EL MÉTODO DE CANTOR- ZASSENHAUS PARA FACTORIZAR POLINOMIOS EN CAMPOS FINITOS*

RUTH NORIEGA SAGÁSTEGUI** AND VILMAR TREVISAN***

Resumen. Este trabajo presenta un estudio sobre los métodos de factorización y cálculo de raíces de polinomios univariantes sobre campos finitos. Presentamos dos algoritmos centrales para factorizarlos, tales como el métodode Berlekamp, y el método del Cantor-Zassenhaus, y sus algoritmos separadores con los cuales se hace eficiente.

Key words. Campos finitos, factorización de polinomios, Berlekamp.

AMS subject classifications. 15A15, 15A09, 15A23

1. Introducción. Juzgamos que este tema es muy importante no solamente por sí mismo sino también por las muchas aplicaciones dentro del Álgebra Computacional, Teoría de Códigos, Teoría Computacional de Números, Criptografía, etc. La Factorización sobre campos finitos es usada como un subalgoritmo para, calcular raíces de polinomios. Estas aplicaciones pueden encontrarse en [13][10].

Los procedmientos usados en la factorización de polinomios son basados en los métodos de Algebra lineal y aritmética polinomial. en el decorrer de los años se buscó encontrar métodos para factorizar polinomios en campos finitos que redujesen cada vez más el tiempo de ejecución del algoritmo. El primero fue introducido por Berlekamp, en 1967 [2]. Su algoritmo reducía el problema para hallar polinomios que formasen una base para el espacio nulo de una matrix nxn sobre $GF(q)$, usando para eso técnicas de álgebra lineal El algoritmo de Berlekamp fue implementado en un número de $O(n^3 + nq)$ operaciones en $GF(q)$ donde n es el grado del polinomio a ser factorizado. Rabin [12] creó su método en 1980 con un tiempo aparentemente inferior al de Berlekamp sin embargo no llegó a probarse matematicamente esta reducción de tiempo. Al hacer la justificación del método, este no tuvo un menor tiempo que el de Berlekamp.

Un Algoritmo con diferencia real fue descrito en 1981 por Cantor y Zassenhaus. Este algoritmo factoriza un polinomio en otros polinomios de diferente grado y después factoriza cada uno de ellos en polinomios irreducibles del mismo grado. El algoritmo puede ser implementado en un tiempo promedio de ejecución de $O(n^2q)$ operaciones en $GF(q)$.

La elección del algoritmo a ser usado para polinomios en campos finitos depende del tamaño del campo en que está, pues algunos métodos son eficientes para la factorización en campos pequeños, más son ineficientes cuando son aplicados a campos finitos grandes.

La factorización de polinomios es un problema ancestral que ha preocupado a los matemaáticos y científicos, por el simple hecho de encontrarse casi siempre con el problema de hallar raíces de polinomio $f(x)$. Sin embargo hasta ahora este problema aun no esta resuelto, salvo en situaciones y bajo condiciones particulares se puede usar un algoritmo construído con ciertas técnicas del álgebra y de las teorías matemáticas. Es en estas circunstancias que es conveniente saber hallar los factores de los poli-

*Este trabajo es financiado con los recursos ordinarios asignados a la UNT.

**Departamento de Matemáticas, Universidad Nacional de Trujillo (bbruthns@hotmail.con).

***Instituto de Matemáticas, Universidade Federal do Rio Grande do Sul (trevisan@pgmat.ufrgs.br).

nomios por lo que nos hemos interesado en estudiar algunos métodos para factorizar polinomios sobre campos finitos ($GF(q)$) donde $q = p^n$, n es el grado del polinomio $f(x)$. Puesto que cualquier polinomio en el anillo de polinomios sobre los enteros se puede proyectar de alguna manera sobre un anillo de polinomios sobre campos finitos, que no es mas que extensiones sobre campos primos finitos, que son de la forma módulo un primo p, en tal sentido nos planteamos el problema

Es posible determinar un algoritmo eficiente para hallar factores de polinomios sobre campos finitos ?

Afirmamos que es posible hallar un Algoritmo eficiente respecto a otros algoritmos existentes que fue descrito por Cantor y Zassenhaus.

Este algoritmo factoriza un polinomio en otros polinomios de diferente grado y después factoriza cada uno de ellos en polinomios irreducibles del mismo grado. El algoritmo puede ser implementado en un tiempo promedio de ejecución de $O(n^2q)$ operaciones en $GF(q)$. En este trabajo presentamos el algoritmo de Cantor-Zassenahus y sus algoritmos separadores.

2. Factorización sobre Campos Finitos. En esta sección estudiaremos los métodos para factorizar polinomios sobre campos finitos, para ello existen muchos métodos de los cuales abordaremos el método de Berlekamp [3].

Su algoritmo reducirá el problema para hallar polinomios que formasen una base para el espacio nulo de una matrix nxn sobre $GF(q)$, usando para eso técnicas de álgebra lineal. El algoritmo de Berlekamp fue implementado en un número de $O(n^3 + nq)$ ooperaciones en $GF(q)$ donde n es el grado del polinomio a ser factorizado.

Dado $U(x) \in F[x]$ donde F es um campo finito, es posible obtener únicos polinomios distintos $u_1(x), u_2(x), ..., u_n(x)$ irreducibles con coeficientes en F, de tal modo que

$$U(x) = u_1^{s_1}(x).u_2^{s_2}....u_s^{s_s}(x)$$

donde $s_1, s_2, .., s_s$ son enteros mayores que cero. La garantia de la existencia y unicidad de los u_i es dado por el hecho de $F[x]$ ser un Dominio Euclidiano, por tanto de Factorización única.

