UNIVERSITAS INDONESIA

ALGORITMA PEMBENTUKAN FINITE FIELD

SKRIPSI

YANUAR SINGGIH SAPUTRA
030501067X

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
DEPARTEMEN MATEMATIKA
DEPOK
DESEMBER 2010

UNIVERSITAS INDONESIA

ALGORITMA PEMBENTUKAN FINITE FIELD

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains

YANUAR SINGGIH SAPUTRA
030501067X

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
DEPARTEMEN MATEMATIKA
DEPOK
DESEMBER 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Yanuar Singgih Saputra**
NPM	**: 030501067X**
Tanda Tangan	**:** 
Tanggal	**: 21 Desember 2010**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Yanuar Singgih Saputra
NPM : 030501067X
Program Studi : Sarjana Matematika
Judul Skripsi : Algoritma Pembentukan *Finite Field*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi S1 Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Dr. Sri Mardiyati, M.Kom. ( )

Pembimbing : Helen Burhan S.Si, M.Si ( )

Penguji : Dr. Sri Mardiyati, M.Kom. ( )

Penguji : Hengki Tasman, S.Si., M.Si. ( )

Penguji : Dr. Alhaji Akbar B. ( )

Ditetapkan di : Depok
Tanggal : 21 Desember 2010

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Sains Jurusan Matematika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia.

Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Ibu Dr. Sri Mardiyati, M.Kom, selaku dosen pembimbing I atas kesabaran, saran, dan bimbingannya selama ini.
2. Ibu Helen Burhan, S.Si, M.Si, selaku dosen pembimbing II atas kesabaran, saran, dan bimbingannya selama ini.
3. Ibu Dra. Rustina, selaku pembimbing akademis penulis atas saran, bimbingan, dan dorongan semangat selama penulis menempuh perkuliahan.
4. Seluruh Dosen Departemen Matematika yang telah memberikan ilmunya kepada saya selama 5 tahun dan menyemangati penulis untuk segera lulus.
5. Seluruh karyawan Departemen Matematika yang telah penulis repotkan selama ini.
6. Papa, Mama, Mimih, Kaka, Aa, dan segenap keluarga besar penulis yang telah memberikan bantuan dukungan material, doa, dan kasih sayang.
7. Sahabat-sahabat penulis, anak-anak Bunaya: Bembi, Gele, Gunung, Ilham, Reza dan Irwanto, atas hari-hari yang menyenangkan selama ini.
8. Keluarga penulis di Matematika, "kakek" Gunung, Nuts-neechan, dan Wicha "bulet", atas dukungan dan waktu yang diberikan selama ini.
9. Ai yang sudah memberikan banyak semangat, dukungan dan tujuan kepada penulis sampai saat ini.

10. Rekan-rekan Math ’05: Akmal, Gyo, Dia, Fika, Inul, Shally, serta teman-teman lain yang tidak mungkin disebutkan satu persatu. Terima kasih atas dukungan dan doanya selama ini.
11. Para “*Planeswalker*” : Ajat, Gyo, Dani, Fauzan, Oza, Rendy, Ilham, Bembi, Gele, Tomat, Faisal, yang telah membuat hari-hari penulis tidak membosankan.
12. Semua rekan-rekan senior dan junior di Math UI, yang telah banyak membantu penulis selama ini.
13. YUI yang selalu menemani dan menghilangkan kebosanan penulis selama penulisan skripsi.

Akhir kata, penulis berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu sains.

Penulis

2010

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Yanuar Singgih Saputra
NPM	: 030501067X
ProgramStudi	: Sarjana Matematika
Departemen	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Algoritma Pembentukan *Finite Field*

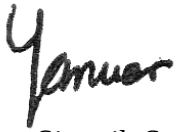
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia / formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 21 Desember 2010

Yang menyatakan



(Yanuar Singgih Saputra)

ABSTRAK

Nama : Yanuar Singgih Saputra
Program Studi : S1 Matematika
Judul : Algoritma Pembentukan *Finite Field*

Suatu field *F* disebut *finite field* jika banyak anggota F berhingga, sedangkan jika banyak anggota F tak berhingga maka field F disebut *infinite field*. Salah satu contoh *finite field* adalah *field* $GF(p^n)$ yang merupakan *extension field* dari Z_p. Untuk membentuk $GF(p^n)$ secara manual dengan elemen yang relatif besar cukup sulit, sehingga diperlukan bantuan komputer untuk membentuk $GF(p^n)$. Pembentukan $GF(p^n)$ ini terdiri atas pembentukan elemen-elemen pembangkit $GF(p^n)$ dan operasi-operasi yang didefinisikan di dalamnya.

Kata kunci : $GF(p^n)$, *finite field*, *extension field*, algoritma

xi + 44 halaman; 9 tabel; 10 gambar
Daftar Pustaka : 5 (1996 – 2009)

ABSTRACT

Name : Yanuar Singgih Saputra
Study Program : S1 Matematika
Title : Finite Field Construction Algorithm

Field F is a finite field if number of elements in F are finite, otherwise if F has infinite number of elements, F is an infinite field. An example of finite field is field $GF(p^n)$ that an extension field of Z_p. Constructing $GF(p^n)$ in manual with a relatively big number of elements is a hard way, so we need computer help to construct $GF(p^n)$. $GF(p^n)$ construction contains of contruction of $GF(p^n)$ generator and operations on it.

Keywords : $GF(p^n)$, finite field, extension field, algorithm

xi + 44 pages; 9 tables; 10 figures
References : 5 (1996 – 2009)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Kriptografi memiliki dua konsep utama, yaitu enkripsi dan dekripsi. **Enkripsi** adalah proses dimana pesan yang akan dikirim diubah menjadi suatu pesan rahasia. Sedangkan **dekripsi** adalah proses dimana sandi dikembalikan menjadi pesan asalnya.

Untuk mengubah suatu pesan menjadi sandi atau sebaliknya, digunakan suatu kunci yang harus dijaga kerahasiaanya. Berdasarkan kesamaan kuncinya, terdapat dua jenis kunci yaitu kunci simetris dan kunci asimetris. Kunci simetris menggunakan sebuah kunci rahasia yang sama untuk proses enkripsi dan dekripsi. Sedangkan kunci asimetris menggunakan dua buah kunci yang berbeda pada proses enkripsi dan dekripsi. Kedua kunci tersebut umumnya dikenal dengan sebutan *public key* dan *private key*.

Ilmu aljabar yang banyak digunakan dalam kriptografi adalah *finite field*. Beberapa sistem enkripsi menggunakan anggota *finite field* sebagai *public key* maupun *private key*.

Tingkat keamanan dari *key* tersebut cukup dipengaruhi oleh banyaknya anggota *finite field* yang digunakan. Namun, pembentukan *finite field* tertentu dengan jumlah anggota yang banyak cukup sulit untuk dilakukan secara manual, sehingga dibutuhkan bantuan komputer untuk membentuk *finite field* tersebut.

Salah satu metode untuk memperoleh suatu *finite field* dengan jumlah anggota yang banyak adalah dengan membentuk *extension field* dari *field* yang relatif kecil. Misalkan terdapat *field* F, maka dapat dicari suatu *field* E sedemikian sehingga $F \subset E$. Dalam hal ini, dikatakan bahwa E merupakan ***extension field*** dari F. Sehingga pada tugas akhir ini akan dibahas cara membentuk suatu *finite field* berdasarkan suatu *finite field* tertentu.

1.2. Perumusan Masalah

Masalah yang dibahas pada tugas akhir ini adalah

"Bagaimana algoritma pembentukan suatu *extension field* berdasarkan suatu *field* yang diberikan?"

1.3. Batasan Masalah

Batasan masalah dalam penulisan tugas akhir ini adalah bahwa algoritma pembentukan *extension field* digunakan hanya untuk *finite field*.

1.4. Tujuan Penulisan

Tujuan penulisan yang dilakukan dalam tugas akhir ini adalah membuat algoritma pembentukan *finite field* berdasarkan suatu *field* yang diberikan dan implementasinya dalam MATLAB.

1.5. Metode Penelitian

Metode penelitian yang digunakan dalam proses penulisan tugas akhir ini adalah studi literatur, yaitu dengan cara mempelajari buku-buku referensi yang berhubungan dengan topik tugas akhir serta mengambil data-data penunjang lainnya melalui internet.

1.6. Sistematika Penulisan

Penulisan tugas akhir ini akan dibagi dalam empat bagian besar, yang dimulai dengan Bab 1 Pendahuluan, yang berisi tentang latar belakang, perumusan masalah, batasan masalah, tujuan penulisan, metode penelitian, dan sistematika penulisan.

Selanjutnya adalah Bab 2 Landasan Teori, yang membahas mengenai dasar-dasar teori tentang *field*, ruang vektor atas *field*, polinomial atas *field* dan *extension field* yang akan digunakan dalam mengerjakan tugas akhir ini.

Algoritma pembentukan *finite field* akan dibahas pada Bab 3. Hasil implementasi pada komputer akan dibahas pada Bab 4. Sedangkan, kesimpulan dan saran mengenai tugas akhir ini akan dibahas pada Bab 5.

BAB 2
LANDASAN TEORI

Untuk membentuk suatu *finite field* diperlukan pemahaman tentang *field*, ruang vektor atas *field*, polinomial atas *field* dan *extension field*. Pada bab ini akan dibahas pengertian-pengertian tersebut beserta sifat-sifat yang akan digunakan dalam proses pembentukannya.

2.1. *Field*

Definisi 2.1.1

Misalkan S adalah himpunan tak kosong dan misalkan $S \times S$ adalah himpunan semua pasangan berurut (s, t) dimana $s, t \in S$. Maka pemetaan $S \times S$ pada S disebut **operas biner**. Himpunan S bersama satu atau lebih operasi biner pada S disebut dengan **sistem matematika**. (Lidl, Rudolf, and Niederreiter, Harald, 1997, p. 2)

Definisi 2.1.2

Sistem matematika $(F, +, \bullet)$ disebut ***field*** jika:

1. $(F, +)$ memenuhi aksioma-aksioma berikut:
   - Jika $a, b \in F$, maka $a + b = b + a$.
   - Jika $a, b, c \in F$, maka $a + (b + c) = (a + b) +$ c.
   - Terdapat elemen identitas $\mathbf{0} \in F$ sehingga untuk setiap $a \in F$ berlaku $a + \mathbf{0} = \mathbf{0} + a = a$.
   - Jika $a \in F$, maka terdapat $-a \in F$ sehingga $a + (-a) = (-a) + a = \mathbf{0}$.
2. $(F - \{\mathbf{0}\}, \bullet)$ memenuhi aksioma-aksioma berikut:
   - Jika $a, b \in F - \{\mathbf{0}\}$, maka $a \bullet b = b \bullet a$.
   - Jika $a, b, c \in F - \{\mathbf{0}\}$, maka $a \bullet (b \bullet c) = (a \bullet b) \bullet$ c.
   - Terdapat elemen unit $\mathbf{1} \in F - \{\mathbf{0}\}$ sedemikian sehingga untuk setiap $a \in F - \{\mathbf{0}\}$ berlaku $a \bullet \mathbf{1} = \mathbf{1} \bullet a = a$.

- Jika $a \in F - \{\mathbf{0}\}$, maka terdapat $a^{-1} \in F - \{\mathbf{0}\}$ sedemikian sehingga $a \bullet a^{-1} = a^{-1} \bullet a = \mathbf{1}$.

