ABSTRACT

AHMADI. **The Construction of Arithmetic Algorithms GF(5^m) Generated by Cyclic Group Properties**. Supervised by SUGI GURITMAN and NUR ALIATININGTYAS.

To construct a cryptographic algorithm, many arithmetic concepts are needed. ElGamal encryption for example, can be defined over cyclic group Z_p^*, the usual arithmetic concepts. If the use of this arithmetic is associated with security aspect, then it requires large computational work. This thesis aims to construct arithmetic algorithm as an alternative arithmetic that can be applied to any cryptographic scheme, especially public key scheme. This algorithm is imposed from finite field $GF(5^m)$. Thus, the procedures to construct arithmetic algorithm are as follows. The first step is to choose primitive polynomial $M(x) \epsilon Z_5[x]$ of lower degree. The second step is to find primitive root $M(\alpha) = 0$, thus the equation $M(x) = 0$ has a root α in $GF(5^m)$. The resulted arithmetic algorithms are computational procedures for standard operation in $GF(5^m)$:addition, multiplication, division, invertion, and exponentiation. It can be concluded that constructed arithmetic algorithms $GF(5^m)$ are better than standard algorithms because some operations can be reduced using primitive polynomial or cyclic group properties, and using reduction of zero.

Keywords: arithmetic, cyclic group, $GF(5^m)$, primitive polynomial, cryptography.

I. Pendahuluan

Masalah keamanan merupakan salah satu aspek penting dari sebuah sistem informasi. Untuk menjamin keamanan sebuah informasi yang bersifat rahasia diperlukan suatu teknik pengamanan baik secara fisik maupun non fisik. Salah satu teknik pengamanan secara non fisik yaitu dengan mengenkripsi informasi rahasia menggunakan teknik kriptografi.

Kriptografi secara terminologi dasarnya terdiri dari dua tipe yaitu kriptografi *simetrik* dan kriptografi asimetrik atau sering disebut sebagai kriptografi kunci publik. Kunci *simetris* adalah jenis kriptografi yang paling umum digunakan. Kunci untuk membuat pesan yang disandikan sama dengan kunci untuk membuka pesan yang disandikan itu. Kunci *asimetris* merupakan pasangan kunci kriptografi yang salah satunya digunakan untuk proses enkripsi dan yang satu lagi untuk dekripsi. Contoh algoritme terkenal yang menggunakan kunci asimetrik adalah skema yang ditemukan oleh ElGamal pada tahun 1985. Skema ini didasarkan pada pemecahan problem logaritma diskret. Keamanan algoritme ini sangat tergantung pada pemilihan bilangan prima p. Semakin besar p maka algoritme ini akan semakin aman, akan tetapi semakin besar pula beban komputasi yang digunakan. Oleh karena itu pada masa sekarang, orang sudah mulai mencari alternatif lain untuk menggantikan aritmetik modular diantaranya aritmetik yang dibangkitkan struktur *finite field* $GF(p^m)$, kurva eliptik kriptografi, dan hipereliptik kriptografi.

Sebenarnya sudah banyak sekali penelitian tentang $GF(p^m)$ yang pernah dilakukan, diantaranya adalah paper berjudul *"Montgomery Multiplication in GF(2^k)"* menunjukkan operasi perkalian $c = a.b.r^{-1}$ dalam *field GF(2^k)* dimana r adalah unsur tetap dari *field* dapat diimplementasikan lebih cepat dalam perangkat lunak dibandingkan dengan operasi perkalian standar (Koc, 1998), paper berjudul *"Analysis and Construction of Galois Field for Efficient Storage Reliability*" juga menganalisis adanya implementasi berdasarkan tabel dan teknik optimasi operasi perkalian dan pembagian dalam $GF(2^l)$ (Greenan, 2007), dan thesis berjudul *"Konstruksi Algoritme Aritmetik $GF(2^n)$ Dengan Operasi Perkalian Dibangkitkan Dari Sifat Grup Siklik*" menyebutkan bahwa algoritme aritmetik hasil konstruksi cukup cepat dalam perhitungan komputasinya (Rosdiana, 2009).