En este sección trataremos justamente de como obtener los u_i . Y haremos un breve estudio para obtener polinomios libre de cuadrados. Para lo cual precisamos conocer la:

2.1. Descomposición libre de cuadrados. Sea un polinomio p(x) de $R[x]$ donde R es un anillo de característica cero.[1] Es posible que $p(x)$ tenga factores múltiples, es decir, que exista un polinomio q tal que q^2 divida $p(x)$ (quizás una potencia más alta de q que divida a $p(x)$), pero en este caso todavía es verdad que q^2 divide a $p(x)$.

Obviamente podemos encontrar todos los factores múltiples haciendo una factorización completa de $p(x)$, pero existe una forma más simple que ahora describiremos.

Es suficiente considerar $p(x)$ mónico. Supongamos que $p(x)$ sea factorizado en un producto de factores lineales:

$$p(x) = \prod_{i=1}^{n}(x - a_i)^{n_i},$$

[1] hacemos esta hipótesis para excluir el caso de x^{p+1} sobre un campo com p elementos: este polinomio es irreducible, pero su derivada es cero

donde los a_i pueden ser cantidades algebraicas sobre R (hacemos esta factorización sólo para la prueba). La derivada de $p(x)$ es:

$$p'(x) = \sum_{n}^{i=1}(n_i(x-a_i)^{n_i-1}\prod_{\substack{j=1\\ i\neq j}}^{n}(x-a_j)^{n_j})$$

Es obvio que para cada i, $(x-a_i)^{n_i-1}$ divide $p(x)$ y $p'(x)$. Sin embargo cada polinomio que divide a $p(x)$ es un producto de los $(x-a_i)$ para una potencia menor o igual que n_i. Por lo tanto el mcd(máximo común divisor) esta determinado: este es el producto de los $(x-a_i)$,para una potencia n_i-1 o n_i. Pero esta potencia no puede ser n_i, pues $(x-a_i)^{n_i}$ divide todos los términos de $p'(x)$ excepto uno, y por lo tanto no puede dividir $p'(x)$.

Así que hemos verificado el siguiente proceso:

$$d(x) := mcd(p(x), p'(x)) = \prod_{i=1}^{n}(x-a_i)^{n_i-1},$$

entonces

$$q(x) := \frac{p(x)}{mcd(p(x), p'(x))} = \prod_{i=1}^{n}(x-a_i)$$

por tanto

$$mcd(q(x), mcd(p(x), p'(x))) = \prod_{\substack{i=1\\ n_i>1}}^{n}(x-a_i),$$

por lo que deducimos

$$h(x) := \frac{q(x)}{mcd(q(x), mcd(p(x), p'(x)))} = \prod_{i=1 n_i=1}^{n}(x-a_i).$$

El lado izquierdo de esta ecuación es el producto de todos los factores no-múltiples de $p(x)$ y hemos mostrado como calcularlo usando sólo operaciones de derivación, división exacta y mcd. Estas operaciones no nos hicieron salir de $R[x]$ lo que implica que este producto puede ser calculado en $R[x]$ y por un método eficiente más rápido.

En resumen, hemos calculado como un resultado intermediario el $mcd(p(x), p'(x))$, que tiene los mismos factores múltiples de $p(x)$, pero con la potencia reducida en 1, el mismo procedimiento que se aplicó a $p(x)$ es ahora aplicado a $mcd(p(x), p'(x))$, entonces estamos calculando todos los factores de $mcd(p(x), p'(x))$ de multiplicidad uno, es decir los factores de $p(x)$ de multiplicidad 2; y similaremente para los factores de multiplicidad 3,...etc.

Esto es, por medio de cálculos simples en $R[x]$ podemos factorizar de la forma $\prod p(x)_i^i$, donde $p(x)_i$ es el producto de todos los factores de p(x) de multiplicidad i. En esta descomposición de p(x), cada $p(x)_i$ no tiene factores múltiplos, y los $p(x)_i$ son relativamente primos. Esta descomposición es conocida con el nombre de **Libre de cuadrados**.

Ejemplo 2.1. *Sea el polinomio*

$$p(x) = x^{12}+12x^{11}+56x^{10}+123x^9+113x^8+x^7-59x^6-95x^5-122x^4+71x^3+103x^2-72x+12.$$

Calculamos el $mcd(p(x), p'(x))$*.,*

$$d_1(x) = x^5 + 6xt + 10x^3 - 7x + 2$$

entonces,

$$q_1(x) = x^7 + 6x^6 + 10x^5 + 3x^4 + 2x^3 - x^2 - 15x + 6$$

así tenemos que,

$$h_1(x) = x^4 + 2x^3 - x^2 + 3x - 3.$$

continuamos con el proceso anterior aplicado a $d_1(x)$

$$d_2(x) = mcd(d_1(x), \frac{d}{dx}d_1(x)) = x^2 + 2x - 1,$$

y

$$q_2(x) = \frac{d_1(x)}{d_2(x)} = x^3 + 4x^2 + 3x - 2,$$

de donde obtenemos

$$h_2(x) = x + 2$$

Aqui calculamos

$$d_3(x) = mcd(d_2(x), \frac{d}{dx}(x)) = 1,$$

así que

$$h_3(x) = d_2(x) = x^2 + 2x - 1.$$

Concluímos por lo tanto que la descomposición libre de cuadrados de p(x) es,

$$p(x) = (x^4 + 2x^3 - x^2 + 3x - 3)(x + 2)^2(x^2 + 2x - 1)^3.$$

2.2. El Método de Berlekamp. El algoritmo de Berlekamp es un algoritmo que aun se sigue usando en la factorización de polinomios sobre campos finitos, a pesar de haber sido el primer algoritmo desarrollado, y es fácil de entender.