3. Jika $a, b, c \in F$, maka $a \bullet (b + c) = a \bullet b + a \bullet c$ dan $(b + c) \bullet a = b \bullet a + c \bullet a$.

(I.N. Herstein, 1996)

Berdasarkan banyak anggotanya, suatu *field F* disebut *finite field* jika banyak anggota *F* berhingga, sedangkan jika banyak anggota *F* tak berhingga maka *field F* disebut *infinite field*.

Finite field disebut juga *Galois Field*, dan untuk *Galois Field* yang memiliki *q* elemen dinotasikan dengan *GF*(*q*).

Berikut ini adalah contoh *finite field* dan *infinite field* :

1. Contoh *finite field* adalah himpunan bilangan bulat modulo *p* yang dinotasikan sebagai $\mathbb{Z}_p$, dengan *p* adalah bilangan prima, dengan operasi penjumlahan modulo *p* dan perkalian modulo *p*.
2. Contoh *infinite field* adalah himpunan bilangan rasional (dinotasikan sebagai $\mathbb{Q}$) dengan operasi penjumlahan dan perkalian.
3. Contoh lain dari *infinte field* adalah himpunan bilangan real (dinotasikan sebagai $\mathbb{R}$) dengan operasi penjumlahan dan perkalian.

Himpunan bagian *F'* dari suatu *field F* disebut ***subfield*** dari *F* jika *F'* membentuk *field* dengan operasi-operasi yang sama dengan *field F*. (Lidl, Rudolf, and Niederreiter, Harald, 1997, p. 30)

Seperti yang telah diketahui, himpunan bilangan rasional $\mathbb{Q}$ merupakan subset dari himpunan bilangan real $\mathbb{R}$. Jadi, *field* $\mathbb{Q}$ adalah subset dari *field* $\mathbb{R}$ atau dikatakan $\mathbb{Q}$ adalah *subfield* dari $\mathbb{R}$ terhadap operasi-operasi yang sama dengan $\mathbb{R}$.

Definisi 2.1.3

Field F yang tidak memiliki *proper subfield F'* disebut *prime field*. (Lidl, Rudolf, and Niederreiter, Harald, 1997, p. 30)

Definisi 2.1.4

Misalkan *x* adalah elemen *field F* dan *p* adalah suatu bilangan bulat positif tak nol, maka *px* didefinisikan sebagai penjumlahan *x* sebanyak *p* kali.

$$px = \underbrace{x + x + \cdots + x}_{(\ p \text{ kali }\ )}$$

Definisi 2.1.5

Suatu *field F* dikatakan memiliki **karakteristik** $p \neq 0$ jika terdapat bilangan bulat positif *p*, sedemikian sehingga untuk setiap $x \in F$ berlaku $px = \mathbf{0}$, dan tidak ada bilangan bulat positif lain yang lebih kecil dari *p* memenuhi sifat tersebut. (I. N. Herstein, 1996, p. 178)

Jika suatu *field F* tidak memiliki karakteristik $p \neq 0$ untuk bilangan bulat positif *p*, maka *field F* dikatakan *field* dengan karakteristik 0. Hal ini ditegaskan kembali pada teorema berikut ini

Teorema 2.1.6

Karakteristik dari suatu *field* adalah 0 atau suatu bilangan prima. (I. N. Herstein, 1996, p. 178)

Berikut ini adalah teorema yang menghubungkan karakteristik dari suatu *field* dan *prime subfield*-nya.

Teorema 2.1.7

Misalkan *F* adalah *field*. Jika *F* memiliki karakteristik 0, maka *prime subfield* dari *F* isomorfik dengan $\mathbb{Q}$; jika *F* memiliki karakteristik *p*, bilangan prima, maka *prime subfield* dari *F* isomorfik dengan $\mathbb{Z}_p$. (Lidl, Rudolf, and Niederreiter, Harald, 1997, p. 30)

Berikut ini akan dibahas pengertian dari suatu ruang vektor atas suatu *field*.

2.2. Ruang Vektor atas *Field*

Berikut akan diberikan pengertian ruang vektor atas suatu *field* dan definisi subruang dari suatu ruang vektor.

Definisi 2.2.1

Misalkan V merupakan himpunan tak kosong dan didefinisikan dua buah operasi pada V, yaitu operasi penjumlahan $+ : V \times V \rightarrow V$ dan operasi perkalian dengan skalar $\bullet : F \times V \rightarrow V$. Maka sistem matematika $(V, +, \bullet)$ disebut **ruang vektor atas *field*** F jika:

1. $(V, +)$ memenuhi:
   - Jika $v_1, v_2 \in V$ maka $v_1 + v_2 \in V$.
   - Jika $v_1, v_2, v_3 \in V$ maka $v_1 + (v_2 + v_3) = (v_1 + v_2) + v_3$.
   - Terdapat elemen identitas $\mathbf{0} \in V$ sehingga $v + \mathbf{0} = \mathbf{0} + v = v$ untuk setiap $v \in V$.
   - Jika $v \in V$, maka terdapat $-v \in V$ sehingga $v + (-v) = (-v) + v = \mathbf{0}$.
2. Jika $v \in V$ dan $\alpha \in F$, maka $\alpha \bullet v \in V$.
3. $\alpha \bullet (v_1 + v_2) = \alpha \bullet v_1 + \alpha \bullet v_2$, untuk setiap $\alpha \in F$ dan $v_1, v_2 \in V$.
4. $(\alpha + \beta)v = \alpha \bullet v + \beta \bullet v$, untuk setiap $\alpha, \beta \in F$ dan $v \in V$.
5. $\alpha \bullet (\beta \bullet v) = (\alpha \bullet \beta) \bullet v$, untuk setiap $\alpha, \beta \in F$ dan $v \in V$.
6. $\mathbf{1} \bullet v = v$ untuk setiap $v \in V$, dimana $\mathbf{1}$ adalah elemen unit dari F.

(I. N. Herstein, 1996)

Untuk penulisan selanjutnya, notasi F selalu digunakan untuk *field* sedangkan V selalu digunakan untuk ruang vektor atas *field* F.

Definisi 2.2.2

Subruang W dari ruang vektor V adalah subset tak kosong dari V sedemikian sehingga $\alpha \bullet w \in W$ dan $w_1 + w_2 \in W$ untuk setiap $\alpha \in F$ dan $w, w_1, w_2 \in W$. (I. N. Herstein, 1996, p. 181)

Teorema 2.2.3

Misalkan *V* adalah sembarang ruang vektor atas *F* dan $v_1, v_2, \dots, v_n$ adalah vektor-vektor dari *V*. Misalkan pula

$$\langle v_1, v_2, \dots, v_n \rangle = \{\alpha_1 \bullet v_1 + \alpha_2 \bullet v_2 + \cdots + \alpha_n \bullet v_n | \alpha_1, \alpha_2, \dots, \alpha_n \in F\}$$

Maka $\langle v_1, v_2, \dots, v_n \rangle$ adalah ruang vektor atas *F* dan subruang dari *V*. Subruang $\langle v_1, v_2, \dots, v_n \rangle$ disebut subruang dari *V* yang **dibangun** atau **direntang** oleh $v_1, v_2, \dots, v_n$ atas *F*. (I. N. Herstein, 1996, p. 181)

Definisi 2.2.4

Misalkan $V = \langle v_1, v_2, \dots, v_n \rangle$ ruang vektor yang direntang oleh $\{v_1, v_2, \dots, v_n\}$. Maka, Ruang vektor *V* atas *F* dikatakan berdimensi hingga. (I. N. Herstein, 1996, 185)

Definisi 2.2.5

Misalkan *V r*uang vektor atas *F*; maka $\{v_1, v_2, \dots, v_n\}$ dengan $v_i \in V$ dikatakan bebas linier jika $\alpha_1 \bullet v_1 + \alpha_2 \bullet v_2 + \cdots + \alpha_n \bullet v_n = 0$, dimana $\alpha_1, \dots, \alpha_n \in F$, hanya dipenuhi oleh nilai-nilai $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$. (I. N. Herstein, 1996, p. 185)

Jika $\{v_1, v_2, \dots, v_n\}$dengan $v_i \in V$ tidak bebas linier maka $v_1, v_2, \dots, v_n$ dikatakan bergantung linier.

Berdasarkan definisi-definisi di atas, basis dari suatu ruang vektor *V* atas *F* didefinisikan sebagai berikut.

Definisi 2.2.6

Misalkan *V* adalah ruang vektor berdimensi hingga atas *F*; maka $v_1, v_2, \dots, v_n$ adalah **basis** dari *V* atas *F* jika $v_1, v_2, \dots, v_n$ merentang *V* atas *F* dan bebas linier. (I. N. Herstein, 1996, p. 187)

Teorema 2.2.7

Misalkan *V* adalah ruang vektor atas *F* yang berdimensi hingga. Maka dua sembarang basis dari *V* atas *F* memiliki jumlah anggota yang sama, dan jumlah tersebut merupakan **dimensi** dari *V* atas *F*, dinotasikan dengan $\dim_F(V)$. (I. N. Herstein, 1996, p. 187)

Berikut ini adalah teorema-teorema yang berkaitan dengan suatu ruang vektor berdimensi hingga.

Teorema 2.2.8

Misalkan *V* adalah ruang vektor atas *F* sedemikian sehingga $\dim_F(V) = n$. Jika $m > n$, maka setiap himpunan bagian dari *V* dengan jumlah vektor *m* bergantung linier. (I. N. Herstein, 1996, p. 188)

Teorema 2.2.9

Misalkan *V* adalah ruang vektor atas *F* dengan $\dim_F(V) = n$. Maka sembarang *n* vektor dari *V* yang saling bebas linier membentuk basis dari *V* atas *F*. (I. N. Herstein, 1996, p. 188)

2.3. Polinomial atas *Field*

Pada subbab ini akan dibahas mengenai polinomial atas suatu *field F*. Polinomial tersebut berguna dalam pembentukan *finite field*.

Misalkan *F* suatu *field*, himpunan semua $p(x) = a_0 + a_1x + \cdots + a_nx^n, n \geq 0$ dimana $a_i \in F,\ i = 0, 1, \ldots, n$ dinotasikan dengan $F[x]$. (I. N. Herstein, 1996, p. 152)

Dalam $F[x]$ persamaan, penjumlahan, dan perkalian didefinisikan sebagai berikut.

1. Persamaan
   $p(x) = a_0 + a_1x + \cdots + a_nx^n$ dan $q(x) = b_0 + b_1x + \cdots + b_nx^n$ dikatakan sama jika dan hanya jika $a_i = b_i$ untuk $i \geq 0$.

2. Penjumlahan

   Misalkan $p(x) = a_0 + a_1x + \cdots + a_nx^n$ dan $q(x) = b_0 + b_1x + \cdots + b_mx^m$, maka $p(x) + q(x) = c_0 + c_1x + \cdots + c_sx^s$ dimana $c_i = a_i + b_i$ untuk setiap $i \geq 0$.

3. Perkalian

   Misalkan $p(x) = a_0 + a_1x + \cdots + a_nx^n$ dan $q(x) = b_0 + b_1x + \cdots + b_mx^m$, maka $p(x)q(x) = c_0 + c_1x + \cdots + c_tx^t$ dimana $c_t = a_ib_0 + a_{i-1}b_1 + \cdots + a_1b_{i-1} + a_0b_i$ untuk setiap $i \geq 0$.

