa al símbolo de JACOBI.

Antes de exponer el algoritmo para el cálculo del símbolo de Jacobi, se hace necesario dar algunas definiciones y resultados que sobre él se tienen.

En la teoría de números se estudian varios tipos de congruencias, en partícular las congruencias cuadráticas, es decir, congruencias de la forma $x^2 \equiv a \bmod m$ donde a es tal que m.c.d.(a , m) = 1 ; si esta congruencia admite solución, a se dice ser un *residuo cuadrático*, en caso contrario será un *residuo no-cuadrático*.

El símbolo de LEGENDRE , denotado por $\left(\frac{a}{p} \right)$ con m.c.d.(a, p) = 1, se define

$$\left(\frac{a}{p} \right) = \begin{cases} 1 & \text{si } a \text{ es un residuo cuadrático} \\ -1 & \text{en otro caso.} \end{cases}$$

Sea P un número impar y sea $P = p_1 \cdots p_r$ su descomposición canónica, supongase también que m.c.d.(a , P) = 1 . El símbolo de Jacobi se define como el producto

$$\left(\frac{a}{P} \right) = \left(\frac{a}{p_1} \right) \cdots \left(\frac{a}{p_r} \right)$$

El algoritmo de Solovay y Strassen queda justificado por el siguiente criterio, debido a Euler.

Teorema: Si m.c.d.(a , p) = 1 y si p es un número primo impar, entonces $\forall\, a \in \mathbb{Z}$ $\left(\frac{a}{p} \right) = a^{(p-1)/2} \bmod p$

El símbolo de Jacobi puede ser calculado haciendo uso de la *ley recíproca*.

Teorema: Si m y n son enteros positivos impares y si m.c.d.(a , m) = 1 = m.c.d.(b , n) = 1 entonces

i. $\left(\frac{a}{m}\right)\left(\frac{b}{m}\right) = \left(\frac{a\,b}{m}\right)$

ii. $\left(\frac{1}{n}\right) = (-1)^{(n-1)/2}$

iii. $\left(\frac{1}{n}\right) = (-1)^{(n^2-1)/8}$

iv. $\left(\frac{m}{n}\right) = (-1)^{(m-1)(n-1)/4}\left(\frac{n}{m}\right)$

Se presentan dos algoritmos para el cálculo del símbolo de Jacobi, el primero recursivo resulta ser muy elegante, pero ineficiente cuando a_i y N son muy grandes, el otro es iterativo.

Algoritmo Jacobi (Forma recursiva)

Entrada: a_i y N

Salida : $S = \left(\frac{a_i}{N}\right)$

```
begin
     if a_i = 1 then sj := 1  else
     begin
     if a_i es par then  sj := Jacobi( a_i / 2 , N )*(-1)^((N^2-1)/8)
                   else  sj := Jacobi(N mod a_i, a_i)*(-1)^((a_i-1)(N-1)/4)
     end ;
     S := sj
end
```

Algoritmo Jacobi (Forma iterativa)

Entrada: a_i y N

Salida : $S = \left(\frac{a_i}{N}\right)$

```
begin
     u  := 0  ; i1 := 0  ; u1 := 0 ; x1 := 0 ; x2 := 0 ; z := 0  ;
     m  := 0  ; ai := a_i ; nn := N mod 8     ; n1 := N ;
     while n1 > 1 do
     begin
          i1 := 0;
```

```
        while  a1 = u * 2 do
        begin
             u  := a1 div 2   ;
             i1 := i1 + 1     ;
             a1 := u
        end  ;
        if i1 es impar then      m := m (nn^2-1) div 8  ;
        u1 := a1 mod 4 ;  x1 := nn-1     ;  x2 := u1-1 ;
        m  := (x1*x2) div 4 + m      ; z  := n1 mod a1 ;
        n1 := a1       ; a1 := z     ; nn := n1 mod 8  ;
        if n1 ≤ 1 then
        begin
             m := m mod 2  ;
             if m = 0 then sj := 1
                      else sj := -1
        end
    end
end
```

El segundo algoritmo que se presenta, fué ideado por MICHAEL O. RABIN. Antes de presentar el algoritmo en seudo-código, se dan algunas definiciones y resultados que muestran la efectividad de dicho algoritmo.