El Método de Berlekamp usa el siguiente resultado de fundamental importancia

THEOREM 2.1. *Sea* $f(x) \in GF(q)[x]$ *mónico y* $h(x) \in GF(q)[x]$ *de tal modo* $(h(x))^p \equiv h(x) \, mod f(x)$*. Entonces*

$$f(x) = \prod_{c \in GF(q)} mcd\,(f(x), h(x) - c) \tag{2.1}$$

Demostración: Cada máximo común divisor del lado derecho de (2.1) divide a $f(x)$.

Como los polinomios $h(x) - c$, $c \in GF(q)$ son primos relativos, entonces son los máximos común divisores de éstos con $f(x)$ también son primos relativos, por tanto el producto de todos estos máximos comunes divisores dividen a $f(x)$.

Como se cumple $x^q - x = x(x-1)(x-2)(x-3)\dots(x-(q-1))modq$ se tiene la identidad

$$h(x)^q - h(x) = \prod_{c \in GF(q)} (h(x) - c)$$

Como por hipótesis $f(x)$ divide a $h(x)^q - h(x)$, entonces $f(x)$ divide el producto del lado derecho, entonces $f(x)$ divide el lado derecho del (2.1), por tanto son iguales. □

Ahora presentamos el algoritmo de Berlekamp.

Sea $U(x) = u_n x^n + ... + u_1 x + u_0$, un polinomio mónico con coeficientes en el conjunto $\{0, 1, 2, ..., p-1\}$, usando aritmética módulo **p**, queremos expresar $U(x)$ en un producto de r polinomios irreducibles.

Asumimos que U(x) es libre de cuadrados, esto es, $U(x)$ no tiene factores múltiples.

Por tanto como estamos en un Dominio de Factorización Única, suponemos que

$$U(x) = p_1(x).p_2(x)....p_r(x)$$

es el producto de factores relativamente primos y diferentes entre sí.

Para descubrir los $p_j(x)$ conociendo sólo $U(x)$, la idéa de Berlekamp ver[2] es hacer uso del Teorema del Resto, que es válido tanto para polinomios como para enteros.

Si $(s_1, s_2, ..., s_r)$ es una r-upla de enteros módulo **p**, el Teorema del Resto implica que existe un único $V(x)$ tal que,

$$= \begin{cases} V(x) \equiv s_1 \text{ mód } p_1(x) \\ V(x) \equiv s_2 \text{ mód } p_2(x) \\ \quad \vdots \\ V(x) \equiv s_r \text{ mód } p_r(x) \end{cases} \tag{2.2}$$

donde,

$\delta(V) < \delta(p_1) + \delta(p_2) + ... + \delta(p_r) = \delta(U)$.

El conocimiento del polinomio $V(x)$ nos proporciona una forma de obtener los factores de $U(x)$.

Por ejemplo, si $r = 2$ y $s_1 \neq s_2$, $mcd(U-(x), V(x) - s_1)$ es divisible por $p_1(x)$ y no por $p_2(x)$.

Observamos que es posible obtener información sobre los factores de $U(x)$, partiendo de soluciones apropiadas para el sistema anterior(2.2) asi:

En primer lugar se observa que $V(x)$ satisface la condición :

$$\begin{cases} V(x)^p \equiv s_j^p = s_j \equiv V(x) \quad \text{mód } p_j(x) \\ V(x)^p \equiv V(x) \quad (\text{mód } U(x)) \end{cases} \tag{2.3}$$

donde, $1 \leq j \leq r$ e $\delta(V) \leq \delta(U)$.

En segundo lugar tenemos la identidad polinomial básica:

$$x^p - x \equiv (x-0)(x-1)...(x-(p-1)) \quad \text{mód } p$$

luego se tiene:

$$V(x)^p - V(x) = V(x)(V(x)-1)...(V(x)-(p-1)) \tag{2.4}$$

Si V(x) satisface (2.3), se sigue que $U(x)$ divide el lado izquierdo de (2.4),(por definición de módulo); así cada factor irreducible de $U(x)$ debe dividir uno de los p-factores primos del lado derecho de (2.4).

En otras palabras todas las soluciones de (2.3) deben tener la forma de (2.2), es decir para algunos $s_1, s_2, ..., s_r$ existen exactamente p^r soluciones de (2.3).

Las soluciones $V(x)$ del sistema (2.3) son esenciales para la factorización de $U(x)$. Hasta puede parecer mas difícil hallar todas las soluciones para (2.3), que factorizar $U(x)$; pero ésto no es verdad, pues el conjunto de soluciones para (2.3) es cerrado para la adición. Y así hemos probado (usando el Método de Knuth p.422)) el siguiente resultado:

THEOREM 2.2 (**El Teorema de Berlekamp**). *[1967] Las soluciones V(x) de*

$$= \begin{cases} V(x) \equiv s_1 \quad \text{mód } p_1(x) \\ V(x) \equiv s_2 \quad \text{mód } p_2(x) \\ \qquad \vdots \\ V(x) \equiv s_r \quad \text{mód } p_r(x) \end{cases}$$

son precisamente las soluciones de

$$\begin{cases} V(x)^p \equiv s_j^p = s_j \equiv V(x) \quad \text{mód } p_j(x) \\ V(x)^p \equiv V(x) \quad \text{mód } U(x) \end{cases}$$

donde, $1 \leq j \leq r$ *e* $\delta(V) \leq \delta(U)$.

