Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Matemática Computacional
Recibido: 01/06/2018 | Aceptado: 10/09/2018

Algoritmos para la determinación de los homomorfismos de inmersión de Campos de Galois

Algorithms for determination of the immersion homomorphisms of Galois Fields

Oristela Cuellar Justiz[1*], Evaristo J. Madarro Capó[2], Guillermo Sosa Gómez[3], Gonzalo Palencia Fernández[4], Pablo Freyre Arrozarena[5]

[1]Universidad de las Ciencias Informáticas, La Habana, Cuba.
[2]Universidad de La Habana,La Habana, Cuba.
[3]Universidad de Guadalajara, Jalisco, México.
[4]Universidad Central de Las Villas, Villa Clara, Cuba.
[5]Universidad de La Habana, La Habana, Cuba.

*Autor para correspondencia: oristelacj@uci.cu

Resumen

Los homomorfismos entre estructuras algebraicas son de mucha utilidad tanto en la matemática, como en la ciencia de la computación. En particular, los homorfismos entre campos de Galois son utilizados en la criptografía, en los llamados esquemas de cifrado homomórfico Zhang and Yue (2013), y en la teoría de códigos, por ejemplo, en la denominada decodificación local Grigorescu et˜al. (2006). Por lo que puede ser necesario conocer cuáles son las funciones que constituyen homomorfismos entre campos de Galois. En este trabajo se propone un algoritmo para la determinación de los homomorfismos de inmersión que existen entre los campos $GF(p^n)$ y $GF(p^m)$ cuando $n \mid m$.

Palabras claves: Homomorfismo, Inmersión, Campos de Galois

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Abstract
Homomorphisms between algebraic structures are very useful in both mathematics and computer science. In particular, homomorphisms between Galois Fields are used in Cryptography, in the called homomorphic encryption schemes Zhang and Yue (2013), and in coding theory, for example, in the so-called local decoding Grigorescu et˜al. (2006). So it may be necessary to know what functions are homomorphisms between Galois Fields. In this work an algorithm for determining embedding homomorphisms between the $GF(p^n)$ and $GF(p^m)$ fields is proposed, when $n \mid m$.

Keywords: *Homomorphisms, Immersion,Galois Fields*

Introducción

El problema de la determinación de los homomorfismos de campos finitos es de gran importancia en la demostración de teoremas y propiedades, y desde el punto de vista computacional al permitir realizar los cómputos de manera más eficiente. La existencia de los homomorfismos muestra que los campos asociados no son esencialmente diferentes, sino que únicamente se diferencian en los nombres de sus elementos o en los símbolos de las operaciones. Esto implica que cualquier resultado conocido para un campo es válido para el otro.

En la literatura se reportan aplicaciones de los homomorfismos en varios esquemas de cifrado y en la teoría de códigos. En los materiales consultados para la realización de este trabajo encontramos aplicaciones de los homomorfismos en: los esquemas de cifrado homomórfico contemporáneo sobre campos numéricos ciclotómicos; en la Teoría de Códigos en los llamados decodificadores locales(LDCs) donde han jugado un papel central en muchos estudios.En el trabajo se presentan tres algoritmos para la determinación de los homomorfismos de inmersión entre campos de Galois de distinta cardinalidad,se realizan los análisis de complejidad de estos y se realizan pruebas numéricos para validar la selección del más eficiente.

Breve descripción sobre homomorfismos en Campos de Galois

Sean F y G campos finitos. Un homomorfismo de campos es una función $h : F \to G$ que satisface:

$$h(a+b) = h(a) + h(b),\ h(ab) = h(a) \cdot h(b) \tag{1}$$

$$h(0) = 0\ y\ h(1) = 1 \tag{2}$$

En Cuellar˜Justiz and Sosa˜Gómez (2013) se da una panorámica sobre los homomorfismos de inmersión y se propone un método para la determinación de los mismos. Retomemos las ideas esenciales.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Los campos $GF(p^m)$ y $GF(p^n)$ son extensiones algebraicas del campo primo $\mathbb{Z}_p = GF(p)$, de grados m y n respectivamente. Si α y β son elementos primitivos de los campos $GF(p^n)$ y $GF(p^m)$, respectivamente, son elementos de orden $p^n - 1$ y $p^m - 1$, respectivamente, en el grupo multiplicativo correspondiente. Un homomorfismo de inmersión $h : GF(p^n) \to GF(p^m)$ convierte α en un elemento β^k de su mismo orden, para ello es necesario que k sea múltiplo del entero $\frac{p^m-1}{p^n-1}$. La cantidad de valores que puede tomar k es igual a $\varphi(p^n - 1)$, siendo φ la función de Euler, que asigna a cada número natural la cantidad de naturales menores que $p^n - 1$ y primos relativos con él.

Un homomorfismo de campos h es también una aplicación lineal, es decir, un homomorfismo de espacios vectoriales, considerando ambos campos como espacios vectoriales sobre su subcampo primo $GF(p)$. El homomorfismo puede ser representado matricialmente, tomando como bases los sistemas $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$ y $\{1, \beta, \beta^2, \ldots, \beta^{m-1}\}$ de potencias, linealmente independientes, de los elementos primitivos α y β de los campos $GF(p^n)$ y $GF(p^m)$, respectivamente. Estas bases dan lugar a matrices $m \times n$ representantes de los diferentes homomorfismos de inmersión de $GF(p^n)$ en $GF(p^m)$.

Si $\frac{p^m-1}{p^n-1} = k_1$ y $k_t = tk_1$ para cada $t < p^n - 1$ primo relativo con $p^n - 1$, entonces, los elementos β^{k_t}, con $1 \leq t \leq \varphi(p^n-1)$, son los de orden p^n-1 en el campo $GF(p^m)$. Las funciones h_t de $GF(p^n)$ en $GF(p^m)$, definidas como $h_t(0) = 0$ y $h_t(\alpha^i) = \beta^{ik_t}$ para cada $i \in \{0, 1, 2, \ldots, p^n - 2\}$ son los homomorfismos multiplicativos entre $(GF(p^n))^*$ y $(GF(p^m))^*$.

Por el teorema de existencia de las aplicaciones lineales Molina et˜al. (2004) conocemos que para cada función h_t existe una única aplicación lineal $f_t : GF(p^n) \to GF(p^m)$ tal que $f_t(\alpha^i) = h_t(\alpha^i)$ para todo $i \in \{0, 1, 2, \ldots, n - 1\}$, ya que, $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$ es una base del espacio de partida $GF(p^n)$. Por tanto, los homomorfismos de inmersión son las funciones h_t que coinciden con su aplicación lineal asociada en todo su dominio $GF(p^n)$, es decir, son las que son multiplicativas, aditivas y $\mathbb{Z}_p$-lineales.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Algoritmos para la determinación de los homomorfismos sobre campos de Galois

Algoritmo ORISTO

Algoritmo 1 ORISTO

Entrada: Elementos primitivos α y β, las dimensiones n y m de los campos finitos, la característica p de los campos finitos y los polinomios definidores de los campos, el conjunto PR de todos los elementos primos relativos con el orden $p^n - 1$, del grupo multiplicativo del campo de partida.

Salida: Funciones que son homomorfismos de inmersión entre los campos finitos especificados.

1: $k := \frac{p^m-1}{p^n-1}$, y hacer $r :=\mid PR \mid$; $s := p^n - 1$; $b := 0$
2: Calcular y guardar la n-ésima potencia del elemento primitivo del campo de partida $t := \alpha^n$ como vector columna de n componentes.
3: Para cada entero $i \in PR$ se tienen los posibles homomorfismos $h_w(\alpha) = \beta^{kw}$ para $w = i$ y $w = s - i$,
4: **mientras** $b < n - 1$ **hacer**
5: Calcular las matrices M_w de orden $m \times n$ asociadas a las aplicaciones lineales $f_w(\alpha) = \beta^{kw}$ (sus columnas son las imágenes de los vectores de la base $\{1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{n-1}\}$ del campo de partida.)
6: Calcular $h_w(\alpha^n) = \beta^{kw \cdot n} = \beta^{kw \cdot (n-1)}\beta^{kw} = f_w(\alpha^{n-1})\beta^{kw}$.
7: Calcular $f_w(\alpha^n) = M_w \cdot t$, producto de una matriz M de orden $m \times n$ por un vector columna t de dimensión n.
8: Si para α^n se cumple que $f_w(\alpha^n) = h_w(\alpha^n)$, almacenar h_w e incrementar b.
9: **fin mientras**
10: **devolver** Imprimir las funciones que representan los homomorfismos de inmersión.