Definisi 2.3.1

Jika $p(x) = a_0 + a_1x + \cdots + a_nx^n$ dan $a_n \neq 0$, maka derajat dari *p*(*x*), dilambangkan dengan deg *p*(*x*), adalah *n*. (I. N. Herstein, 1996, p. 153)

Salah satu tipe polinomial yang digunakan dalam pembentukan *finite field* adalah polinomial *monic*. Berikut ini adalah definisi polinomial *monic*.

Definisi 2.3.2

f (*x*) ∈ *F*[*x*] adalah **polinomial *monic*** jika koefisien dari pangkat tertingginya adalah 1. (I. N. Herstein, 1996, p. 157)

Jadi, *f* (*x*) dikatakan *monic* jika memenuhi

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0$$

Seperti yang telah dibahas sebelumnya, kita dapat mengalikan dua buah polinomial dalam *F*[*x*]. Dengan analogi seperti pada bilangan bulat, maka suatu polinomial dapat membagi polinomial lainnya yang dinyatakan pada definisi formal berikut.

Definisi 2.3.3

Misalkan $f(x), g(x) \in F[x]$, dengan $g(x) \neq 0$. $g(x)$ membagi $f(x)$, ditulis dengan $g(x) \mid f(x)$, jika terdapat $a(x) \in F[x]$ sehingga $f(x) = a(x)\, g(x)$.

(I. N. Herstein, 1996, p. 157)

Pada bilangan bulat dikenal istilah **faktor persekutuan terbesar**. Selanjutnya akan dibahas pengertian faktor persekutuan terbesar pada $F[x]$ dan teorema-teorema yang berkaitan.

Definisi 2.3.4

Untuk setiap polinomial $f(x)$ dan $g(x) \in F[x]$, faktor persekutuan terbesar dari $f(x)$ dan $g(x)$ adalah polinomial *monic* $d(x)$, yang bersifat:

(a) $d(x) \mid f(x)$ dan $d(x) \mid g(x)$.

(b) Jika $h(x) \mid f(x)$ dan $h(x) \mid g(x)$, maka $h(x) \mid d(x)$.

(I. N. Herstein, 1996, p. 158)

Definisi 2.3.5

Polinomial $f(x), g(x) \in F[x]$ dikatakan saling **relatif prima** jika faktor persekutuan terbesarnya adalah 1. (I. N. Herstein, 1996, p. 158)

Definisi 2.3.6

Polinomial $p(x) \in F[x]$ dengan deg $p(x) > 0$ dikatakan *irreducible* jika untuk setiap polinomial $f(x) \in F[x]$, maka hubungan yang mungkin terjadi adalah $p(x) \mid f(x)$ atau $p(x)$ relatif prima dengan $f(x)$. (I. N. Herstein, 1996, p. 159)

Definisi 2.3.7

Suatu elemen a disebut akar dari suatu polinomial $f(x) \in F[x]$ jika $f(a) = 0$. (I.N. Herstein, 1996)

Teorema 2.3.8

Jika polinomial berderajat n, $h(x) \in \mathbb{Z}_p[x]$ *irreducible*, maka $h(x) \mid (x^q - x)$, dimana $q = p^n$. (I. N. Herstein, 1996, p.228)

Teorema 2.3.9

Jika $n > 0$, maka $f(x) = x^q - x$, dimana $q = p^n$, tidak memiliki akar yang berulang di *field* yang memiliki karakteristik p. (I.N. Herstein, 1996, p. 226)

2.4. *Extension Field*

Pembentukan suatu *finite field* didasari atas keberadaan *extension field* dari suatu *finite field*. Berikut ini diberikan beberapa pengertian dan teorema tentang *extension field* yang akan digunakan untuk membangun suatu *finite field* berdasarkan suatu *field* yang diberikan.

Definisi 2.4.1

Misalkan F dan E *field*, maka E disebut *extension field* dari F jika F adalah *subfield* dari E. (I. N. Herstein, 1996, p. 191)

Untuk selanjutnya, notasi E akan menyatakan *extension field* dari F.

Akibat Definisi 2.4.1, jika dipandang dari sudut ruang vektor maka berlaku teorema berikut ini.

Teorema 2.4.2

Jika E adalah *extension field* dari F, maka E merupakan suatu ruang vektor atas *field* F. (I. N. Herstein, 1996)

Jika kita memandang E sebagai suatu ruang vektor atas *field* F, maka dimensi dari E atas F, $\dim_F(E)$, disebut *degree* dari E atas F dan dinotasikan sebagai $[E : F]$. (I. N. Herstein, 1996, p. 191). Jika $[E : F]$ *finite*, maka E disebut *finite extension* dari F.

Teorema 2.4.3

Misalkan F adalah *finite field* dengan karakteristik p suatu bilangan prima. Maka F memiliki $q = p^n$ elemen, dimana $n = [\, F : \mathbb{Z}_p \,]$ dan polinomial $x^q - x$ di $\mathbb{Z}_p[x]$ terbagi atas faktor-faktor linier di $F[x]$ yaitu

$$x^q - x = (x - a_1)(x - a_2) \dots (x - a_q)$$

dimana $a_1, a_2, \dots, a_q \in F$. (I.N. Herstein, 1996, p. 225)

Berikut ini adalah beberapa teorema yang berhubungan dengan *finite extension* dari suatu *field*.

Teorema 2.4.4

Misalkan E adalah *finite extension* dari F yang berderajat n. Maka, untuk setiap $u \in E$ terdapat $\alpha_0, \alpha_1, \dots, \alpha_n \in F$, yang tidak semuanya 0, sedemikian sehingga

$$\alpha_0 + \alpha_1 u + \cdots + \alpha_n u^n = 0$$

(I. N. Herstein, 1996, p. 193)

Definisi 2.4.5

Jika $E \supset F$ adalah *field*, maka $a \in E$ dikatakan *algebraic* atas F jika terdapat polinomial $p(x) \neq 0 \in F[x]$ sedemikian sehingga $p(a) = 0$. (I. N. Herstein, 1996, p. 193)

Berdasarkan Teorema 2.4.4 dan Definisi 2.4.5 maka untuk setiap $u \in E$, terdapat polinomial $p(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_n x^n$ dalam $F[x]$ sedemikian sehingga berlaku $p(u) = 0$. Berdasarkan hal tersebut, diberikan definisi berikut.

Definisi 2.4.6

Suatu elemen $a \in E \supset F$ disebut *algebraic* dengan derajat n atas F jika terdapat polinomial $p(x) \in F[x]$ dengan derajat n sedemikian sehingga $p(a) = 0$

dan tidak ada polinomial lain dalam $F[x]$ dengan derajat kurang dari n yang memenuhi sifat tersebut. (I. N. Herstein, 1996, p. 195)

Polinomial $p(x)$ pada definisi di atas dapat dibagi dengan koefisien dari x^n untuk memperoleh polinomial monik $q(x) \in F[x]$ yang memiliki derajat sama dengan $p(x)$ sehingga $q(a) = 0$. Jadi, dapat diasumsikan bahwa $p(x)$ merupakan polinomial monik. Polinomial monik $p(x)$ disebut polinomial minimal untuk a atas F.

Berikut ini diberikan lemma dan definisi yang berhubungan dengan polinomial minimal $p(x)$ dan *extension field*.

Lemma 2.4.7

Misalkan $a \in E$ *algebraic* atas F dengan polinomial minimal $p(x) \in F[x]$, maka $p(x)$ *irreducible* dalam $F[x]$. (I. N. Herstein, 1996, p. 195)

Berdasarkan Teorema 2.4.4 dan Lemma 2.4.7 dapat dikatakan bahwa untuk suatu polynomial $p(x) \in F[x]$, dengan $p(x)$ *irreducible* di $F[x]$, maka akan terdapat suatu *extension field* E dari F sedemikian sehingga $p(x) \in E[x]$ dapat diuraikan atas perkalian faktor-faktor linier dan E merupakan *extension field terkecil* yang mengandung akar-akar ini.

Misalkan $p(x) \in F[x]$ adalah polinomial *irreducible* berderajat $n > 0$ dan $\alpha \notin F$ adalah akar dari $p(x)$. Maka kumpulan semua kombinasi anggota F dengan $\alpha, \alpha^2, \ldots, \alpha^{n-1}$ adalah *field* terkecil yang memuat F dan α dinotasikan dengan $F(\alpha)$.

Definisi 2.4.9

$F(a)$ merupakan *extension field* dari F dan disebut *splitting field* untuk $p(x)$ atas F. (Neil Koblitz, 1999)

Berikut diberikan definisi untuk akar lain dari suatu polinomial minimal $p(x)$ untuk a atas F.

Definisi 2.4.10

Misalkan $p(x)$ polinomial minimal dari a atas F. Suatu akar lain a' dari $p(x)$ disebut konjugate dari a atas F. (Neil Koblitz, 1999)

Hubungan antara a dan a' sebagai konjugate dari a atas F diberikan oleh teorema berikut.

Teorema 2.4.11

Jika a' adalah konjugate dari a atas F, maka *field* $F(a)$ dan $F(a')$ isomorfik. (Neil Koblitz, 1999)

BAB 3
ALGORITMA PEMBENTUKAN *FINITE FIELD*

3.1. Algoritma Pembentukan *Finite Field*

Berdasarkan Teorema 2.4.3 untuk membentuk suatu *finite field* yang memiliki $q = p^n$ elemen diperlukan suatu polinomial $x^q - x$ di $\mathbb{Z}_p[x]$. Sedangkan, berdasarkan Teorema 2.3.8, suatu polinomial *irreducible* berderajat n, $h(x) \in \mathbb{Z}_p[x]$ akan habis membagi $x^q - x$.

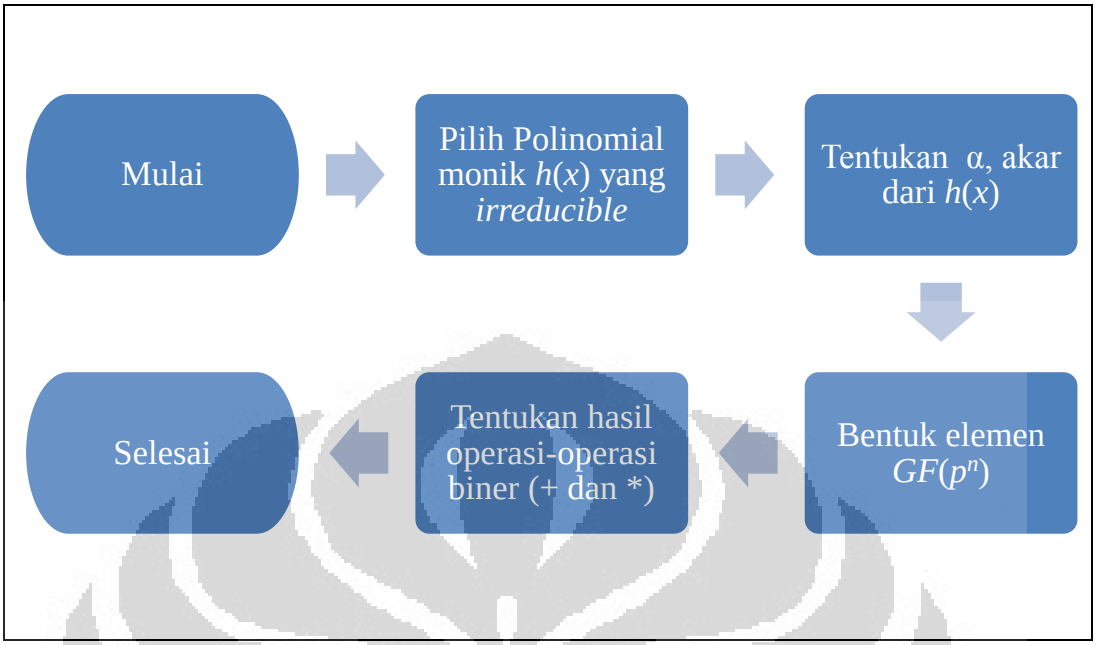
Jadi, dapat disimpulkan bahwa untuk membentuk suatu *finite field* yang memiliki $q = p^n$ elemen, diperlukan suatu polinomial *irreducible* $h(x)$ yang berderajat n di $\mathbb{Z}_p[x]$.