Definición: Sea N un entero, se define el predicado $\Psi(N,b)$ como la conjunción de la primera con la disyunción de las dos últimas de las siguientes proposiciones

i) $1 \leq b \leq N$

ii) $b^{N-1} \not\equiv 1 \bmod N$

iii) $\exists\, i : 2^i \mid (N-1) \quad y \quad 1 < m.c.d.(b^{(N-1)/2^i}-1 , N) < N$

Un número b tal que se cumple $\Psi(N,b)$ se dice ser un *testigo* de la no primicidad de N. Esto es porque si $\Psi(N,b)$ se cumple, entonces N es un número compuesto . En efecto, si la condición *ii)* se cumple, la congruencia del Pequeño Teorema de Fermat dejará de cumplirse, por otro lado, la condición *iii)* implica que N tiene un divisor. En ambos casos se tendrá que N es un número compuesto.

Segundo algoritmo PROBABILISTICO de PRIMOS

Entrada: Un entero impar N mayor que la unidad

Salida : { 'Número compuesto' si N es compuesto
{ 'Número primo' si N es probablemente primo

```
begin
    número = 'Número primo  ;
    for i = 1 to k do
    begin
        res = b_i^(N-1) mod N  ;
        if res ≠ 1 then  ( ℜ(N,b_i) es cierta )
                    número = 'Número compuesto'
        else
        for j=1 to i do
        begin
            x1 = b_i^(2^i m)          ;  x2=x1 mod N  ;
            x3 = m.c.d.( x2 , N ) ;
            if x3 ≠ ( 1 , N ) then ( ℜ(N,b ) es cierta; )
                    número = 'Número compuesto'
        end
    end  ;
    imprime(número)
end
```

LOUIS MONIER demostró en 1980, que el algoritmo de Rabin es más eficiente que el de Solovay y Strassen.

3.3. Aplicaciones

Se presenta una implementación del sistema criptográfico RSA, así como dos algoritmos para generar grandes números con una fuerte certeza de que estos sean primos, además la implementación del algoritmo "p-1" para la factorización de números enteros. Todas las implementaciones fueron hechas en el lenguaje LISP.

En nuestros experimentos hemos trabajado como sigue: Se propone un número N, y se le somete a una serie de divisiones entre los números primos comprendidos en el rango [3,7000],

evitando así que N tenga pequeños factores primos ya que una elección de tal N sería objeto de una fácil factorización por el método "p-1". Si N pasa estas divisiones, se le aplican los dos algoritmos descritos con anterioridad y con ello se tiene una fuerte certeza de que N es efectivamente un número primo. De esta manera hemos encontrado los siguientes números. Ellos fueron utilizados en la implementación del sistema Criptográfico RSA.

D=711040064711110464640070464640640071111046471111046400204l7
P=142222092929292929422222092929000129292942220929294220935297
Q=5400000064710464646464646464646464647111111047046471104071617

Tanto D como Q constan de 59 dígitos y P de 60 dígitos, así N=P·Q constará de 119 dígitos. El número E tal que E·D ≡ 1 mod (P-1)(Q-1) fué calculado por el algoritmo de Euclides, y su valor es E = 209840750836087997958502171785106276723618
732827796535960612771366288143617924S.

Los números P.Q. y D fueron sometidos al algoritmo "p-1" de factorización enteros. Con este algoritmo no se encontro algún factor primo para los números P. Q y D. así que crece aún más la posibilidad de que dichos números generados por los algoritmos de Solovay, Strassen y Rabin sean efectivamente números primos.