Ya hemos dicho que las soluciones de (2.2) proveen información acerca de la factorización de $U(x)$, pero todavía tenemos el problema de encontrarlas, la idéa básica de Berlekamp es notar que la ecuación

$$V(x)^p \equiv V(x) \quad \text{mód } U(x) \tag{2.5}$$

es una ecuación **lineal** para los coeficientes de $V(x)$.

En efecto:

Supongamos que, n es el grado de $U(x)$, y el polinomio $V(x)$ tiene la forma

$$V(x) = v_{n-1}x^{n-1} + \cdots + v_1 x + v_0 = (v_0, v_1, \ldots, v_{n-1}) \cdot \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{n-1} \end{pmatrix}$$

Sabemos que se cumple

$$(V(x))^p = V(x^p) = v_{n-1}x^{p(n-1)} + \cdots + v_1 x^p + v_0 = (v_0, v_1, \ldots, v_{n-1}) \cdot \begin{pmatrix} 1 \\ x^p \\ \vdots \\ x^{p(n-1)} \end{pmatrix}$$

Pero se tiene

$$\begin{aligned} x^{pk} &= q_{k,n-1}x^{n-1} + q_{k-1,n-2}x^{n-2} + \ldots + q_{k,1}x + q_{k,0} \quad \text{mód } U(x) \\ &= (v_{q,0}, q_{k,1}, \ldots, q_{q,n-1}) \cdot \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{n-1} \end{pmatrix} \end{aligned}$$

Denotamos por B la matriz de coeficientes de orden $n \times n$:

$$B = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ q_{n-1,0} & q_{n-1,1} & \cdots & q_{n-1,n-1} \end{pmatrix}$$

Entonces se tiene que la ecuación (2.5) puede escribirse de la siguiente manera

$$(v_0, v_1, \ldots, v_{n-1})\, B \begin{pmatrix} 1 \\ x^p \\ \vdots \\ x^{p(n-1)} \end{pmatrix} = (v_0, v_1, \ldots, v_{n-1}) \cdot \begin{pmatrix} 1 \\ x^p \\ \vdots \\ x^{p(n-1)} \end{pmatrix}$$

de donde, podemos calcular los coeficientes del polinomio $V(x)$ al resolver el sistema lineal

$$(v_0, v_1, ...v_{n-1})B = (v_0, v_1, ...v_{n-1})$$

Si $v = (v_0, v_1, ...v_{n-1})$, resolver el sistema anterior es equivalente a resolver el sistema

$$v(B - I) = 0$$

donde $I = (\delta_{ij})_{n\times n}$, es la matriz identidad.

Lo que significa que los coeficientes de $V(x)$ estarán en el espacio nulo de la matriz $B - I$, para resolver este sistema, usamos cualquier algoritmo del algebra lineal.

A continuación presentamos el algoritmo de Berlekamp

2.2.1. El Algoritmo de Berlekamp. B1.- Constatar que $U(x)$ sea libre de cuadrados.
B2.- Formar la Matriz B , con $x^{pk} \equiv U(x) mod p$
B3.- Triangularizar la matriz $B - I$, hallando su rango r. Luego k=n-r= número de vectores linealmente independientes del nucleo de $B - I$: $v^{[1]}, v^{[2]}, ..., v^{[k]}$, tal que

$$v^{[j]}(B - I) = (0, 0, ..., 0), \qquad 1 \leq j \leq k$$

El primer vector siempre es (1,0,...0) y representa la solución trivial $v^{[1]}(x) = 1$ para (2.3); k es el número de factores irreducibles de $U(x)$.

Por tanto si $r = 1$, $U(X)$ es irreducíble y el procedimento termina.

B4.- Calcular

$$mcd(U(x), v^{[2]}(x) - s) \qquad para \qquad 0 \leq s < p$$

donde $v^{[2]}(x)$, es el polinomio representado por $v^{[2]}$, el resultado será una factorización no trivial de $U(x)$, pues $v^{[2]}(x) - s \neq 0$ y tienen el grado menor que el grado de $U(x)$. Siempre y cuando $V(x)$ satisfaga (2.3),

Si cuando usamos $v^{[2]}(x)$ no separa $U(x)$ en k-factores, los factores seguientes se pueden obtener calculando

$$mcd(v^{[j]}(x) - s, U(x)) \qquad \text{para} 0 \leq s < p$$

y todos los factores de $U(x)$ serán encontrados, para $j = 3, 4, ...$ hasta que los k factores sean obtenidos.

Si escogemos $s_i \neq s_j$ en (2.3), se obtiene la solución $V(x)$ para (2.4) que diferencia $p_i(x)$ de $p_j(x)$; algún $v^{[l]}(x) - s$ será divisible por $p_i(x)$ y no por $p_j(x)$, asíéste procedimento hallará todos los factores de $U(x)$.

2.2.2. Aplicación del Algoritmo de Berlekamp. Veremos la aplicación del algoritmo anterior en el siguiente,

Ejemplo 2.2. *Sea* $f(x) = x^5 - 9x^4 + 3x^3 + x^2 - 2x + 8 \in GF(31)$.