Secara umum, algoritma pembentukan *finite field* $GF(p^n)$ adalah sebagai berikut:

1. Pilih polinomial monik $h(x)$ yang *irreducible* berderajat n di $\mathbb{Z}_p[x]$.
2. Tentukan akar $\alpha \notin \mathbb{Z}_p$ dari polinomial $h(x)$.
3. Bentuk $GF(p^n)$ yaitu himpunan semua kombinasi linear dari α dan elemen $\mathbb{Z}_p$.
4. Tentukan hasil operasi-operasi biner (+ dan *) di $GF(p^n)$.

Karena polinomial monik yang *irreducible* berderajat n di $\mathbb{Z}_p[x]$ jumlahnya lebih dari satu, maka untuk setiap $h(x)$ yang dipilih dapat dibentuk suatu *splitting field* dari $\mathbb{Z}_p[x]$. Elemen-elemen pada masing-masing *extension field* tersebut sama, tetapi berbeda pada operasi yang didefinisikan di *field* tersebut.

Berikut ini adalah diagram alur dari algoritma pembentukan *finite field*



Gambar 3.1. Diagram Alur Algoritma Pembentukan *Finite Field*

Berikut ini akan dijelaskan tahap-tahap yang dapat dilakukan pada tiap langkah algoritma pembentukan *finite field* sesuai dengan teori-teori yang telah dijelaskan sebelumnya.

3.1.1. Algoritma Pemilihan Polinomial Monik $h(x)$ di $\mathbb{Z}_p[x]$

Untuk mendapatkan kumpulan polinomial monic berderajat n di $\mathbb{Z}_p[x]$, akan digunakan algoritma berikut ini

Tabel 3.1. Algoritma Monik

Input : p dan n

Output: P, matriks berukuran $[p^n,(n+1)]$ yang berisi koefisien dari polinomial monic $a_0 + a_1x + ... + x^n$ di $\mathbb{Z}_p[x]$

```
q = p^n;
P = ones(q,n+1); %Matriks satuan berukuran [q,(n+1)]
isi = 0;
brs = 1;
for i=1 to n
    for j=1 to q/p^(i-1)
        for k=1 to p^(i-1)
            P(brs,i) = isi;
            brs = brs+1;
        end for
        isi = mod(isi+1,p);
    end for
    brs=1;
end for
```

Berdasarkan Definisi 2.3.6, polinomial $h(x)$ dikatakan *irreducible* jika untuk setiap $f(x) \in \mathbb{Z}_p[x]$ maka $h(x)$ relatif prima dengan $f(x)$. Jadi, untuk mendapatkan $h(x)$, suatu polinomial monik yang *irreducible* berderajat n di $\mathbb{Z}_p[x]$, dapat dilakukan langkah-langkah berikut ini

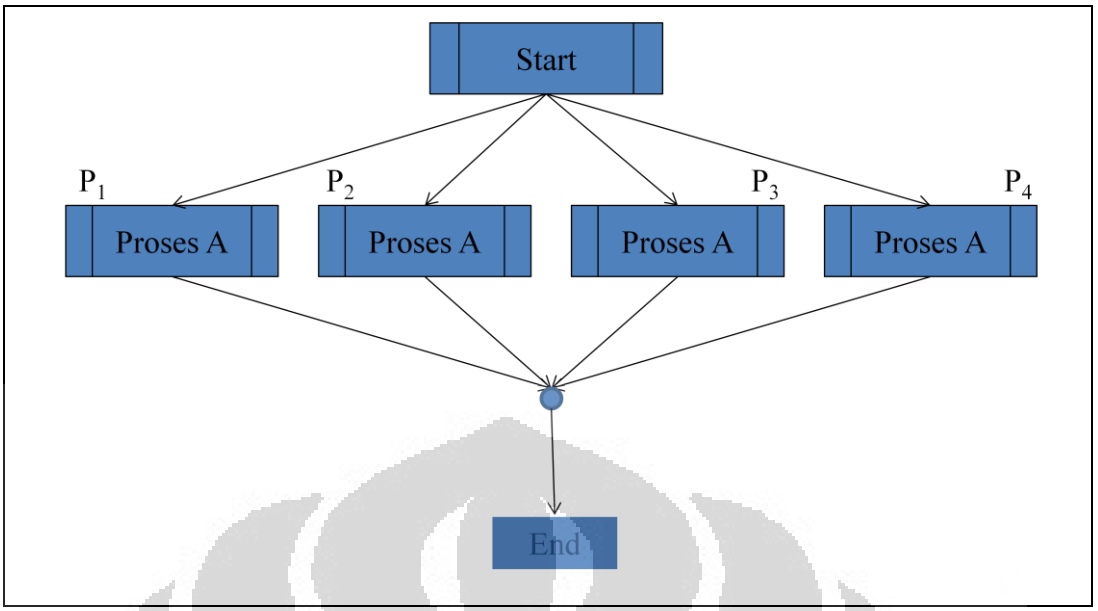
Tabel 3.2. Algoritma Pemilihan Polinomial *Irreducible* $h(x)$ di $\mathbb{Z}_p[x]$

Input: P, kumpulan polinomial monic berderajat n di $\mathbb{Z}_p[x]$
F, kumpulan polinomial berderajat $f < n$ di $\mathbb{Z}_p[x]$
n, derajat polinomial-polinomial P
Output: H, kumpulan polinomial monic berderajat n yang *irreducible* di $\mathbb{Z}_p[x]$

```
i = 1;
while (P(i, 1:end) ≠ NULL) do
    px = P(i, 1:end);
    j = 1;
    fx = F(j);
    while (F ≠ NULL) and (mod(px, fx) ≠ 0) do
        j = j + 1;
        fx = F(j);
    end while
    if (fx == NULL)
        add(H, px);
    end if
    i = i + 1;
end while
```

Proses pemilihan polinomial yang *irreducible* di atas akan memiliki kompleksitas yang cukup besar, karena pada kasus terburuknya, algoritma tersebut harus memeriksa pembagian polinomial *px* dan *fx* sebanyak $q = p^n$. Hal tersebut akan menyebabkan waktu eksekusi proses ini menjadi lambat.

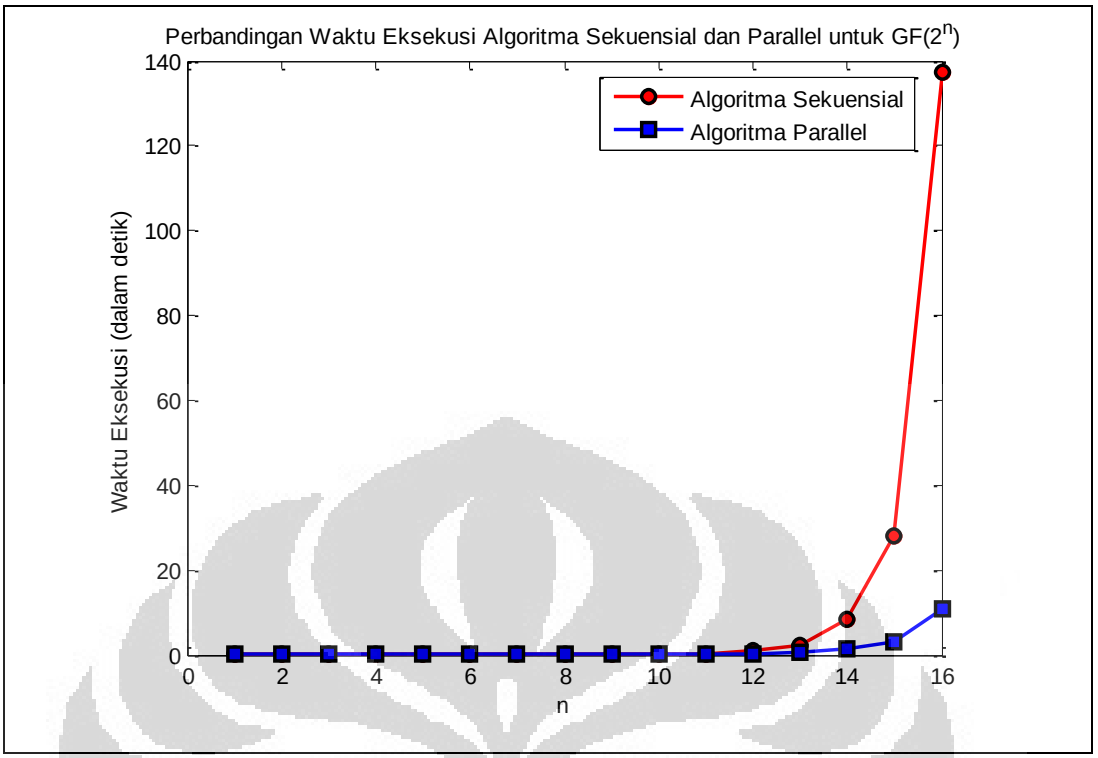
Untuk mempercepat waktu eksekusi, dapat dilakukan perhitungan paralel pada proses ini dengan cara membagi data untuk dikerjakan oleh beberapa *core* prosesor secara bersamaan. Berikut ini adalah gambar yang menggambarkan proses paralel yang digunakan.