Dalam thesis ini peneliti mencoba mengkonstruksi aritmetik yang berbeda dari sebelumnya yaitu aritmetik $GF(5^m)$ dan dengan pendekatan yang sama dari konstruksi yang telah dilakukan oleh Ibu Sri Rosdiana (2009) yaitu mendefinisikan sekaligus mengkonstruksi $GF(5^m)$ didasarkan pada sifat bahwa $GF(5^m)^*$ merupakan *grup siklik* yang dibangkitkan dari akar *primitif*. Penelitian ini bertujuan mengkonstruksi *finite field* $GF(5^m)$ dengan memperhatikan segi kecepatan dan kapasitas memori yang digunakan.

II. Landasan Teori

Teorema 1. Misal $p(x)$ yaitu *polinomial irredusibel* berderajat n atas $\mathbf{Z}_p$. $GF(p^m) =$ $\{a_0 + a_1x + \cdots + a_{n-1}x^{n-1} | a_0, a_1, \ldots, a_{n-1} \epsilon Z_p\}$ merupakan *field*.

Teorema 2. $GF(p^m)^*$ merupakan *grup siklik multiplikatif* ber *order* $p^m - 1$.

Teorema 3. Sembarang $GF(p^m)$ memuat suatu unsur *primitif* atau akar *primitif*.

Definisi 4. ***polinomial*** *minimum* $M(x)$ dari *c* atas $\mathbf{Z}_p$ merupakan *polinomial monik* berderajat terendah dengan koefisien dari $\mathbf{Z}_p$ sehingga $M(c) = 0$.

Teorema 5. Misalkan *m(x)* adalah *polinomial minimum* dengan unsur α dalam *finite field GF(p^m)*. Maka

(i) m(x) irredusibel

*(ii)*jika α akar dari *polinomial f(x)* dengan koefisien didalam *GF (p)*, maka *m(x)* membagi *f(x)*

(iii) *m(x)* membagi $x^{p^m} - x$

(iv) derajat $m(x) \leq m$

(v) Polinomial *minimum* berunsur *primitif* dari $GF(p^m)$ yang berderajat *m* merupakan *polinomial primitif.*

Teorema 6. Semua *field* hingga ber *order* p^m adalah isomorfik.

Teorema 7. Untuk sembarang p prima dan bilangan bulat$m \geq 1$, maka ada tunggal

field ber *order* p^m yang dinotasikan dengan $GF(p^m)$.

Lemma 8. Jika *n*, *r*, *s* bilangan bulat dengan $n \geq 1, r \geq 1, s \geq 1$, maka $n^s - 1 | n^r - 1$ jika dan hanya jika $s | r$.

Teorema 9.

i) *$GF\left(p^r\right)$ memuat sub field (isomorfik dengan) $GF\left(p^s\right)$ jika dan hanya jika* $s | r$.

ii) *Jika* $c \in GF\left(p^r\right)$, *maka* $c \in GF\left(p^s\right)$ *jika dan hanya jika* $c^{p^s} = c$. *Untuk sembarang field jika* $c^2 = c$, *maka c yaitu* 0 *atau* 1.

Teorema 10. $x^{p^m} - x$ = perkalian semua *polinomial monik, irredusibel* atas $\mathbf{Z}_p$ yang derajatnya membagi *m*.

Teorema 11. Jika *p* adalah prima dan m adalah intejer positif, maka berlaku :

1) Produk dari semua *polinomial irredusibel monik* dalam $Z_p[x]$ yang derajatnya membagi *m* atau faktor dari *m* sama dengan $x^{p^m} - x$.