En Cuellar˜Justiz and Sosa˜Gómez (2013) se demostró que para que las funciones h_t y f_t sean la misma función es necesario y suficiente que se verifique la igualdad $h_t(\alpha^n) = f_t(\alpha^n)$. A partir de este teorema se diseñó este algoritmo que permite determinar los homomorfismos de campos de Galois de igual y diferente cardinalidad.

En el paso tres del algoritmo se tiene en cuenta de que si $mcd(s, i) = 1 \Rightarrow mcd(s, s - i) = 1$ y además $r =\mid PR \mid= \varphi(p^n - 1)$.

La complejidad del algoritmo depende de la complejidad del paso tres, es decir, de las operaciones exponenciación y el producto de un vector fila por una matriz. De esta manera, se tiene la siguiente notación asintótica:

$$O(rmax(((M); (n(E)))))$$

Donde E es la complejidad de la exponenciación y M la complejidad del producto de un vector fila por una matriz binaria. La complejidad del producto de una matriz de orden $m \times n$ por un vector columna de dimensión n es de $O(mn)$. La operación de exponenciación en $GF(p^m)$, tiene una complejidad de un $O(m^2log(p)log(m)log(log(m)))$, usando métodos basados en la transformada rápida de Fourier Gao et˜al. (2000) Huguet˜Rotger et˜al. (2012).

$$O(r \cdot max((mn); (nm^2log(p)log(m)log(log(m)))) \longrightarrow O(r(nm^2log(p)log(m)log(log(m))))$$

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Este algoritmo resulta costoso, hay que recorrer el espacio de búsqueda hasta encontrar todas las funciones que sumergen un campo en otro, es decir, los homomorfismos entre ambos campos. De manera que, a medida que la cardinalidad de los campos aumenta el tiempo de ejecución del algoritmo aumenta considerablemente y este puede llegar a ser impráctico.

Algoritmo ORISTO(Variante II)

Desde el punto de vista práctico, en cuanto a tiempo, el algoritmo ORISTO puede ser mejorado teniendo en cuenta la relación existente entre los homomorfismos de inmersión y el automorfismo de Frobenius analizada en Cuellar˜Justiz and Sosa˜Gómez (2013). Teniendo en cuenta esto se diseñó la siguiente variante del algoritmo anterior.

Algoritmo 2 ORISTO. Variante II

Entrada: Elementos primitivos α y β, las dimensiones n y m de los campos finitos, la característica p de los campos finitos y el conjunto PR de todos los elementos primos relativos con el orden $p^n - 1$, del grupo multiplicativo del campo de partida.
Salida: Funciones que son homomorfismos de inmersión entre los campos finitos especificados.
1: $k := \frac{p^m-1}{p^n-1}$, y hacer $r :=\mid PR \mid$; $s := p^n - 1$.
2: Calcular y guardar la n-ésima potencia del elemento primitivo del campo de partida $t := \alpha^n$ como vector columna de n componentes.
3: Para cada entero $i \in PR$, se tienen los posibles homomorfismos $h_w(\alpha) = \beta^{kw}$ para $w = i, s - i$,
4: **mientras** $b < n - 1$ **hacer**
5: Calcular las matrices M_w de orden $m \times n$ asociadas a las aplicaciones lineales $f_w(\alpha) = \beta^{kw}$ (sus columnas son las imágenes de los vectores de la base $\{1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{n-1}\}$ del campo de partida.)
6: Calcular $h_w(\alpha^n) = \beta^{kw \cdot n} = \beta^{kw \cdot (n-1)}\beta^{kw} = f_w(\alpha^{n-1})\beta^{kw}$.
7: Calcular $f_w(\alpha^n) = M_w \cdot t$, producto de una matriz M de orden $m \times n$ por un vector columna t de dimensión n.
8: Si para α^n se cumple que $f_w(\alpha^n) = h_w(\alpha^n)$, terminar e ir al paso 10.
9: **fin mientras**
10: Calcular las p-ésimas potencias de β^{kw} hasta $(\beta^{kw})^{p^{n-1}}$
11: **devolver** Imprimir las funciones que representan los homomorfismos de inmersión.