Para factorizar este polinomio iniciamos verificando si existen factores repetidos, para eso debemos calcular el $mcd(f(x), f'(x))$, *o sea* $mcd(x^5 - 9x^4 + 3x^3 + x^2 - 2x + 8, 5x^4 - 36x^3 + 9x^2 + 2x - 2) = 1$*(usando el algoritmo de Euclides). Por tanto* $f(x)$ *no tiene factores repetidos. Calculamos* $x^{iq} \mod f(x)$ *para* $q = 31$ *y* $0 \leq i \leq 4$.

$$\begin{aligned}
x^{0\times 31} &= x^0 \equiv 1 \\
x^{1\times 31} &= x^{31} \equiv 11 - 11x - 11x^2 + 12x^3 + 11x^4 \\
x^{2\times 31} &= x^{62} \equiv -5 + 7x - 14x^2 - 10x^3 - 4x^4 \\
x^{3\times 31} &= x^{93} \equiv -9 - 12x + 7x^2 + 13x^3 - 11x^4 \\
x^{4\times 31} &= x^{124} \equiv 10 + x + 3x^3 + 11x^4.
\end{aligned}$$

Obteniendo así la matriz 5×5

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
11 & -11 & -11 & 12 & 11 \\
-5 & 7 & -14 & -10 & -4 \\
-9 & 12 & 7 & 13 & -11 \\
10 & 1 & 0 & 3 & 11
\end{pmatrix}$$

y $B - I$ *esta dada por*

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 \\
11 & -12 & -11 & 12 & 11 \\
-5 & 7 & -15 & -10 & -4 \\
-9 & 12 & 7 & 12 & -11 \\
10 & 1 & 0 & 3 & 10
\end{pmatrix}$$

Escalonamos la matriz $B - I$ *a fin de obtener el número de filas no nulas,* $n = 4$. *Sabiendo el valor de* $r = 4$, *podemos encontrar el número de factores irreducibles y mónicos, distintos de* $f \in GF(31)$, $k = n - r = 1$. *Como* $k = 1$, *eso indica que el espacio nulo de* $B - I$ *tiene dimensión 1, entonces el polinomio es irreducible sobre* $GF(31)$.

Ejemplo

Factorizar:

$$U(x) = x^8 + x^6 + 10x^4 + 10x^3 + 8x^2 + 2x + 8 \mod 13.$$

Usando el algoritmo de Euclides tenemos:

$$mcd(U(x), U'(x)) = 1$$

Por tanto $U(x)$ es *libre de cuadrados*, no tiene factores repetidos, y satisface la primera parte(B1) del Algoritmo de Berlekamp.

En B2 formamos la matriz B de orden 8×8. Podemos obtener las filas de B dando solución para:

$$\begin{aligned} x^0 &\quad \text{mód } U(x) \\ x^{13} &\quad \text{mód } U(x) \\ x^{26} &\quad \text{mód } U(x) \\ &\quad \vdots \\ x^{91} &\quad \text{mód } U(x) \end{aligned}$$

y así hallamos la matriz B.

En el paso B3. obtenemos el rango $r = 5$ de la matriz $A = B - I$, y luego $k = 8 - 5$ es el número de vectores linealmente independientes:$v^{[1]}, ..., v^{[K]}$ tal que,

$$v^{[1]}A = v^{[2]}A = ... = v^{[k]}A = (0, ..., 0),$$

entonces $U(x)$ debe tener exactamente 3 factores irreducibles:

$$\begin{aligned} v^{[1]} &= (1, 0, ..., 0) \\ v^{[2]} &= (0, 5, 5, 0, 9, 5, 1, 0) \\ v^{[3]} &= (0, 9, 11, 9, 10, 12, 0, 1) \end{aligned}$$

En el paso B4 , al calcular $mcd(U(x), v^{[2]}(x) - s)$, para $0 \leq s < 13$, obtenemos para:

$$\begin{aligned} s = 2 &\quad \to x^3 + 8x^2 + 4x + 12 \\ s = 0 &\quad \to x^5 + 5x^4 + 9x^3 + 5x + 5 \end{aligned}$$

Por tanto con $v^{[2]}$ obtengo sólo dos de los tres factores.(para los otros valores de s, tengo solamente uno).

Entonces ahora usamos el tercer vector y hacemos el mismo procedimiento. De

$$mcd(v^{[3]} - s, x^5 + 5x^4 + 9x^3 + 5x + 5)$$

obtenemos,

$$\begin{aligned} s = 6 &\quad \to x^4 + 2x^3 + 3x^2 + 4x + 6 \\ s = 8 &\quad \to x + 3 \end{aligned}$$

Por lo tanto

$$U(x) = \prod mcd(V(X) - s, U(x)) = (x^4 + 2x^3 + 3x^2 + 4x + 6)(x^3 + 8x^2 + 4x + 12)(x + 3)$$

3. El Método de Cantor-Zassenhaus. En esta sección presentamos el método de Cantor-Zassenhaus y sus algoritmos computacionales que permiten factorizar polinomios sobre campos finitos. Cantor y Zassenhaaus introdujeron un nuevo algoritmo probabilístico, capaz de hallar soluciones de ecuaciones polinomiales sobre cualquier campo finito.

Este método factoriza un polinomio en otros polinomios de diferente grado y después factoriza cada uno de ellos en polinomios irreducibles del mismo grado. El algoritmo puede ser implementado en un tiempo promedio de ejecución de $O(n^2q)$ operaciones en $GF(q)$.

Suponga que $F = GF(p)$ es un campo finito de característica p, con $q = p^d$ elementos. Un punto fundamental en la computación es hallar los factores irreducibles de un polinomio moníco.

$$\begin{aligned} f(x) &= \sum_{i=0}^{n} a_i x^i, \\ &= x^n + a_{n-1}x^{n-1} + ... + ax + a_0 \in F[x] = GF(p)[x]. \end{aligned}$$

con $a_n = 1$.