Gambar 3.2. Struktur Paralel dalam Algoritma Pemilihan Polinomial Monik

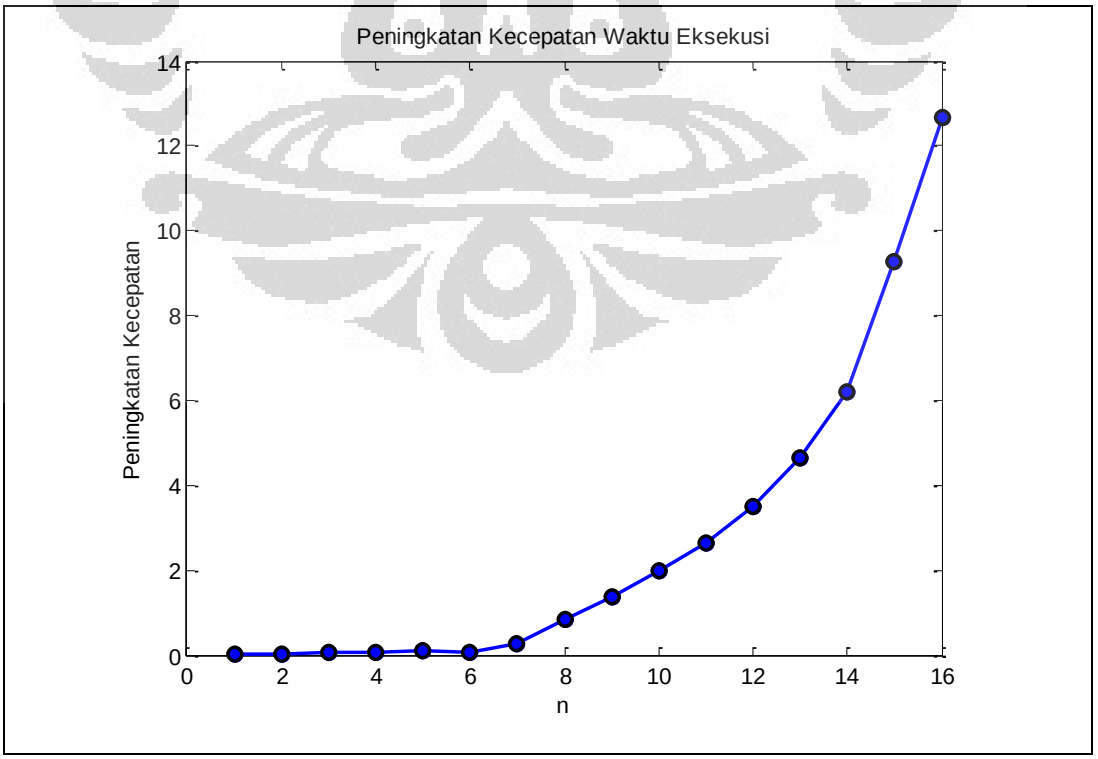
Pada Gambar 3.2 di atas, Proses A adalah proses yang menjalankan algoritma di Tabel 3.2. Sedangkan ● menggambarkan proses penggabungan data yang ada pada keempat prosesor.

Berikut ini adalah grafik perbandingan waktu eksekusi antara program yang menerapkan algoritma secara sekuensial dan paralel, pada saat pemilihan polinomial *irreducible* di $\mathbb{Z}_2[x]$ untuk membangun $GF(2^n)$ dengan $n = 1, 2, \ldots, 16$.



Gambar 3.3. Grafik Perbandingan Waktu Eksekusi Algoritma Sekuensial dan Parallel

Dari data waktu eksekusi yang diperoleh, maka dapat diperoleh grafik peningkatan kecepatan waktu eksekusi seperti berikut



Gambar 3.4. Grafik Peningkatan Waktu Eksekusi

Grafik di atas diperoleh saat program menjalankan algoritma paralel dengan menggunakan 4 *core* yang dimiliki prosesor.

Langkah selanjutnya pada pembentukan *finite field* adalah memilih $h(x) \in H$ dan mencari akar $\alpha \notin \mathbb{Z}_p$ dari $h(x)$. Langkah ini merupakan langkah trivial, sehingga tidak akan dibahas lebih lanjut.

Selanjutnya akan dibentuk elemen $GF(p^n)$ yang merupakan kombinasi linear dari α dan elemen $\mathbb{Z}_p$.

3.1.2. Algoritma Pembentukan Elemen GF(p^n)

Elemen $GF(p^n)$ dapat dibentuk dengan cara mengkombinasikan α dengan elemen-elemen $\mathbb{Z}_p$. Hal ini dapat dilakukan dengan langkah-langkah sebagai berikut:

Tabel 3.3. Algoritma Pembentukan Elemen $GF(p^n)$

Input : p, n, dan P
Output : gf, matriks berukuran [p^n, n] yang berisi koefisien dari kombinasi linear α dan anggota di $\mathbb{Z}_p$[x]

```
q = p^n;
gf = [ ];
isi = 0;
brs = 1;
for i=1 to n
    for j=1 to q/p^(i-1)
        for k=1 to p^(i-1)
            gf(brs,i) = isi;
            brs = brs+1;
        end for
        isi = mod(isi+1,p);
    end for
    brs=1;
end for
```

Langkah selanjutnya adalah menentukan operasi-operasi yang berlaku pada $GF(p^n)$.

3.1.3. Algoritma Pembentukan Operasi pada $GF(p^n)$

Misalkan pada $GF(p^n)$ didefinisikan dua buah operasi biner (+ dan *), maka dapat dibentuk tabel hasil operasi untuk masing-masing operator. Pada $GF(p^n)$, operasi + didefinisikan sebagai penjumlahan modulo p. Sedangkan operasi * didefinisikan sebagai perkalian polinomial modulo $h(x)$

Berikut ini diberikan algoritma yang dapat digunakan untuk membentuk tabel hasil operasi + pada $GF(p^n)$

Tabel 3.4. Algoritma Pembentukan Hasil Operasi + pada $GF(p^n)$

Input : gf, p, n

Output: T, cell berukuran [p^n, p^n] yang berisi koefisien dari penjumlahan setiap anggota di GF(p^n) yang bersesuaian dengan indeks cell

```
q = p^n
for i=1 to q
    for j=i to q
        t = gf(i,1:end) + gf(j,1:end);
        t = mod(t, p);
        T{i,j} = t;
        T{j,i} = t;
    end for
end for
```

Berikut ini diberikan algoritma yang dapat digunakan untuk membentuk tabel hasil operasi * pada $GF(p^n)$

Tabel 3.5. Algortima Pembentukan Hasil Operasi * pada $GF(p^n)$

Input : gf, p, n

Output: T, cell berukuran [p^n, p^n] yang berisi koefisien dari perkalian setiap anggota di GF(p^n) yang bersesuaian dengan indeks cell

```
q = p^n
h1 = fliplr(h); % membalik urutan polinomial
for i=1 to q
    for j=i to q
        % Fungsi poltimes adalah fungsi untuk melakukan
        % perkalian polinomial
        t = fliplr(poltimes(gf(i,:),gf(j,:)));
        t = mod(t, p);
        % Fungsi deconv adalah fungsi untuk melakukan
            % pembagian polinomial. s adalah hasil pembagian dan
            % r adalah sisa pembagian
        [s r] = deconv(t,h1);
        r = fliplr(mod(r,p));
        % Menyamakan panjang polinomial
        if (length(r) < n)
            r = [r zeros(1, n-length(r))];
        end if
        if (length(r) > n) and (r(n+1) == 0)
            r = r(1:n);
        end if
        T{i,j} = r;
        T{j,i} = r;
    end for
end for
```

3.2. Contoh Pembentukan *Finite Field*

Misalkan akan dibentuk $GF(p^n)$ dengan $p = 3$ dan $n = 2$. Polinomial monic berderajat 2 pada $\mathbb{Z}_3[x]$ adalah $x^2, x^2+1, x^2+2, x^2+x, x^2+x+1, x^2+x+2, x^2+2x, x^2+2x+1, x^2+2x+2$. Sedangkan polinomial monic yang *irreducible* pada $\mathbb{Z}_3[x]$ adalah x^2+1, x^2+x+2, x^2+2x+2.

Misalkan α suatu akar dari $h(x)$, maka kumpulan semua kombinasi linear dari α dan anggota di $\mathbb{Z}_3$ membentuk anggota dari *GF(9)*, yaitu {0, 1, 2, α+1, α+2, 2α, 2α+1, 2α+2}.

Jika $h(x) = x^2 + x + 2$ di $\mathbb{Z}_3[x]$ dipilih, maka akar α akan memenuhi $\alpha^2 + \alpha + 2 = 0$ atau dapat ditulis sebagai $\alpha^2 = 2\alpha + 1$. Operasi penjumlahan yang digunakan adalah penjumlahan modulo 3, sedangkan operasi perkalian yang digunakan adalah perkalian modulo $h(x) = x^2 + x + 2$. Berikut ini diberikan tabel hasil operasi + dan *.

Tabel 3.6. Tabel Penjumlahan Modulo 3

+	0	1	2	*x*	*x*+1	*x*+2	2*x*	2*x*+1	2*x*+2
0	0	1	2	*x*	*x*+1	*x*+2	2*x*	2*x*+1	2*x*+2
1	1	2	0	*x*+1	*x*+2	*x*	2*x*+1	2*x*+2	2*x*
2	2	0	1	*x*+2	*x*	*x*+1	2*x*+2	2*x*	2*x*+1
X	*x*	*x*+1	*x*+2	2*x*	2*x*+1	2*x*+2	0	1	2
x+1	*x*+1	*x*+2	*x*	2*x*+1	2*x*+2	2*x*	1	2	0
x+2	*x*+2	*x*	*x*+1	2*x*+2	2*x*	*x*+1	2	0	1
2*x*	2*x*	2*x*+1	2*x*+2	0	1	2	*x*	*x*+1	*x*+2
2*x*+1	2*x*+1	2*x*+2	2*x*	1	2	0	*x*+1	*x*+2	*x*
2*x*+2	2*x*+2	2*x*	2*x*+1	2	0	1	*x*+2	*x*	*x*+1

Tabel 3.7. Tabel Perkalian Modulo $h(x) = x^2 + x + 2$

*	0	1	2	x	x+1	x+2	2x	2x+1	2x+2
0	0	0	0	0	0	0	0	0	0
1	0	1	2	x	x+1	x+2	2x	2x+1	2x+2
2	0	2	1	2x	2x+2	2x+1	x	x+2	x+1
X	0	x	2x	2x+1	1	x+1	x+2	2x+2	2
x+1	0	x+1	2x+2	1	x+2	2x	2	x	2x+1
x+2	0	x+2	2x+1	x+1	2x	2	2x+2	1	x
2x	0	2x	x	x+2	2	2x+2	2x+1	x+1	1
2x+1	0	2x+1	x+2	2x+2	x	1	x+1	2	2x
2x+2	0	2x+2	x+1	2	2x+1	x	1	2x	x+2

Selanjutnya kita akan coba melihat pembentukan *finite field* jika dipilih suatu $h(x)$ yang lain. Misalkan $h(x) = x^2 + 1$ di $\mathbb{Z}_3[x]$ dipilih, operasi penjumlahan yang digunakan adalah penjumlahan modulo 3, sedangkan operasi perkalian yang digunakan adalah perkalian modulo $h(x) = x^2 + 1$.

Berikut ini diberikan tabel hasil operasi + dan *.