La complejidad computacional del algoritmo varía en cuanto al parámetro que determina el espacio de búsqueda. De manera que, el nuevo algoritmo solo analiza c elementos, donde c es la cantidad de elementos chequeados hasta encontrar el primer homomorfismo, con $c < r$. Así, se tiene la siguiente notación asintótica para esta variante

$$O(c\ max((mn); ((nm^2 log(p) log(m) log(log(m))))) \longrightarrow O(c(n\ m^2 log(p) log(m) log(log(m))$$

En el paso 4 del algoritmo las potencias se calculan elevando la función obtenida a la p-ésima potencia y así sucesivamente hasta encontrar los n homomorfismos de inmersión. Esta variante mejora el tiempo de cómputo pues alcanza de manera más rápida la solución final.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Algoritmo EVOR

A pesar de haber logrado realizar una disminución considerable en cuanto a tiempo de ejecución en la variante del algoritmo ORISTO utilizando los automorfismos de Frobenius, ahora se verá otro enfoque encontrado en of˜Tartu (10 de Abril de 2015) para determinar los homomorfismos de inmersión de un campo en otro, que permite disminuir la complejidad de las operaciones del algoritmo.

Sean $F = GF(p^n)$ y $G = GF(p^m)$ campos finitos de característica p y n es un divisor de m. Sean el campo F determinado por el polinomio irreducible f y α una raíz de dicho polinomio por tanto $f(\alpha) = 0$ y el campo G determinado por el polinomio irreducible g y β es una raíz de dicho polinomio $g(\beta) = 0$.

Para determinar un homomorfismo es suficiente determinar el valor de $h(\alpha)$ que satisface las igualdades (1) y (2). Un homomorfismo $h : F \to K$ tal que $h(\alpha) = s(\beta)$ existe si y solo si $s(\beta)$ es raíz del polinomio f of˜Tartu (10 de Abril de 2015). Como F y G son campos finitos cualquier homomorfismo $k : F \to G$ es inyectivo.

De acuerdo a lo analizado, para determinar los homomorfismos de campos finitos tenemos que hallar cuáles elementos del campo de llegada, son raíces del polinomio que define el campo de partida, en otras palabras hallar las raíces del polinomio que define el campo de partida, en el campo de llegada. Aunque en la literatura consultada se utiliza este procedimiento para determinar los isomorfismos de campos finitos definidos por diferentes polinomios, este método puede ser generalizado a la determinación de los homomorfismos de campos finitos de diferente cardinalidad.

La cantidad de homomorfismos de inmersión es igual al número de raíces del polinomio irreducible que define el campo de partida. Por lo tanto, el método de búsqueda de los homomorfismos de inmersión se reduce a la búsqueda de las raíces de f en el campo de llegada $GF(p^m)$. Aquí se necesita disponer de los algoritmos de búsqueda de raíces, la búsqueda exhaustiva puede ser útil para campos relativamente pequeños, pero para campos grandes no, ya que, el cardinal del espacio de búsqueda crece exponencialmente con el incremento de la dimensión del campo sobre el cual se realiza la búsqueda. Incluso si se utilizan los algoritmos de búsqueda de raíces Cuellar˜Justiz et˜al. (2013), Lidl and Niderraiter (1988), Lidl and Niederreiter (1994), Chen (1982), Von Zur˜Gathen and Panario (2001) a medida que el grado del polinomio en cuestión sea mayor, el costo computacional se incrementa considerablemente.