Para encontrar una factorización de un polinomio $f(x)$ sobre un campo finito usando el método de Cantor-Zassenhaus debemos seguir los siguientes pasos:

3.1. Algoritmo de Cantor Zasshenhaus. ALGORITMO 3.1 (ACZ).

ACZ1.-

Verificar si f no tiene factores repetidos, es monico y $f(0) \neq 0$.

ACZ2.-

factorizar f en un producto: $f(x) = \prod_{i=1}^{n} h_i(x)$, donde $h_i(x)$ contiene solo factores irreducibles de grado i o multiplo de i.

ACZ3.-

factorizar cada $h_i(x)$ en factores irreducibles.

La factorización del Paso [ACZ2] debe ser realizada usando el algoritmo de factorización de grado diferente [NDDF].

La factorización del paso [ACZ3] se realiza usando el algoritmo de grado Uniforme[FGU].

3.2. El Algoritmo de Factorización de grado diferente. En este algoritmo procuramos descomponer a $f(x)$ en $h_i(x)$ o sea $f(x) = h_1(x)...h_i(x)$, donde cada $h_i(x)$ representa el producto de m factores irreducibles pertenecientes a $GF(q)$ de grado i.

Como sabemos $h_i(x)$ es el producto de todos los polinomios mónicos irreducibles de grado i sobre $GF(q)$.

Por el teorema: çada campo finito y cada $k \in \mathbb{N}$ el producto de todos los polinomios mónicos irreducibles sobre el $GF(q)$ cuyo grado divide a i, es igual a $x^{q^i} - x$,", entonces debemos tener $h_i(x) = x^{q^i} - x$. Si calculamos el $mcd(f(x), x^{q^i} - x)$ obtendremos solamente los factores de f, que son precisamente los $h_i(x)$.

Para encontrarlos comenzamos esta factorización escogiendo $i = 1$ y calculando la ecuación x^{q^i} mód $f(x)$; el polinomio resultante será denominado $r_1(x)$.

Luego calculamos $h_1(x)$ mediante el $mcd(f(x), r_1(x) - x)$.

Si $h_1(x) = f(x)$, el algoritmo de factorización de grado diferente termina, pues ya descubrimos todo el conjunto de $h_i(x)$ que factoriza a $f(x)$.

En caso contrario tomamos $i = i+1$ y encontramos el nuevo $f(x)$ que es obtenido dividiendo $f(x)$ por $h_1(x)$.

Verificamos si este $f(x)$ tiene grado menor que $2i$; en caso de que lo tenga el algoritmo acaba, y $f(x)$ pasa a ser parte del conjunto de los $h_i(x)$. Pero si $2i < gradof(x)$, calculamos un nuevo $r(x)$.

Como vimos anteriormente para encontrar $r(x)$ debemos calcular $x^{q^i} modf(x)$, pero dependiendo del campo en el cual se esta trabajando se vuelve muy arduo, por eso podemos usar otra alternativa para encontrar $r(x)$.

Para esto suponga que

$$r_{i-1}(x) = b_{n-1}x^{n-1} + ...b_1x + b_0$$

un polinomio de $GF(q)[x]$, y por tanto se cumple

$$r_i(x) = (r_{i-1}(x))^q = b_{n-1}x^{q(n-1)} + ... + b_1x^q + b_0.$$

Si precalculamos los valores

$$B_i(x) = x^{iq} modf(x), \quad \text{para} \quad i = 0, ..., n-1$$

y almacenamos los coeficiente B_i como la $i$ésima fila de la matriz B de orden $n \times n$, y tendremos que $r_i(x) = r_{i-1}(x)B$, o sea, debemos multiplicar $r(x)$ por la matriz B y el vector resultante será el nuevo $r(x)$.

Y así repetimos el algoritmo hasta obtener el grado de $f(x) < 2i$ o en otro caso $h_i(x) = f(x)$.

El algoritmo de Factorización de grado diferente $NDDF$ construye la matríz B como el algoritmo de Berlekamp y puede ser descrito de la siguiente manera.

Algoritmo 3.2 (NDDF).

NDDF1.-

Hacer $i \leftarrow 1$;
$r(x) \leftarrow x^q \mod f(x)$

NDDF2.-

Calcular $h_i \leftarrow mcd(f(x), r(x) - x)$,

NDDF3.-

Si $h_i(x) = f(x)$, *entonces termina;*
de otra manera $f(x) \leftarrow \frac{f(x)}{h_i(x)}$, *y* $i \leftarrow i + 1$

NDDF4.-

Si $2i > grad\,(f(x))$, *entonces* $h_{grad(f)} \leftarrow f(x)$ *y termina;*
de otra manera, $r(x) \leftarrow r(x).B$ *luego ir al paso 2.*

3.3. El Algoritmo de Factorización de Grado Uniforme. Después de haber encontrado el conjunto de los $h_i(x)$ que factorizan a $f(x)$, la tarea siguiente es encontrar los n factores irreducibles que componen cada $h_i(x)$ para eso usamos el algoritmo de grado uniforme, que tiene el siguiente procedimiento.

Se escoge de manera aleatoria un polinomio $t(x)$ en $GF(q)[x]$ de grado menor que $2i - 1$ y calculamos el nuevo $t(x)$ a través del cálculo de

$$(t(x))^{\frac{q^i-1}{2}} mod h_i(x).$$

La probabilidad de que el polinomio $t(x)$ sea un separador es igual a $1/2$, podemos ver la prueba en [6].

Ahora calculamos $g(x)$ hallando el $mcd(h_i(x), t(x)-1)$. Al obtener la factorización de los $h_i(x)$, encontramos la factorización de $f(x)$.