2) Misalkan *f(x)* adalah *polinomial* berderajat *m* dalam $Z_p[x]$, maka f*(x) irredusibel* atas $Z_p[x]$ jika dan hanya jika $gcd\left(f(x), x^{p^i} - x\right) = 1$, untuk setiap $1 \leq i \leq \left[\frac{m}{2}\right]$

Teorema 12. Misalkan *p* adalah bilangan prima dan misalkan mempunyai faktor-faktor prima yang berbeda dari p^m*-1* adalah $r_1, r_2, \ldots, r_t$ maka *polinomial irredusibel* $f(x) \epsilon GF(p)[x]$ adalah *primitif* jika dan hanya jika untuk setiap $1 \leq i \leq t$ berlaku $x^{\frac{p^m-1}{r_i}} \neq 1(mod\ f(x))$

III. Hasil dan Pembahasan

Untuk mengkonstruksi aritmetik *GF (5^m)* dalam penelitian ini langkah pertamanya adalah dengan memilih *polinomial primitif* berderajat *m* atas Z_5, misal $M(x) \in Z_5[x]$ dimana $M(x) = a_0 + a_1x^1 + \cdots + a_nx^m$ sehingga diperoleh unsur *primitif* α dengan M(α) = 0. Kemudian, tentukan *basis* dari *GF(5^m)* sebagai ruang vektor atas Z_5, yaitu $\{1, \alpha^1, \alpha^2, \ldots, \alpha^{m-1}\}$. Dengan demikian, diperoleh himpunan $GF(5^m) =$ $\{a_0 + a_1\alpha^1 + \cdots a_{m-1}\alpha^{m-1} | a_0, a_1, \ldots, a_{m-1} \in GF(5)\}$. Kemudian semua unsur dari $GF(5^m)$ direpresentasikan ke dalam bentuk ruang vektor berdimensi *m* atas Z_5.

Bagaimana memilih polinomial primitif? Polinomial primitif adalah polinomial irredusibel yang akarnya adalah generator dari *GF(5)**. Oleh karena itu, diperlukan pemilihan polinomial irredusibel terlebih dahulu.

Algoritme 1. Pengetesan *polinomial irredusibel*

Deskripsi : Mengetes Vektor Apakah Irredusibel atau Redusibel
Input : Vektor A
Output : True atau False
1. a = banyaknya unsur A – 1, $m = (a/2)$ dibulatkan ke bawah
2. W = [0, 1]
3. Untuk I dari 1 sampai m, lakukan secara berulang :
   3.1. W = W pangkat 5 modulo A
   3.2. U= Jumlahkan W dengan [0, 4]
   3.3. H = FPB(U,A)
   3.4. h = banyaknya unsur H
   3.5. jika h > 1, maka return(false)
4. Return (True)

Setelah kita mendapatkan polinomial irredusibel, kemudian memeriksa apakah polinomial irredusibel yang didapatkan tersebut merupakan polinomial primitif atau bukan. Untuk memeriksa polinomial irredusibel adalah primitif, digunakan Algoritme 2.

Algoritme 2. Pengetesan *polinomial primitif*

Deskripsi : Mengetes Vektor Irredusibel Apakah Primitif atau Bukan
Input : Vektor A
Output : True atau False
1. m = banyaknya unsur A – 1, h = $5^m - 1$
2. F = faktorkan h
3. a = banyaknya unsur F
4. Untuk i dari 1 sampai a, lakukan secara berulang
   4.1. k = h/*i*
   4.2. H = [0, 1] pangkat k modulo A
   4.3. Jika H = [1], maka return(false)
5. Return(True)

Untuk mempercepat komputasi, dipilih *polinomial primitif* yang bersuku terkecil. Hal ini sangatlah beralasan, sebab operasi pada *finite field* $GF(5^m)$ dilakukan dalam modulo *f(x)*, dimana *f(x)* merupakan *polinomial primitif*. Dengan memilih *polinomial primitif* bersuku terkecil akan mengakibatkan proses komputasi yang dijalankan lebih cepat dibandingkan dengan menggunakan *polinomial primitif* biasa.