Se propone a continuación un algoritmo que evalúa el polinomio f en busca de las raíces pero utiliza también las ideas de diseño del algoritmo ORISTO. Esta propuesta que reduce el espacio de búsqueda de manera significativa, tiene en cuenta también que si para un polinomio f de grado n sobre un campo GF(p) una de sus raíces es α, el resto de las raíces son los elementos conjugados con α con respecto al campo $GF(p)$, o sea, $\alpha, \alpha^p, \alpha^{p^2}, \cdots, \alpha^{p^{n-1}}$ Lidl and Niderraiter (1988) Lidl and Niederreiter (1994) Mullen and Panario (2013).

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Los exponentes de dichas potencias pertenecen al mismo coseto p-ciclotómico módulo $p^n - 1$. Por lo tanto, es suficiente encontrar una de dichas raíces y además a la hora de realizar la búsqueda no es necesario explorar todo el campo es necesario solo ver cuál de las potencias del elemento primitivo del campo de llegada cuyo exponente es uno de los cosetos líderes (el menor de los elementos de la clase) es raíz del polinomio que define el campo de partida. También tenemos en cuenta que si el polinomio que define el campo de partida es primitivo de grado n, sus raíces son elementos de orden $p^n - 1$ y estas raíces se transforman por el homomorfismo en elementos de su mismo orden en el campo de llegada.

Algoritmo 3 EVOR

Entrada: Elemento primitivo β del segundo campo finito y el polinomio P que define el campo finito de partida y Q que define el campo de llegada, la dimensión n, m de los dos campos respectivamente, la característica p de los campos finitos, el conjunto CL de todos los cosetos líderes módulo $p^n - 1$, que son primos relativos con el orden del grupo multiplicativo del campo de partida.
Salida: Funciones que son homomorfismos de inmersión entre los campos finitos especificados.
1: $k := \frac{p^m-1}{p^n-1}$, y hacer $r :=\mid CL \mid$; $s := p^n - 1$.
2: Para cada entero $i \in CL$, calcular $h_i = i \cdot k$,
3: **mientras hacer**
4: Calcular β^{h_i} y guardar $t := \beta^{h_i}$
5: Si β^{h_i} es raíz del polinomio P, calcular los elementos conjugados con β^{h_i} con respecto a $GF(p)$ e ir al paso 7.
6: **fin mientras**
7: **devolver** Imprimir las funciones que representan los homomorfismos de inmersión.

El algoritmo en pseudocódigo aparece en el Anexo. La complejidad del algoritmo depende de la complejidad del paso dos es decir, principalmente, de las operaciones exponenciación y evaluación de polinomios. De esta manera, se tiene la siguiente notación asintótica:

$$O(cmax((E_v);(E)))$$

Donde (E) es la complejidad de la exponenciación y (E_v) es la complejidad de evaluar un elemento en un polinomio y c es la cantidad de cosetos líderes chequeados hasta encontrar el primer homomorfismo, con $c < r = \varphi(p^n - 1)/n = \varphi(s)/n$.

El polinomio $P(x) \in GF(p)[x]$. La evaluación de este polinomio de grado n sobre $GF(p^m)$, se realiza mediante el método de Horner que tiene un costo $O(n)$ operaciones. (19)(21), mientras que la operación de exponenciación en $GF(p^m)$, tiene una complejidad de un $O(m^2 log(p) log(m) log(log(m)))$, usando métodos basados en la transformada rápida de Fourier (20),(21).

Se tiene entonces, $O(c(m^2 log(p) log(m) log(log(m)))$

El cálculo de las p-ésimas potencias del homomorfismo es una simple multiplicación del exponente por p de manera sucesiva y reducción del mismo módulo $p^m - 1$, en caso de ser necesario.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

También puede tenerse en cuenta que si $(a_0, a_1, \cdots, a_{m-1})$ son las coordenadas de un elemento $a \in GF(q) = GF(p^m)$ en una base normal $(\alpha, \alpha^p, \cdots, \alpha^{p^{m-1}})$, entonces el elemento a^p tiene por coordenadas $(a_{m-1}, a_0, \cdots, a_{m-2})$. La elevación a la potencia p implica simplemente una permutación circular de los coeficientes y por tanto, su costo computacional es despreciable. Si $p = 2$ es la elevación al cuadrado, la que tiene costo cero Huguet˜Rotger et˜al. (2012).