Tabel 3.8. Tabel Penjumlahan Modulo 3

+	0	1	2	x	x+1	x+2	2x	2x+1	2x+2
0	0	1	2	x	x+1	x+2	2x	2x+1	2x+2
1	1	2	0	x+1	x+2	x	2x+1	2x+2	2x
2	2	0	1	x+2	x	x+1	2x+2	2x	2x+1
X	x	x+1	x+2	2x	2x+1	2x+2	0	1	2
x+1	x+1	x+2	x	2x+1	2x+2	2x	1	2	0
x+2	x+2	x	x+1	2x+2	2x	x+1	2	0	1
2x	2x	2x+1	2x+2	0	1	2	x	x+1	x+2
2x+1	2x+1	2x+2	2x	1	2	0	x+1	x+2	x
2x+2	2x+2	2x	2x+1	2	0	1	x+2	x	x+1

Tabel 3.9. Tabel Perkalian Modulo $h(x) = x^2 + 1$

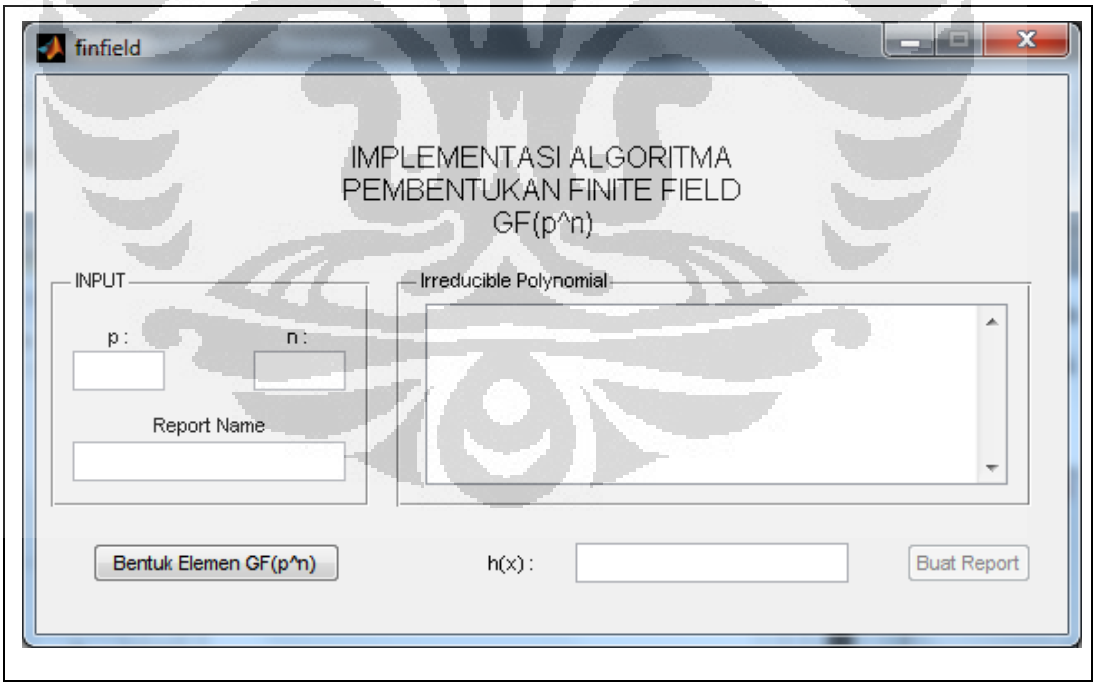
×	0	1	2	x	x+1	x+2	2x	2x+1	2x+2
0	0	0	0	0	0	0	0	0	0
1	0	1	2	x	x+1	x+2	2x	2x+1	2x+2
2	0	2	1	2x	2x+2	2x+1	x	x+2	x+1
x	0	x	2x	2	x+2	2x+2	1	x+1	2x+1
x+1	0	x+1	2x+2	x+2	2x	1	2x+1	2	x
x+2	0	x+2	2x+1	2x+2	1	x	x+1	2x	2
2x	0	2x	x	1	2x+1	x+1	2	2x+2	x+2
2x+1	0	2x+1	x+2	x+1	2	2x	2x+2	x	1
2x+2	0	2x+2	x+1	2x+1	x	2	x+2	1	2x

BAB 4
IMPLEMENTASI

Pada bab ini kita akan melihat implementasi dari algoritma pembentukan *finite field* dengan menggunakan *software* MATLAB. Berikut ini adalah spesifikasi mesin serta *software* yang digunakan saat menjalankan implementasi algoritma ini.

Prosesor	: Intel Core i5-750 @ 2.66 GHz
Memori	: 4 GB
Operating System	: Windows 7 (x64)
Versi MATLAB	: 7.8.0 (R2009a) 64 bit

Program yang digunakan pada bab ini menggunakan algoritma paralel pada saat pemilihan polinomial monik $h(x)$ yang *irreducible*. Tampilan program ini adalah sebagai berikut



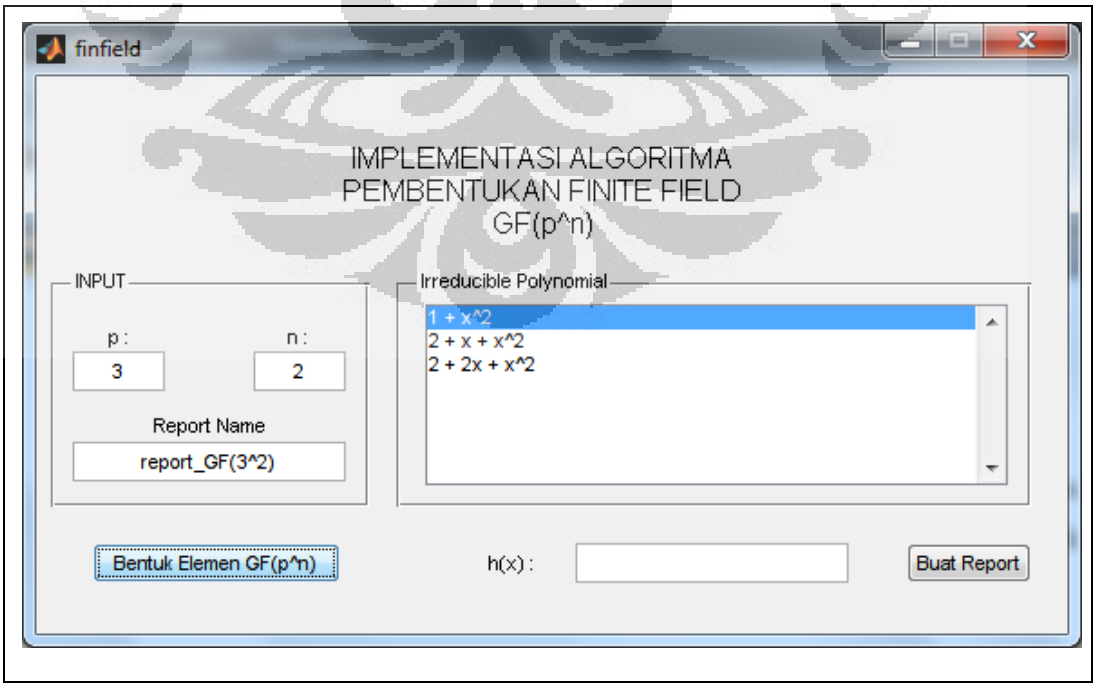
Gambar 4.1. Tampilan Awal Program

Input yang diberikan untuk program di atas adalah berupa nilai p, n yang akan digunakan untuk membentuk $GF(p^n)$ dan nama file yang akan digunakan untuk mencetak hasil pembentukan *finite field* ini.

Misalkan akan dibentuk $GF(3^2)$ seperti contoh pada subbab 3.2, maka inputnya adalah sebagai berikut

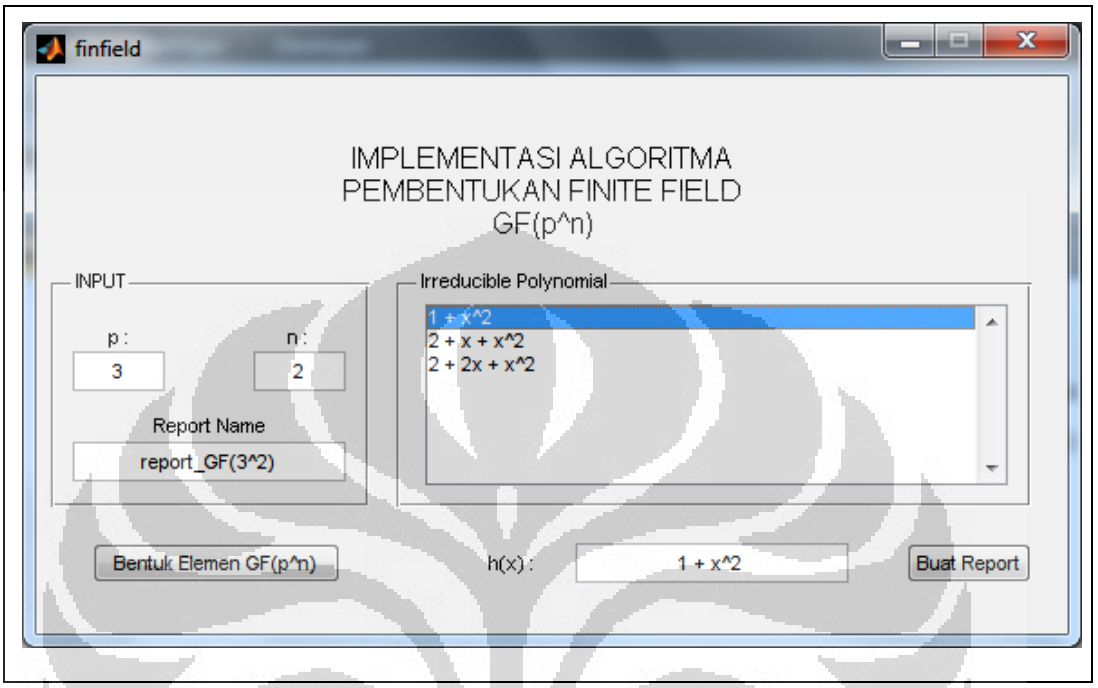
Gambar 4.2. Input untuk Pembentukan $GF(3^2)$

Setelah memasukkan input seperti gambar di atas, tekan tombol "Bentuk Elemen GF(p^n)". Berikut ini adalah tampilan program setelah elemen $GF(p^n)$ dibentuk



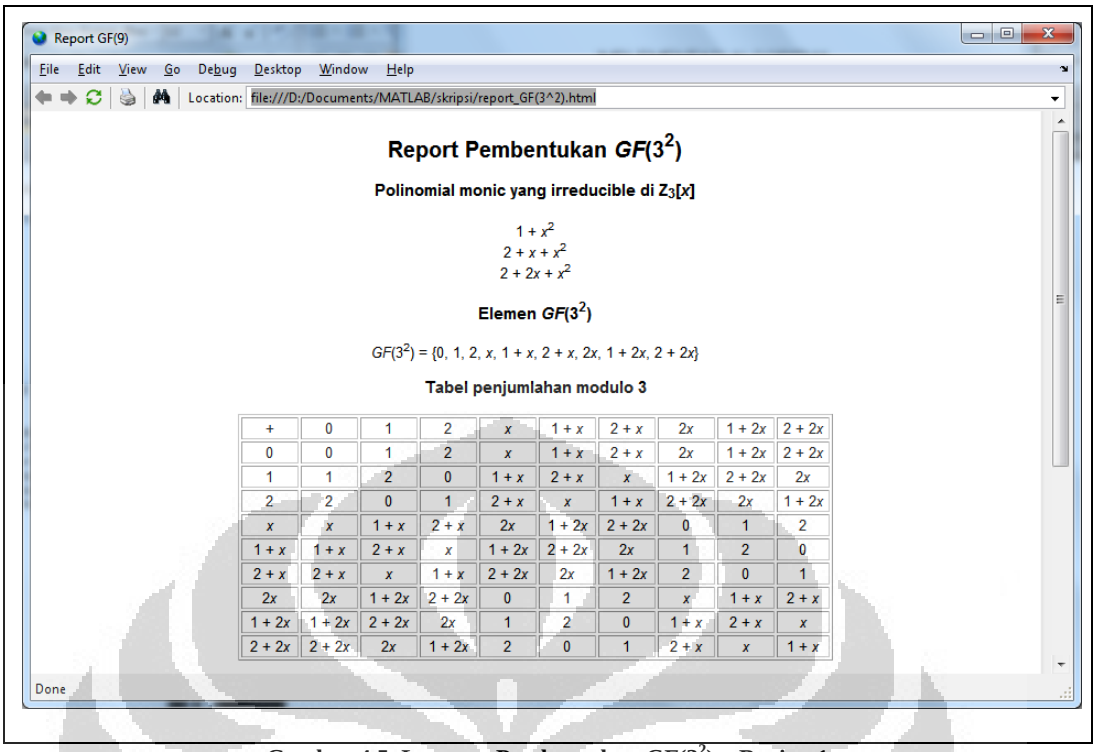
Gambar 4.3. Tampilan Program Setelah Elemen GF(p^n) Dibentuk

Hasil yang diperoleh pada tabel *irreducible polynomial* adalah semua polinomial *irreducible* di $\mathbb{Z}_p[x]$. Misalkan dipilih $h(x) = 1 + x^2$, maka input $h(x)$ akan berubah menjadi "1 + x^2".