El algoritmo de Factorización de Grado Uniforme sigue los siguientes pasos:

ALGORITMO 3.3 (FGU).

FGU1.- Escoger aleatoriamente $t(x)$ en $GF(q)[x]$ de grado menor que $2i-1$.

FGU2.- Calculamos el nuevo $t(x)$ a través del calculo de $(t(x))^{\frac{q^i-1}{2}}$ mód $h_i(x)$.

FGU3.- Calculamos $g(x)$ a través del cálculo $mdc(h_i(x), t(x)-1)$.

3.3.1. Aplicación del Algoritmo de Cantor Zassenhaus . Sea el polinomio sobre $GF(11)$

$$f(x) = 4x^7 + 5x^6 + x^5 + 4x^4 + 3x^3 + 4x^2 - 4$$

Para factorizar este polinomio debemos verificar primeramente que sea libre de cuadrados, es decir que no tenga factores repetidos, para eso calculamos el

$$\begin{aligned} mdc(f(x), f'(x)) &= mdc(4x^7 + 5x^6 + x^5 + 4x^4 + 3x^3 + 4x^2 - 4, \\ &\quad 28x^6 + 30x^5 + 16x^4 + 9x^2 + 8x) \\ &= 1 \end{aligned}$$

Por tanto $f(x)$no tiene factores repetidos.

Formamos la Matriz B, calculando x^{iq} mód $f(x)$ donde $0 \leq \mathrm{i} \leq n-1$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & -3 & -5 & 2 & 3 & 3 & -1 \\ -1 & -3 & -2 & 5 & 2 & -1 & -1 \\ 1 & -4 & 0 & 3 & 2 & -4 & -5 \\ -1 & -5 & 0 & -4 & 5 & 7 & 2 \\ -5 & 2 & 4 & -2 & -5 & -3 & 4 \\ 3 & 3 & 0 & 4 & 4 & 3 & 1 \end{bmatrix}$$

Tomamos $i = 1$ y calculamos $r_1(x) = x^{11} mod f(x)$. Entonces $r_1(x) = -5x^6 - 3x^5 - 5x^4 + 2x^3 + 3x^2 + 3x - 1$.

Obtenemos $h_1(x)$ calculando el

$$\begin{aligned} h_1(x) &= mcd(f_1(x), r_1(x) - x) \\ &= mcd(4x^7 + 5x^6 + x^5 + 4x^4 + 3x^3 + 4x^2 - 4, \\ &\quad -5x^6 - 3x^5 - 5x^4 + 2x^3 + 3x^2 + 3x - 1 - x) \\ &= x^3 + x^2 + 3x - 1. \end{aligned}$$

Como $h_1(x)$ no es igual a $f(x)$ calculamos un nuevo $f(x)$, por

$$\begin{aligned} f(x) &= \frac{f(x)}{h_1(x)} \\ &= \frac{4x^7 + 5x^6 + x^5 + 4x^4 + 3x^3 + 4x^2 - 4}{x^3 + x^2 + 3x + 5} \\ &= 4x^4 + x^3 - x^2 + 4x - 3 \end{aligned}$$

El próximo $i = 2$, verificamos si $2i$ es mayor que el grado de $f(x)$, como no lo es,pues 4 no es mayor que 4, continuamos la factorización encontrando un nuevo $r(x)$, o sea $r_2(x) = r_1(x).B = x$

El $h_2(x)$ obtenido a través del cálculo del $mcd(4x^4 + x^3 - x^2 + 4x - 3, x - x) = 4x^4 + x^3 - x^2 + 4x - 3$.

Por tanto $h_2(x) = 4x^4 + x^3 - x^2 + 4x - 3$. como es igual a $f(x)$ el algoritmo de factorización de grado diferente termina.

Nuestra tarea es ahora factorizar los $h_i(x)$ encontrados.

Sabemos de antemano que $f(x)$ tiene 3 factores irreducibles de grado 1 y dos factores irreducibles de grado 2.

Comenzaremos encontrando la factorización de $h_1(x)$. Para eso escogemos aleatoriamente un polinomio $t(x) = x + 1 \in GF(11)$ y calculamos el nuevo $(t(x))^{\frac{11^1-1}{2}} = (x + 1)^5$.

Haciendo $(x + 1)^5$ mód $h_1(x)$ obtenemos $x^2 + 9x + 8$.

$$g(x) = mdc(x^3 + x^2 + 3x + 5, x^2 + 9x + 8 - 1) = x + 3$$

Como sabemos que la factorización de $f(x)$ tiene tres factores de grado 1, vamos ha repetir el proceso, escogiendo aleatoriamente otro polinomio $t(x)$, ahora $t(x) = x + 3 \in GF(11)$.

Calculamos el nuevo $t(x)$a través del cálculo de

$$(t(x))^{\frac{11^1-1}{2}} = (x + 3)^5.$$

Haciendo $(x + 3)^5 mod h_1(x)$ obtenemos $7x^2 + 6x + 10$.

$$g(x) = mcd(x^3 + x^2 + 3x + 5, 7x^2 + 6x + 10) = x + 5$$

Como ya encontramos dos factores irreducibles de grado 1. Para encontrar el tercero basta dividir $h(x)$ por el producto de los dos factores ya encontrados $mod 11$. Encontraremos el otro factor que es $x + 4$.

El paso siguiente es ahora factorizar $h_2(x)$, para eso escogemos aleatorimente el polinomio $t(x) = x + 1 \in GF(11)$y calculamos el nuevo $t(x)$ a través de $(t(x))^{\frac{11^2-1}{2}} = (x + 1)^{60}$.