La implementación fué diseñada para computadoras personales compatibles. Por la capacidad de memoria se ha estado trabajando con números N de no más de 120 dígitos, pero aún así, su factorización requerirá de no menos de un siglo de tiempo de máquina según la tabla mostrada en la Introducción. Con este tipo de números, es posible manejar mensajes M partidos en lotes de 60 caracteres; puede tomarse cualquier convención para relacionar los caracteres que forman el alfabeto U (ver pag.) y un subconjunto de números naturales; en particular se tomó la convención más simple. 10 ↔ blanco, 11 ↔ A , ... , 36 ↔ Z, 37 ↔ 0, ... , 46↔9. Así el alfabeto utilizado es U = (A , ...,Z, 0 , ... ,9). Como una de tantas pruebas a la que fue sujeto el programa, se utilizaron los números anteriores, para encriptar el siguiente mensaje

YO NEZAHUALCOYOTL LO PREGUNTO
ACASO DE VERAS SE VIVE CON RAIZ EN LA TIERRA
NO PARA SIEMPRE EN LA TIERRA SOLO UN POCO AQUI
AUNQUE SEA DE JADE SE QUIEBRA
AUNQUE SEA DE ORO SE ROMPE
AUNQUE SEA PLUMAJE DE QUETZAL SE DESGARRA
NO PARA SIEMPRE EN LA TIERRA SOLO UN POCO AQUI
PERCIBO LO SECRETO LO OCULTO
OH VOSOTROS SENORES ASI SOMOS
DE CUATRO EN CUATRO NOSOTROS LOS HOMBRES
TODOS HABREMOS DE IRNOS
TODOS HABREMOS DE MORIR EN LA TIERRA
COMO UNA PINTURA NOS IREMOS BORRANDO
COMO UNA FLOR NOS IREMOS SECANDO
AQUI SOBRE LA TIERRA
COMO VESTIDURA DE PLUMAJE DE AVE ZACUAN
DE LA PRECIOSA AVE DE CUELLO DE HULE
NOS IREMOS ACABANDO
MEDITADLO SENORES AGUILAS Y TIGRES
AUNQUE FUERAIS DE JADE
AUNQUE FUERAIS DE ORO
TAMBIEN ALLA IREIS
AL LUGAR DE LOS DESCARNADOS
TENDREMOS QUE DESAPARECER
NADIE HABRA DE QUEDAR
ESTOY EMBRIAGADO LLORO ME AFLIJO PIENSO DIGO
EN MI INTERIOR LO ENCUENTRO
SI YO NUNCA MURIERA
SI NUNCA DESAPARECIERA
ALLA DONDE NO HAY MUERTE
ALLA DONDE ELLA ES CONQUISTADA
QUE ALLA VAYA YO
SI YO NUNCA MURIERA
SI YO NUNCA DESAPARECIERA
A DONDE IREMOS DONDE LA MUERTE NO EXISTE
MAS POR ESTO VIVIRE LLORANDO
QUE TU CORAZON SE ENDERECE
AQUI NADIE VIVIRA PARA SIEMPRE

AUN LOS PRINCIPES A MORIR VINIERON
HAY INCINERAMIENTO DE GENTE
QUE TU CORAZON SE ENDERECE
AQUI NADIE VIVIRA PARA SIEMPRE.

Obtuvimos entonces el código siguiente

509904657392803463302881649213194587323627490498297807082589530874871237409068525964727899689833714422678203145930421760204370434955575474040543994574827523588495506631005180350575195983919504837191353100133625395372599685421687585290555266438867452137312887681449474174998721187473261769616084547642430748473349183099923718371344235476885381903556934111376334585787526284672972201793117035709154559351181150907704875882054385895843693172632301580898625100588598667128495069392385176287287938621952195933651577006317423082311282341867972212662032812932657235565112333478204789234858850137611369617747543374324812168455377473474042743029449155977727525732392932644684815538640993444698784933057513417037924323164792762684870158966314864486589908668172780203019309762499929451321944491929624104599086084373838472946437384525901535956975102697187723590641262172918797352199266942287382769758476469879309396461926161020296154389984950064329942439049588780221471532079878620184163201710545266639513318776961788578183142684790140098888623259680960339597290604033966903866590196689271121570695398834789448296959940068266584186436475088712110640165256373624954782393858283489658591630263347958698553252905855310830980052290672903418198831812494189739613492242557445200271505409597609664834033785666241075549188962315445138611372561355476188322983396439015397782744573041460622113433689470139990475598989861130692428602031742001978472448900732930789411853581886157421061877370419396200387840459530130762105613177878807962951725783912682656331445370130728024064214980941746105901141390676697987031625342648174466428475153263330355036852554631417096104708943142860698159319661351417674723225930736402295984910175032306340027424982697394630621925277900564090548813501293703374194301741985359099826413137396230777176383860987887288840464045465783695