1. Penjumlahan *Polinomial*

Algoritme 3. Penjumlahan

Deskripsi : Menambahkan vektor A dan B dalam modulo 5
Input : Vektor $A = [a_0, a_1, \dots, a_t]$, vektor $B = [b_0, b_1, \dots, b_t]$, dan intejer positif m
Output : Vektor $C = [c_0, c_1, \dots, c_r]$
1. Tentukan vektor A, dan vektor B
2. Jika $A = [0]$, maka C = B
3. Jika B= [0], maka C = A
4. Jika s=t, maka
   4.1. $C = \{[(a_0 + b_0) mod\ 5, (a_1 + b_1) mod\ 5, \dots, (a_s + b_s) mod\ 5]\}$
   4.2. Lakukan reduksi nol dengan menggunakan algoritme 4
5. Jika s < t, maka
   5.1. $C = \{[(a_0 + b_0) mod\ 5, (a_1 + b_1) mod\ 5, \dots, b_t\ mod\ 5]\}$
   Lainnya
   $C = \{[(a_0 + b_0) mod\ 5, (a_1 + b_1)\ mod\ 5, \dots, a_s\ mod\ 5$
6. Return (C)

Keistimewaan Algoritme ini adalah pada setiap melakukan operasi selalu melibatkan Algoritme Reduksi Nol. Algoritme Reduksi Nol dapat mengurangi jumlah operasi pada konstruksi algoritme-algoritme aritmetik berikutnya sehingga secara

otomatis akan mampu mempercepat proses komputasi. Hal ini mengakibatkan banyaknya operasi pada Algoritme 4 ini dapat diminimalkan sehingga akan mempercepat proses komputasinya.

Algoritme 4. Algoritme Reduksi Nol

Deskripsi : Mereduksi nol pada posisi sebelah kanan
Input : Vektor T dan bilangan posotof *t*.
Output : Vektor *R*
1. T = R, t = banyaknya unsur T
2. Untuk j selama T[t] = 0 dan t >1, lakukan berulang
   2.1. Ganti unsur ke-*j* dari vektor T dengan himpunan kosong
   2.2. t = t - 1
3. Return (R)

Algoritme Reduksi Nol di atas jika digunakan pada sebuah algoritme dapat mengurangi jumlah operasi pada algoritme itu sendiri sehingga secara otomatis akan mampu mempercepat proses komputasi.

2. Perkalian *Polinomial*

Algoritme 5. Kelipatan vektor

Deskripsi : Mengalikan vektor A dengan skalar *n*
Input : Vektor $A = [a_0, a_1, \dots, a_t]$, dan intejer positif $n \in \{0, 1, \dots, 4\}$
Output : Vektor C
1. Jika $n = 0$, maka return([0])
2. Jika $n = 1$, maka return(A)
3. Selainnya $C = (n * A) mod\ 5$
4. Return (C)

Algoritme 6. Geser satu

Deskripsi : Menggeser vektor A satu langkah
Input : Vektor $A = [a_0, a_1, \dots, a_t]$, dan intejer positif m
Output : Vektor C
1. L = DatP[m]
2. t = banyaknya unsur vektor A
3. Jika t < m, maka tambahkan 0 di sebelah kiri pada vektor A, selainnya :
4. Tambahkan 0 di sebelah kiri pada vektor A yang telah direduksi pada unsur ke-m
5. Lakukan reduksi nol menggunakan Algoritme 3
6. Lipatkan vektor L dengan unsur ke-t pada vektor A menggunakan Algoritme 5
7. Jumlahkan hasil dari langkah ke-5 dengan hasil dari langkah ke-6 menggunakan Algoritme 4
8. Return(C)

Keistimewaan dari algoritme ini adalah hanya mengubah strukturnya saja dan menggeser bersifat konstan. Oleh karena itu apabila digunakan pada suatu algoritme akan mempercepat algoritme tersebut.