Este algoritmo mejora la complejidad computacional de determinar los homomorfismos de inmersión entre campos finitos con respecto a ORISTO y ORISTO II.

Análisis Experimental

Los resultados de los experimentos numéricos realizados se recogen en la tabla 2 que muestra los tiempos de ejecución de los algoritmos para diferentes casos tomando como campo de partida $GF(2^8)$ .La tabla 3 muestra un análisis comparativo de los algoritmos en cuanto a su complejidad computacional. Las figuras 1 y 2 ayudan a interpretar los resultados obtenidos.

Tabla 1: PC donde se realizaron los cálculos.

Sistema Operativo	Windows 7 Professional 64 bits (6.1, compilación 7601)
CPU	Intel (R) Core (TM) i3-4160 CPU @ 3.60 GHz
Memoria	4.00 GB

Tabla 2: Tiempo de ejecución.

Algoritmos	2^8 a 2^{16}	2^8 a 2^{64}	2^8 a 2^{32}	2^8 a 2^{64}	2^8 a 2^{128}	2^8 a 2^{256}	2^8 a 2^{512}
Oristo	0.506	0.698	1.169	10.925	51.073	510.701	-
Oristo (Variante II)	0.015	0.032	0.042	2.42	10.805	63.901	386.912
EVOR	0.012	0.025	0.037	2.2.67	9.971	61.094	378.542

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

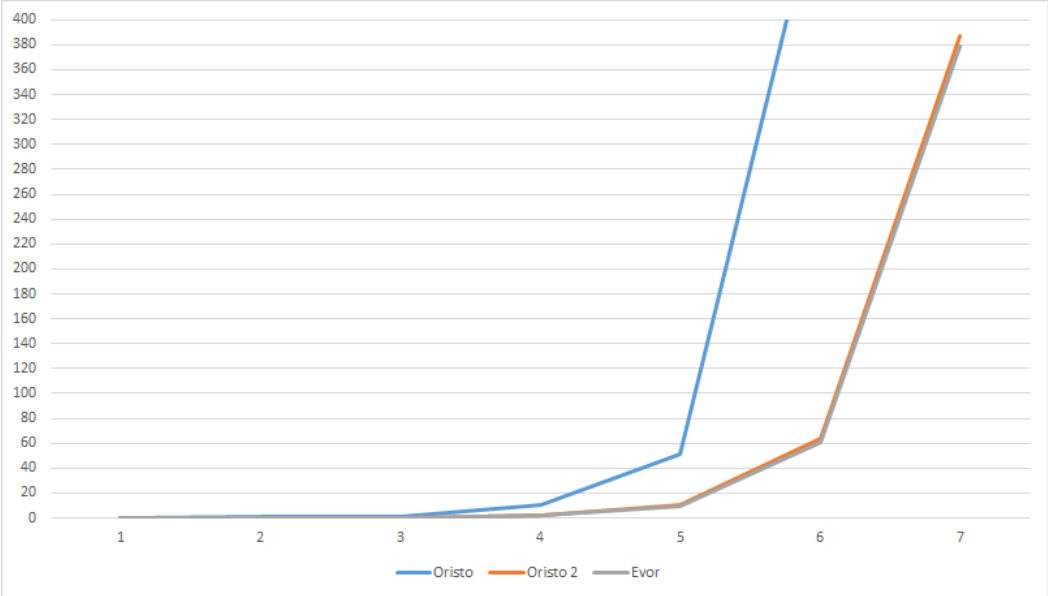

Figura 1: Visualización de los tres métodos en diferentes espacios.

De esta manera se puede observar que al igual que en la complejidad computacional, en la práctica, en cuanto a tiempo de ejecución, el algoritmo EVOR supera a los restantes algoritmos.

Tabla 3: Tabla comparativa de los métodos en cuanto a Complejidad Computacional.