78524670895091494390548977223271306160121159226336O
42501892141757583502107666175295607418884931971208948830781475353600
39671646443503608972955789037398945226243013693390 6
19838846448698056589530749599324274986868591630369498697944304453290
94304635781965825857401200151352285779782338100688
37619822690008713259944451351441614951669854227880611330869404537438
83581705623823517206753082910838925872648031355215 2
29833628620354972288569726679864598795052018123316535348936909150000
21553055481641471942046924164124856722482667495929 1
81437053316059490477512187486029340914329542144650906341647100293030
01312318161967682554898663566142367446002370265287 5
12238967737956087308752720299366574620896328753523993673588704924080
87749137498169783997347341795969939300843866080094 0

El mensaje recuperado fue

!YO NEZAHUALCOYOTL LO PREGUNTO ACASO DE VERAS SE VIVE CON RAIZ EN LA TIERRA NO PARA SIEMPRE EN LA TIERRA SOLO UN POCO AQUI AUNQUE SEA DE JADE SE QUIEBRA AUNQUE SEA DE ORO SE ROMPE AUNQUE SEA PLUMAJE DE QUETZAL SE DESGARRA NO PARA SIEMPRE EN LA TIERRA SOLO UN POCO AQUI PERCIBO LO SECRETO LO OCULTO OH VOSOTROS SENORES ASI SOMOS DE CUATRO EN CUATRO NOSOTROS LOS HOMBRES TODOS HABREMOS DE IRNOS TODOS HABREMOS DE MORIR EN LA TIERRA COMO UNA PINTURA NOS IREMOS BORRANDO COMO UNA FLOR NOS IREMOS SECANDO AQUI SOBRE LA TIERRA COMO VESTIDURA DE PLUMAJE DE AVE ZACUAN DE LA PRECIOSA AVE DE CUELLO DE HULE NOS IREMOS ACABANDO MEDITADLO SENORES AGUILAS Y TIGRES AUNQUE FUERAIS DE JADE AUNQUE FUERAIS DE ORO TAMBIEN ALLA IREIS AL LUGAR DE LOS DESCARNADOS TENDREMOS QUE DESAPARECER NADIE HABRA DE QUEDAR ESTOY EMBRIAGADO LLORO ME AFLIJO PIENSO DIGO EN MI INTERIOR LO ENCUENTRO SI YO NUNCA MURIERA SI NUNCA DESAPARECIERA ALLA DONDE NO HAY MUERTE ALLA DONDE ELLA ES CONQUISTADA QUE ALLA VAYA YO SI YO NUNCA MURIERA SI YO NUNCA DESAPARECIERA A DONDE IREMOS DONDE LA MUERTE NO EXISTE MAS POR ESTO VIVIRE LLORANDO QUE TU CORAZON SE ENDERECE AQUI NADIE VIVIRA PARA SIEMPRE AUN LOS PRINCIPES A MORIR VINIERON HAY INCINERAMIENTO DE GENTE QUE TU CORAZON SE ENDERECE AQUI NADIE VIVIRA PARA SIEMPRE !

Las pruebas fueron hechas en una **TELEVIDEO AT**, con palabra de

16 bits a una velocidad de 10 MHz, el tiempo aproximado del encriptamiento del mensaje fué de 3.20 min. y el de recuperación de 1.35 min. Cabe hacer notar que entre más grande sean los números primos utilizados o más grande sea el mensaje los algoritmos serán más lentos.