Algoritme 7. Perkalian *Polinomial*

Deskripsi : Mengalikan dua vektor, kemudian hasil perkaliannya dimoduluskan dengan vektor primitif P

Input : Vektor $A = [a_1, a_2, \dots, a_t]$, vektor $B = [b_0, b_1, \dots, b_t]$, dan intejer positif m

Output : Vektor $R = [r_1, r_2, \dots, r_c]$ dan c = nops(R).

1. Tentukan vektor A, vektor B, s = banyaknya unsur A, dan t = banyaknya unsur B
2. Jika $A = [0]$, atau B$= [0]$, maka return($[0]$)
3. Jika s < t, maka
   - 3.1. S=B, T=A, a=s, s=t, t=a
   - 3.2. R= Lipatkan vektor T dengan (op(1,S)) menggunakan algoritme 5.
   - 3.3. Untuk j dari 1 sampai (s – 1), lakukan
     - 3.3.1. T = Lakukan Geser satu dengan algoritme 6
     - 3.3.2. V = Lipatkan vektor T dengan (op(j+1,S)) menggunakan algoritme 5.
     - 3.3.3. R = Jumlahkan vektor R dan V dengan menggunakan algoritme 4 secara rekursif
4. Return (R)

Karena pada algoritme ini melibatkan Algoritme Kelipatan Vektor, Algoritme Geser Satu, dan Algoritme Penjumlahan yang melibatkan Algoritme Reduksi Nol maka dapat dikatakan bahwa algoritme ini cukup baik karena tidak membutuhkan ruang yang cukup besar dan proses komputasinya cukup cepat.

3. Invers *Polinomial*

Algoritme 8. Negasi vektor

Deskripsi : Mengubah vektor A dengan negasinya

Input : Vektor $A = [a_1, a_2, \dots, a_s]$,

Output : Vektor $C = [c_1, c_2, \dots, c_s]$

1. Petakan unsur x pada vektor A dengan $-x$
2. Return (C)

Algoritme 9. Pembagian *polinomial*

Deskripsi : Membagi dua polinomial tanpa modulo m

Input : Vektor T dan vektor S

Output : Vektor $C = [Q, R]$ dimana vektor Q sebagai hasil pembagian dan vektor R sebagai sisa pembagian

1. Jika S=[0], maka “false”
2. R=T; Q=[0]; r = banyaknya unsur T; s = banyaknya unsur S; g = unsur ke-s dari vektor S
3. Jika s = 1, maka
   - 3.1. Q = lipatkan vektor R dengan operan S
   - 3.2. R = [0]
   - 3.3. Return ([Q, R])
4. Untuk i selama $r \geq s$, lakukan
   - 4.1. k = r – s; t = unsur ke-r dari vektor R
   - 4.2. h = (t*g) mod 5
   - 4.3. jika k = 0, maka
     - 4.3.1. K = lipatkan vektor S dengan h
     - 4.3.2. Q = jumlahkan vektor Q dengan [h]
   - 4.4. Selainnya
     - 4.4.1. H = lipatkan vektor S dengan h
     - 4.4.2. K = [seq(0,j=1..k),op(H)]
     - 4.4.3. Q = jumlahkan (Q,[seq(0,j=1..k),h])
5. K = negasikan vektor K
6. R = jumlahkan vektor K dan vektor R
7. Return ([Q, R])

Algoritme 10. Invers *Polinomial*

Deskripsi : Invers polinomial dari suatu bilangan bulat m
Input : Vektor $T = [t_1, t_2, \dots, t_s]$, dan intejer positif m
Output : Vektor $H = [h_1, h_2, \dots, h_r]$ dan r = nops(C).