Algoritmos	2^{16} a 2^{32}	2^{16} a 2^{64}	2^{16} a 2^{128}	2^{16} a 2^{256}	2^{16} a 2^{512}
Oristo	517.668	-	-	-	-
Oristo (Variante II)	0.84	117.229	12.941	382.73	-
EVOR	0.71	75.445	9.956	339.873	-

Conclusiones

Se cumplió el objetivo propuesto con la realización de este trabajo, al proponerse un algoritmo para la determinación de los homomorfismos de inmersión de un campo finito en otro de igual o mayor cardinalidad, que resultan muy útiles en la solución de diversos problemas. Las pruebas numéricas confirman que al igual que en la complejidad computacional, en cuanto a tiempo de ejecución el algoritmo EVOR despunta sobre los restantes algoritmos. Así, se concluye en la selección de este algoritmo para la determinación de los homomorfismos inmersión entre campos de Galois.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

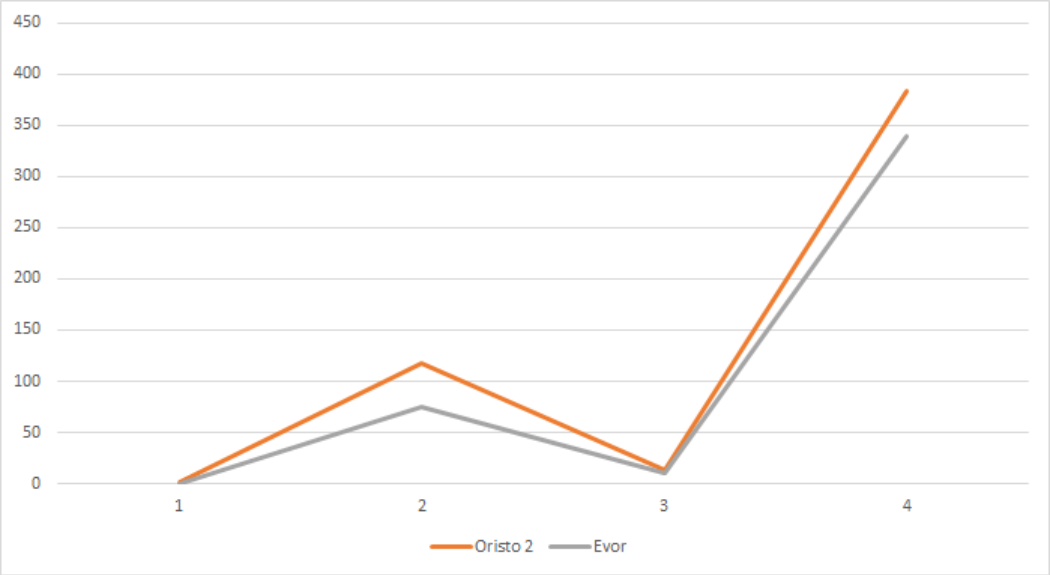

Figura 2: Visualización de los dos mejores métodos en diferentes espacios.

Anexos

Pseudocódigo de los algoritmos presentados para la determinación de los homomorfismos de inmersión en campos de Galois.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Algoritmo 4 ORISTO

1: $k := \frac{p^m-1}{p^n-1}$;
2: $r :=\mid PR \mid$;
3: $s := p^n - 1$;
4: $g := \alpha^n$;
5: $x := 0$;
6: **para** i:=1 to r **hacer**
7: **para** j:=0 to n-1 **hacer**
8: $M_i^j = \beta^{k \cdot i \cdot j}$;
9: $M_{s-i}^j = \beta^{k \cdot (s-i) \cdot j}$;
10: **fin para**
11: $t_i := M_i^{n-1} \cdot \beta^{k \cdot i}$;
12: $t_i := M_{s-i}^{n-1} \cdot \beta^{k \cdot (s-i)}$;
13: **si** $M_i \times g = t_i$ **entonces**
14: $\lambda_x := \beta^{k \cdot i}; x++$
15: **si no**, **si** $M_{s-i} \times g = t_{s-i}$ **entonces**
16: $\lambda_{x+1} := \beta^{k \cdot (s-i)}$; $x++$
17: **fin si**
18: **si** $x = n$ **entonces**
19: ir a paso 21
20: **fin si**
21: **fin para**
22: **devolver** λ

M_i^j es la columna j-ésima de la matriz M_i, asociada a la aplicación lineal f_i cuyas columnas son las imágenes de los vectores de la base $\{1, \alpha, \alpha^2, \alpha^3, \cdots, \alpha^{n-1}\}$