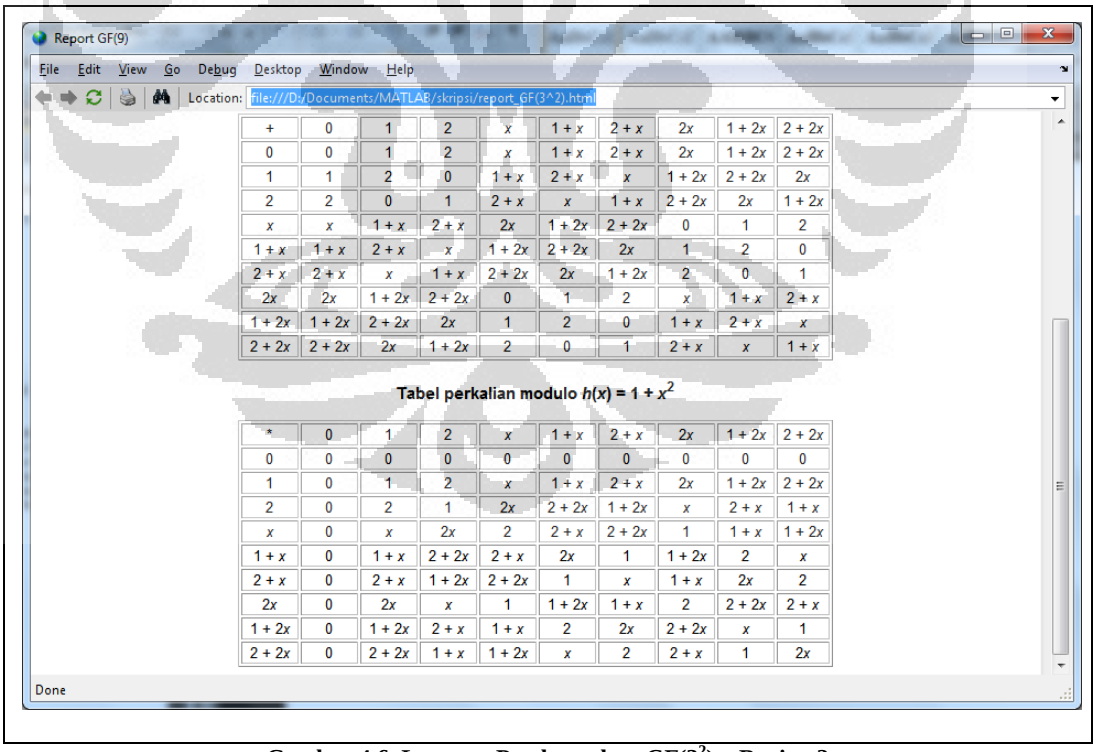
Gambar 4.4. Tampilan Program untuk Input $h(x)$

Setelah tombol "Buat Report" di tekan, maka program akan membuat file laporan dengan nama sesuai yang diberikan oleh *user*. File laporan untuk contoh di atas adalah sebagai berikut



+	0	1	2	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$
0	0	1	2	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$
1	1	2	0	$1+x$	$2+x$	x	$1+2x$	$2+2x$	$2x$
2	2	0	1	$2+x$	x	$1+x$	$2+2x$	$2x$	$1+2x$
x	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$	0	1	2
$1+x$	$1+x$	$2+x$	x	$1+2x$	$2+2x$	$2x$	1	2	0
$2+x$	$2+x$	x	$1+x$	$2+2x$	$2x$	$1+2x$	2	0	1
$2x$	$2x$	$1+2x$	$2+2x$	0	1	2	x	$1+x$	$2+x$
$1+2x$	$1+2x$	$2+2x$	$2x$	1	2	0	$1+x$	$2+x$	x
$2+2x$	$2+2x$	$2x$	$1+2x$	2	0	1	$2+x$	x	$1+x$

Gambar 4.5. Laporan Pembentukan $GF(3^2)$ ~ Bagian 1



+	0	1	2	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$
0	0	1	2	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$
1	1	2	0	$1+x$	$2+x$	x	$1+2x$	$2+2x$	$2x$
2	2	0	1	$2+x$	x	$1+x$	$2+2x$	$2x$	$1+2x$
x	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$	0	1	2
$1+x$	$1+x$	$2+x$	x	$1+2x$	$2+2x$	$2x$	1	2	0
$2+x$	$2+x$	x	$1+x$	$2+2x$	$2x$	$1+2x$	2	0	1
$2x$	$2x$	$1+2x$	$2+2x$	0	1	2	x	$1+x$	$2+x$
$1+2x$	$1+2x$	$2+2x$	$2x$	1	2	0	$1+x$	$2+x$	x
$2+2x$	$2+2x$	$2x$	$1+2x$	2	0	1	$2+x$	x	$1+x$

*	0	1	2	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$
0	0	0	0	0	0	0	0	0	0
1	0	1	2	x	$1+x$	$2+x$	$2x$	$1+2x$	$2+2x$
2	0	2	1	$2x$	$2+2x$	$1+2x$	x	$2+x$	$1+x$
x	0	x	$2x$	2	$2+x$	$2+2x$	1	$1+x$	$1+2x$
$1+x$	0	$1+x$	$2+2x$	$2+x$	$2x$	1	$1+2x$	2	x
$2+x$	0	$2+x$	$1+2x$	$2+2x$	1	x	$1+x$	$2x$	2
$2x$	0	$2x$	x	1	$1+2x$	$1+x$	2	$2+2x$	$2+x$
$1+2x$	0	$1+2x$	$2+x$	$1+x$	2	$2x$	$2+2x$	x	1
$2+2x$	0	$2+2x$	$1+x$	$1+2x$	x	2	$2+x$	1	$2x$

Gambar 4.6. Laporan Pembentukan $GF(3^2)$ ~ Bagian 2

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan pembahasan Bab 3, untuk membentuk $GF(p^n)$ *extension field* dari $\mathbb{Z}_p$ yang memiliki jumlah elemen $q = p^n$ diperlukan suatu polinomial $h(x)$ yang *irreducible* berderajat n di $\mathbb{Z}_p[x]$.

$GF(p^n)$ berhasil dibentuk dengan menggunakan algoritma pembentukan *finite field*. Implementasi algoritma pembentukan *finite field* ini akan membuat file laporan yang berisi berisi list elemen-elemen $GF(p^n)$, polinomial *irreducible* di $\mathbb{Z}_p[x]$ serta tabel-tabel hasil operasi di $GF(p^n)$. Contoh yang digunakan pada implementasi ini adalah pembentukan $GF(3^2)$.

Dalam proses pembentukan $GF(p^n)$, pemilihan polinomial *irreducible* berderajat n di $\mathbb{Z}_p[x]$ menyebabkan waktu eksekusi algoritma pembentukan *finite field* menjadi besar, karena pada kasus terburuknya algoritma ini akan menjalankan proses pencarian polinomial *irreducible* sebanyak jumlah elemennya, yaitu p^n.

Untuk mengatasi masalah ini, algoritma pembentukan *finite field* dapat dijalankan dengan menggunakan komputasi paralel. Dalam penulisan tugas akhir ini, penulis menerapkan komputasi paralel di algoritma pembentukan *finite field* dengan menggunakan prosesor yang memiliki 4 *core*.

Dari percobaan yang dijalankan, terlihat bahwa dengan menerapkan komputasi paralel pada pembentukan *finite field* waktu eksekusi yang dibutuhkan akan lebih kecil. Grafik perbandingan waktu eksekusi dapat dilihat pada Gambar 3.3.

5.2. Saran

Penelitian lebih jauh mengenai algoritma pembentukan *finite field* dapat dilakukan dengan cara memecah masalah p^n menjadi $(p^f)^m$ dengan $n = f * m$. Hal ini diharapkan mampu mengurangi *cost* yang ditimbulkan dari perkalian polinomial berderajat n.

DAFTAR PUSTAKA

Blaise, B. (2010). *Introduction to Parallel Computing*. Retrieved September 26, 2010, https://computing.llnl.gov/tutorials/parallel_comp/

Fahrurozi, A. (2009). *Pembentukan Extension Field.* Depok: Universitas Indonesia Fakultas Matematika dan Ilmu Pengetahuan Alam.

Herstein, I. (1996). *Abstract Algebra* (3rd Edition ed.). New Jersey: Prentice-Hall Inc.

Koblitz, N. (1999). *Algebraic Aspects of Cryptography.* Berlin: Springer.

Lidl, R., & Niederreiter, H. (1997). *Finite Fields. Encyclopedia of Mathematics and Its Applications* (Vol. 20). London, England: Cambridge University.