4. Bibliografía

1. L. M. Adleman, C. Pomerance y R. S. Rumely.
"On distinguishing prime numbers from composite numbers"
Ann. of Math., Vol. 117 (1983), 13-206.

2. L. M. Adleman, R. L. Rivest y A. Shamir.
"A method for obtaining digital signatures and public-key cryptosystems"
Comm. ACM, Vol. 21 (1978), 120-126.

3. J. Brillhart, D. H. Lehmer, J. L. Selfridge, B. Tuckerman y S. S. Wagstaff Jr.
"Factorizations of $b^n \pm 1$, b=2, 3, 5, 6, ,7, 10, 11, 12 up to high powers"
Contemporary Mathematics, Vol. 22, American Mathematic Society, Providence-Rhode Island, 1983.

4. H. Cohen y H. W. Lenstra Jr.
"Primality testing and Jacobi sums"
Math. Comput., Vol. 42 (1984), 297-330.

5. H. Cohen y A. K. Lenstra.
"Implementation of a new primality test"
Math. Comput., Vol. 48 (1987), 103-121.

6. W. Diffie y M. E. Hellman.
"New directions in Crytography"
IEEE Trans. Inform. Theory, IT-22 (1976), 644-654.

7. J. D. Dixon.
"Factorization and primality test"
Amer. Math. Monthly, Vol. 11 (1984), 333-352.

8. "EL FINANCIERO".
Periódico, 30 de diciembre de 1987.
México D. F., pag. 41.

9. E. Grosswald.
Topics from the theory of numbers
Birkhauser, Boston. 1984.

10. M. E. Hellman.
"An overview of public key cryptography"
IEEE Communications Society Magazine, (1978), 24-32.

11. M. E. Hellman y R. C. Merkle.

"Hiding information and signatures in trapdoor knapsacks"
IEEE Trans. Inform. Theory, IT-24 (1978), 525-530.

12. A. G. Konheim.
Cryptography : a primer.
John Wiley and Sons, New York, 1981.

13. A. Lempel.
"Cryptology in transition"
Computing Surveys, Vol. 11 (1979), 285-303.

14. G. L. Miller.
"Riemann's hypothesis and test for primality"
J. Comput. System Sci., Vol. 13 (1976), 300-317.

15. L. Monier.
"Evaluation and comparison of two efficient probabilistic primality testing algorithms"
Theoret. Comput. Sci., Vol. 12 (1984), 97-108.

16. J. M. Pollard.
"Theorems on factorization and primality testing"
Proc. Camb. Philos. Soc., Vol. 76 (1974), 521-528.

17. C. Pomerance.
"Recent developments in primality testing"
Math. Intelligencer, Vol. 3 (1981), 97-105.

18. C. Pomerance, J. W. Smith y R. Tuler.
"A pipeline architecture for factoring large integers with the quadratic sieve algorithm"
SIAM J. Comput., Vol. 17 (1988), 387-403.

19. M. O. Rabin.
"Probabilistic algorithm for testing primality"
J. Number Theory, Vol. 12 (1980), 128-138.

20. J. E. Shockley.
Introduction to number theory.
Holt, Rinehart and Winston, Inc., New York, 1967.

21. G. J. Simmons.
"Cryptology : The Mathematics of secure communication"
Math. Intelligencer, Vol. 1 (1979), 233-246.

22. R. Solovay y V. Strassen.
"A fast Monte-Carlo test for primality"
SIAM J. Comput. Vol. 6 (1977), 84-85.

23. R. E. Tarjan.
"Complexity of combinatorial algorithms"
SIAM Review. Vol. 20 (1978), 457-491.

24. I. M. Vinogradov.
Fundamentos de la teoría de los números
MIR. Moscú, 1977.

25. H. C. Williams.
"A modification of the RSA public-key encryption procedure"
IEEE Trans. Inform. Theory. IT-26 (1980), 726-729.

26. H. C. Williams.
"A p+1 method of factoring"
Math. Comput., Vol. 39 (1982), 225-234.

27. H. C. Williams.
"Factoring a computer"
Math. Intelligencer, Vol. 6 (1984), 29-36.