Algoritmo 5 ORISTO (Variante II)

1: $k := \frac{p^m-1}{p^n-1}$;
2: $r :=\mid PR \mid$;
3: $s := p^n - 1$;
4: $g := \alpha^n$;
5: **para** i:=1 to r **hacer**
6: **para** j:=0 to n-1 **hacer**
7: $M_i^j = \beta^{k \cdot i \cdot j}$;
8: $M_{s-i}^j = \beta^{k \cdot (s-i) \cdot j}$;
9: **fin para**
10: $t_i := M_i^{n-1} \cdot \beta^{k \cdot i}$;
11: $t_i := M_{s-i}^{n-1} \cdot \beta^{k \cdot (s-i)}$;
12: **si** $M_i \times g = t_i$ **entonces**
13: $\lambda_0 := \beta^{k \cdot i}$; ir a paso 18
14: **si no**, **si** $M_{s-i} \times g = t_{s-i}$ **entonces**
15: $\lambda_0 := \beta^{k \cdot (s-i)}$; ir a paso 18
16: **fin si**
17: **fin para**
18: **para** i:=1 to n-1 **hacer**
19: $\lambda_i := (\lambda_{i-1}^p)$
20: **fin para**
21: **devolver** λ

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 58-70
http://rcci.uci.cu

Algoritmo 6 EVOR

1: $k := \frac{p^m-1}{p^n-1}$;
2: $r :=\mid CL \mid$;
3: $s := p^n - 1$;
4: **para** i:=1 to r **hacer**
5: $h_i = i \cdot k$;
6: $t := \beta^{h_i}$;
7: **si** (t es una raíz de P) **entonces**
8: $\lambda_0^i = t$
9: **para** j:=1 to n-1 **hacer**
10: $\lambda_j^i := (\lambda_{j-1}^i)^p$
11: **fin para** ir a paso 14
12: **fin si**
13: **fin para**
14: **devolver** λ

Referencias

Chin-Long Chen. Formulas for the solutions of quadratic equations over GF(2^m). *Information Theory, IEEE Transactions on*, 28(5):792–794, 1982.

Oristela Cuellar Justiz and Guillermo and Sosa Gómez. Inmersión de un campo de Galois GF(2^n) en otro de mayor cardinalidad. *Revista Ciencias Matemáticas. Habana. Cuba*, 27, 2013.

Oristela Cuellar Justiz, Guillermo Sosa Gómez, Evaristo Madarro Capó, Luis Antonio Perfetti Villamil, Eberto Morgado Morales, and Yairon Cid Ruiz. Comparación de los métodos de solución de ecuaciones cuadráticas y cúbicas sobre campos finitos de característica dos. *Boletín de la Sociedad Cubana de Matemática y Computación*, only(Especial Number), 2013.

Shuhong Gao, Daniel Panario, Victor Shoup, et al. Algorithms for exponentiation in finite fields. *Journal of Symbolic Computation*, 29(6):879–889, 2000.

Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. Local decoding and testing for homomorphisms. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 375–385. Springer, 2006.

Llorenç Huguet Rotger, Josep Rifà Coma, and Juan Gabriel Tena Ayuso. Criptografia avançada, febrer 2012. 2012.

R Lidl and G Niderraiter. Konechnye polya. *Finite fields. Vol. 1]*, pages 433–822, 1988.

Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge university press, 1994.

Josefa Marín Molina, Ángel Balaguer Beser, and Elena Alemany Martínez. *Un curso de álgebra con ejercicios I.* Servicio de Publicaciones, 2004.

Gary L Mullen and Daniel Panario. *Handbook of finite fields.* CRC Press, 2013.

University of Tartu. `http://mathwiki.cs.ut.ee`, 10 de Abril de 2015.

Joachim Von Zur Gathen and Daniel Panario. Factoring polynomials over finite fields: A survey. *Journal of Symbolic Computation*, 31(1):3–17, 2001.

Long Zhang and Qiuling Yue. A fast integer-based batch full-homomorphic encryption scheme over finite field. *IACR Cryptology ePrint Archive*, 2013:793, 2013.