LAMPIRAN

finfield.m

```
function varargout = finfield(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @finfield_OpeningFcn, ...
                   'gui_OutputFcn',  @finfield_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function finfield_OpeningFcn(hObject, eventdata, handles,
varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = finfield_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)

function edit1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pushbutton1_Callback(hObject, eventdata, handles)

global p n gf ir;
p = str2double(get(handles.edit1, 'String'));
n = str2double(get(handles.edit2, 'String'));
[monic ir r gf] = par_poly_main(p, n);
for i=1:size(ir,1)
    listIr{i} = sprintPol(ir(i,:),'x');
end
set(handles.listbox1, 'String', listIr);
set(handles.pushbutton2, 'Enable', 'On');



function edit4_Callback(hObject, eventdata, handles)



function edit4_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pushbutton2_Callback(hObject, eventdata, handles)

global p n h gf ir tab_kali tab_tambah;

h = str2num(get(handles.edit4, 'String'));
tab_tambah = sum_table(gf, p, n);
[h tab_kali] = times_table(h, ir, gf, p, n);
nfile = get(handles.edit3, 'String');
nfile = [nfile '.html'];
createReport(nfile, p, n, h, gf, ir, tab_kali, tab_tambah);
open(nfile);
```

```
function listbox1_Callback(hObject, eventdata, handles)

global ir
val = get(handles.listbox1, 'Value');
set(handles.edit4, 'String', sprintPol(ir(val,:), 'x'));


function listbox1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

par_poly_main.m

```
function [monic, ir, r, gf] = par_poly_main(p, n)
q = p^n;
monic = ones(q,n+1);

for i=1:n
    A = repmat(0:p-1,p^(i-1),1);
    A1 = reshape(A,p^i,1);
    A2 = repmat(A1,p^(n-i),1);
    monic(:,i) = A2;
end

r = [];
ir = [];

parfor i=1:q
    poly = monic(i,:);
    if gfprimck2(poly,p) == -1
        r = [r;monic(i,:)];
    else
        ir = [ir;monic(i,:)];
    end
end

gf = monic(1:q,1:n);
```

gfprimck2.m

```
function ck = gfprimck2(a, p)
%GFPRIMCK Check whether a polynomial over a Galois field is
primitive.
%   CK = GFPRIMCK(A) checks whether the degree-M GF(2) polynomial
A is
%   a primitive polynomial for GF(2^M), where M = length(A)-1.
The
%   output CK is as follows:
```

```
%       CK = -1   A is not an irreducible polynomial;
%       CK =  0   A is irreducible but not a primitive polynomial;
%       CK =  1   A is a primitive polynomial.
%
%   CK = GFPRIMCK(A, P) checks whether the degree-M GF(P)
polynomial
%   A is a primitive polynomial for GF(P^M).  P is a prime number.
%
%   Note: This function performs computations in GF(P^M) where P
is prime. To
%   work in GF(2^M), you can also use the ISPRIMITIVE function.
%
%   The row vector A represents a polynomial by listing its
coefficients
%   in order of ascending exponents.
%   Example:  In GF(5), A = [4 3 0 2] represents 4 + 3x + 2x^3.
%
%   See also GFPRIMFD, GFPRIMDF, GFTUPLE, GFMINPOL, GFADD.

%   Copyright 1996-2007 The MathWorks, Inc.
%   $Revision: 1.14.4.5 $   $Date: 2007/08/03 21:17:40 $

% Error checking.
error(nargchk(1,2,nargin,'struct'));

% Error checking - P.
if nargin < 2
    p = 2;
elseif (  numel(p)~=1 || isempty(p) || ~isreal(p) || p<=1 ||
floor(p)~=p ||~isprime(p) )
    error('comm:gfprimck:InvalidP','The field parameter P must be
a positive prime integer.');
end

% Error checking - A.
if ( isempty(a) || ndims(a)>2 || any(any( abs(a)~=a | floor(a)~=a
| a>=p )) )
    if (p == 2)
        error('comm:gfprimck:InvalidAForP2','Polynomial
coefficients must be either 0 or 1 for P=2.');
    else
        error('comm:gflineq:InvalidAElements','Polynomial
coefficients must be real integers between 0 and P-1.');
    end
end
[m_a, n_a] = size(a);
if ( m_a>1 && n_a==1 )
    error('comm:gflineq:ANotRowVector','Polynomial input must be
represented as a row vector.');
end

% Allocate space for the result, assume primitive.
ck = ones(m_a,1);

% Each row is interpreted as a separate polynomial.  Cycle through
each row.
for k = 1:m_a,
```

```
    % First remove high-order zeros.
    at = gftrunc(a(k,:));
    m = length(at) - 1;

    % The polynomial is divisible by x, hence is reducible.
    % The only exception is when the polynomial is x ...
    if (at(1,1) == 0)
        if numel(at)==2
            ck(k) = 0;
        else
            ck(k) = -1;
        end


        % This polynomial is actually a constant.
    elseif ( m == 0 )
        ck(k) = 1;

        % The typical case.
    else

        % First test if the current polynomial is irreducible.
        n = p^(floor(m/2)+1)-1;
        % 'test_dec' is a vector containing the decimal(scalar)
representations of
        % the polynomials that could be divisors of 'at'.
        test_dec = p+1:n;
        % test_dec's that correspond to polynomials divisible by X
can be removed.
        test_dec = test_dec( mod(test_dec,p)~=0 );
        len_t = length(test_dec);
        test_poly = zeros(1,m);
        idx = 1;
        % Loop through all polynomials that could be divisors of
'at'.
        while ( idx <= len_t )
            % Expand the scalar value to a polynomial in GF(P).
            tmp = test_dec(idx);
            for idx2 = 1:m
                test_poly(idx2) = rem(tmp,p);
                tmp = floor(tmp/p);
            end
            [ignored, r] = gfdeconv(at,test_poly,p);
            if ( max(r) == 0 )
                ck(k) = -1;
                break;
            end
            idx = idx + 1;
        end

    end
end

%--end of gfprimck--
```

sprintPol.m

```
function hasil = sprintPol(p, s)

n = length(p);

hasil = [];

for i = 1 : n
    if p(i) ~= 0
        if i == 1
            hasil = [hasil num2str(p(i))]; %#ok<*AGROW>
        else
            if p(i) == 1
                hasil = [hasil s];
            else
                hasil = [hasil num2str(p(i)) s];
            end

            if i > 2
                hasil = [hasil '^' num2str(i-1)];
            end
        end

        if i < n
            if ~isempty(find(p(i+1:end) ~= 0, 1))
                hasil = [hasil ' + '];
            end
        end
    end
end

if sum(p == zeros(1,n)) == n
    hasil = '0';
end
```

sum_table.m

```
function T = sum_table(gf, p, n)

q = p^n;
for i=1:q
    for j=i:q
        t = gf(i,:) + gf(j,:);
        t = mod(t, p);
        T{i,j} = t;
        T{j,i} = t;
    end
end
```

times_table.m

```
function [h T] = times_table(h, ir, gf, p, n)

q = p^n;
benar = 0;

[row col] = size(ir);
for i=1:row
    if(size(ir,2) ~= length(h))
        break;
    end
    if(sum(h == ir(i,:)) == col)
        benar = 1;
        break;
    end
end
if(benar == 0)
    errordlg('h(x) bukan merupakan polynom irreducible di GF(q)');
end

if (benar == 1)
    h1 = fliplr(h);
    for i=1:q
        for j=i:q
            t = fliplr(poltimes(gf(i,:),gf(j,:)));
            t = mod(t, p);
            [s r] = deconv(t,h1);
            r = fliplr(mod(r,p));
            if length(r) < n
                r = [r zeros(1, n-length(r))];
            end
            if (length(r) > n) && (r(n+1) == 0)
                r = r(1:n);
            end
            T{i,j} = r;
            T{j,i} = r;
        end
    end
end
```

createReport.m

```
function createReport(nfile, p, n, h, gf, ir, tab_kali,
tab_tambah)

fid = fopen(nfile,'w');
[m m1] = size(gf);

% Membuat header report.html
fprintf(fid, '<html>\n');
fprintf(fid, '<head><title>Report GF(%d)</title></head>\n', p^n);
```

```
fprintf(fid, '<body>\n');

% Membuat isi report.html
fprintf(fid, '<div style="text-align:center; margin-left:auto;
margin-right:auto">');
fprintf(fid, '<h2>Report Pembentukan
<i>GF</i>(%d<sup>%d</sup>)</h2>', p, n);

% Mencetak semua polinomial monic yang irreducible di Zp[x]
fprintf(fid, '<h3>Polinomial monic yang irreducible di
Z<sub>%d</sub>[<i>x</i>]</h3>',p);
for i=1:size(ir,1)
    fprintf(fid, '%s', printPol(ir(i,:), 'x', 1));
    fprintf(fid, '<br />');
end

% Mencetak semua elemen GF(p^n)
fprintf(fid, '<h3>Elemen <i>GF</i>(%d<sup>%d</sup>)</h3>', p, n);
fprintf(fid, '<i>GF</i>(%d<sup>%d</sup>) = {', p, n);
for i=1:m
    fprintf(fid, '%s', printPol(gf(i,:), 'x', 0));
    if i~=m
        fprintf(fid, ', ');
    else
        fprintf(fid, '}');
    end
end

% Mencetak tabel penjumlahan di GF(p^n)
fprintf(fid, '<h3>Tabel penjumlahan modulo %d</h3>', p);
fprintf(fid, ['<table width="' num2str(m*m1*30) 'px" border="1px"
style="margin:auto; text-align:center;">']);
fprintf(fid, '<tr><td>+</td>');
for i=1:m
    fprintf(fid, '%s', ['<td>' printPol(gf(i,:), 'x', 1)
'</td>']);
end
fprintf(fid, '</tr>');

for i=1:m
    fprintf(fid, '%s', ['<tr><td>' printPol(gf(i,:), 'x', 1)
'</td>']);
    for j=1:m
        fprintf(fid, ['<td>' printPol(tab_tambah{i,j}, 'x', 1)
'</td>']);
    end
    fprintf(fid, '</tr>');
end
fprintf(fid, '</table>');

% Mencetak tabel perkalian di GF(p^n)
fprintf(fid, '<h3>Tabel perkalian modulo <i>h</i>(<i>x</i>) =
%s</h3>', printPol(h, 'x'));
fprintf(fid, ['<table width="' num2str(m*m1*30) 'px" border="1px"
style="margin:auto; text-align:center;">']);
fprintf(fid, '<tr><td>*</td>');
for i=1:m
```

```
    fprintf(fid, '%s', ['<td>' printPol(gf(i,:), 'x', 1)
'</td>']);
end
fprintf(fid, '</tr>');

for i=1:m
    fprintf(fid, '%s', ['<tr><td>' printPol(gf(i,:), 'x', 1)
'</td>']);
    for j=1:m
        fprintf(fid, '%s', ['<td>' printPol(tab_kali{i,j}, 'x', 1)
'</td>']);
    end
    fprintf(fid, '</tr>');
end
fprintf(fid, '</table>');

% Membuat footer report.html
fprintf(fid, '</div>');
fprintf(fid, '</body></html>');
fclose(fid);
```

printPol.m

```
function hasil = printPol(p, s, mode)

n = length(p);

hasil = [];

for i = 1 : n
    if p(i) ~= 0
        if i == 1
            hasil = [hasil num2str(p(i))]; %#ok<*AGROW>
        else
            if p(i) == 1
                hasil = [hasil '<i>' s '</i>'];
            else
                hasil = [hasil num2str(p(i)) '<i>' s '</i>'];
            end

            if i > 2
                hasil = [hasil '<sup>' num2str(i-1) '</sup>'];
            end
        end

        if i < n
            if ~isempty(find(p(i+1:end) ~= 0, 1))
                hasil = [hasil ' + '];
            end
        end
    end
end

if sum(p == zeros(1,n)) == n
    if mode == 1
```

```
        hasil =
'    0    ';
    else
        hasil = '0';
    end
end
```

benchmark.m

```
function benchmark(p, n, N)

clc;
T0 = zeros(N, 16);
T1 = T0;
for i = 1 : N
    display(['Percobaan ke-' num2str(i)]);
    [t0 t1] = test(p, n);
    T0(i,:) = t0;
    T1(i,:) = t1;
end

nfile = ['data ' num2str(p) '^n.mat']; %#ok<*NASGU>

save(nfile, 'T0', 'T1');
```

test.m

```
function [t0 t1] = test(p,n)

clear t0 t1;
if matlabpool('size') == 0
    matlabpool local 4;
end
t0 = zeros(1,n);
t1 = t0;
for i=1:n
    disp(['i = ' num2str(i)]);
    tic
    poly_main(p, i);
    t0(i) = toc;
    tic
    par_poly_main(p,i);
    t1(i) = toc;
end
matlabpool close
```