Haciendo $(x + 1)^{60}$ mód $h_2(x)$ obtenemos $3x^3 - 3x^2 - x + 1$.

$$g(x) = mcd(4x^4 + x^3 - x^2 + 4x - 3, 3x^3 - 3x^2 - x) = x^2 - x - 4.$$

Como existen dos factores irreducibles de grado dos y ya encontramos uno, dividimos $h_2(x)$ por $g(x)$ y así obtenemos el otro factor que es $4x^2 + 5x + 20$.

Por tanto la factorización de $f(x)$es

$$f(x) = (x + 1)(x + 3)(4x^2 + 5x + 20)(x^2 - x - 4).$$

3.3.2. Costo del Algortimo. El mayor costo del algoritmo esta en el paso 2, en general para elevar $r_i(x)$ a una potencia grande se usa el método binario [2] (el método binario para p^n con p primo y aritmética modular es óptimo).

El número de multiplicaciones módulo $f(x)$ es de orden : $(lnq)^2$.

A continuación presentamos un ejemplo basado en un programa computacional hecho en MAPLE, es el mismo ejemplo dado para Berlekamp.

EJEMPLO 3.1. *Sea*

$$f(x) = x^8 + x^6 + 10x^4 + 10x^3 + 8x^2 + 2x + 8 \pmod{13}$$

Aplicando el algoritmo NDDF tenemos

```
> nddf(pol,13);
```

$$i = 1$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 7 & 11 & 10 & 12 & 5 & 11 \\ 3 & 6 & 4 & 3 & 0 & 4 & 7 & 2 \\ 4 & 3 & 6 & 5 & 1 & 6 & 2 & 3 \\ 2 & 11 & 8 & 8 & 3 & 1 & 3 & 11 \\ 6 & 11 & 8 & 6 & 2 & 7 & 10 & 9 \\ 5 & 11 & 7 & 10 & 0 & 11 & 7 & 12 \\ 3 & 3 & 12 & 5 & 0 & 11 & 9 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \end{bmatrix}$$

$$11 + 3\,x + 2\,x^2 + 6\,x^3 + 10\,x^4 + 12\,x^5 + 9\,x^6$$

$$i = 2$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 7 & 11 & 10 & 12 & 5 & 11 \\ 3 & 6 & 4 & 3 & 0 & 4 & 7 & 2 \\ 4 & 3 & 6 & 5 & 1 & 6 & 2 & 3 \\ 2 & 11 & 8 & 8 & 3 & 1 & 3 & 11 \\ 6 & 11 & 8 & 6 & 2 & 7 & 10 & 9 \\ 5 & 11 & 7 & 10 & 0 & 11 & 7 & 12 \\ 3 & 3 & 12 & 5 & 0 & 11 & 9 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \end{bmatrix}$$

$$2 + 10\,x + 5\,x^2 + 12\,x^3 + 12\,x^4 + 7\,x^6 + 4\,x^7$$

$$i = 3$$

$$[\,x+3, 1, x^3+8\,x^2+4\,x+12, x^4+2\,x^3+3\,x^2+4\,x+6\,]$$

f ha sido descompuesto en un producto de factores, donde todos los subfactores tienen el mismo grado. Ahora analizamos como descomponer estos s factores,

$x+3$ debe tener factores de grado 1,

$x^3+8x^2+4x+12$ debe tener factores de grado 3,

$x^4+2x^3+3x^2+4x+6$ debe tener factores de grado 4.

En este caso, cada uno de los factores es irreducible y por tanto no será necesario usar el algoritmo de grado uniforme.

Referencias

[1] E.R.Berlekamp,*Factoring polynomials over large finite fields,*Math.Comp.,vol.24,1970,pp.713-735.

[2] E.R.Berlekamp,*Factoring polynomials over finite fields,*Bell System Tech.J.,vol.46,1967,pp.1853-1859.

[3] E.R.Berlekamp,*Algebraic Coding Theory,* McGraw-Hill,1968.

[4] D.Cantor and H.Zassenhaus,*A new Algorithm for factoring polynomials over finite fields*,Math.Comp.,vol.36(1981),pp.587-592.

[5] B.Buchberger,G.E.Collins, and R.Loos.*Computer Algebra, Simbolic and Algebraic Computation,*Springer-Verlag.1983.New York.

[6] J.calmet,*Algebraic Algorithms in GF(q).*Discrete Mathematics,No.56, vol.1,1985,pp.101-109.

[7] J.H. Davenport, Y. Siret, E. Tournier. *Computer Algebra: Systems and algorithms for Algebraic Computation* Academic press, 1988,England

[8] I.N.Herstein,*Topics in Algebra.*Jhon Wiley Sons.Inc.1975

[9] D.E.Knuth,*The Art of Computer Programming,*vol.2:*Seminumerical Algorithms,*Addisson-Wesley, Reading,Mass.,USA, 1969.

[10] John D.Lipson,*Elements of Algebra and Algebraic Computations*,Addison-wesley,1981

[11] M.Pohst and H. Zassenhaus,*Algorithmic Algebraic Number Theory,* Cambridge University Press, 1989, Cambridge, England.

[12] M.O.Rabin, *Probabilistic Algorithms in Finite Fields.*SIAM J. Comput.,9,1980, pp.273-288.

[13] V.Trevisan, and P. S. Wang, *Practical Factoricing univariante polynomial Factorization over Finite Fields*, proceddings of ISSAC; Bonn, Germany, Julio 1991.

[14] , http://mathworld.wolfram.com/Berlekamp-ZassenhausAlgorithm.html, accesado, 10 de Octubre , 2010.