1. S = Negatifkan vektor (DatP(m)) menggunakan algoritme 5
2. Jika $T = [0]$, maka "salah"
3. Jika banyaknya unsur T = 1, maka return (T)
4. RA = [op(S), seq(0, j = 1…(m – t)), 1];
   RB = T; QA = [0]; QB = [1];
5. L = Bagi vektor RA dan RB menggunakan algoritme 9
6. RA = RB
7. RB = Operan kedua dari L
8. Untuk i selama RB tidak nol, maka lakukan langkah berikut berulang-ulang
   8.1. Tmp = QA; QA = QB
   8.2. H = Kalikan vektor QB dengan op(1,L) menggunakan algoritme 7
   8.3. R = Negasikan vektor H menggunakan algoritme 8
   8.4. QB = Jumlahkan vektor Tmp dengan vektor R menggunakan algoritme 4
   8.5. L = Lakukan pembagian vektor RA dengan vektor RB menggunakan algoritme 9
   8.6. RA = RB; RB = operan kedua dari L
9. H = Lipatkan vektor QB dengan op(RA) menggunakan algoritme 4.5
10. Return (H)

4. Pembagian *Polinomial*

Algoritme 11. Pembagian *Polinomial*

Deskripsi : Membagi dua polinomial dengan modulo m
Input : Vektor $A = [a_1, a_2, \dots, a_s]$, vektor $B = [b_1, b_2, \dots, b_s]$, dan intejer positif m
Output : Vektor $C = [c_1, c_2, \dots, c_s]$

1. iB = Inverskan vektor B menggunakan algoritme 10
2. Kalikan vektor A dengan vektor iB menggunakan algoritme 7
3. Return (C)

Algoritme Pembagian Polinomial ini hanya melibatkan Algoritme Invers Polinomial dan Perkalian Polinomial yang didalamnya melibatkan operasi reduksi nol dan kelipatan vektor yang tentunya membutuhkan ruang yang sedikit dan berakibat pada semakin cepatnya proses komputasi yang dijalankan.

5. Eksponen *Polinomial*

Algoritme 12. Eksponen *Polinomial*

Deskripsi : Mengeksponensialkan polinomial dalam modulo m
Input : Vektor $A = [a_1, a_2, \dots, a_s]$, intejer x dan intejer positif m
Output : Vektor $H = [h_1, h_2, \dots, h_r]$

1. $p = 5^m - 1$
2. k = moduluskan (x, p) menggunakan Algoritme 13
3. Jika $k \geq 0$, maka
   3.1. X = konversi nilai k dalam basis 2; t = banyaknya unsur X
   3.2. G = A; H = [1]
   3.3. Jika $X_1 = 1$, maka H = G
   3.4. Untuk i mulai dari 2 sampai t, lakukan
      3.4.1. G = kalikan vektor G dengan G menggunakan Algoritme 7
      3.4.2. Jika $X_i = 1$, maka H = kalikan vektor H dengan G menggunakan Algoritme 7
4. n = -k
5. X = konversi nilai n dalam basis 2; t = banyaknya unsur X
6. G = inverskan vektor A dengan Algoritme 10; H = [1]
7. Jika $x_1 = 1$ maka H = G
8. Untuk i mulai dari 2 sampai dengan t, lakukan
   8.1. G = kalikan vektor G dengan G menggunakan Algoritme 7
   8.2. Jika $X_i = 1$, maka H = kalikan vektor H dengan G menggunakan Algoritme 7
9. Return (H)

Algoritme 13. Modulus bilangan negatif

Deskripsi	: Membagi dua polinomial dengan modulo m
Input	: integer positif *m* dan *a*
Output	: integer *b*
	1. Hitung $b = a \bmod m$
	2. $c = (m/2)$, hasilnya dibulatkan ke bawah
	3. Jika b > c, maka
	3.1. $b = -(m - b)$
	4. Return(b)

IV. Kesimpulan

Dari hasil penelitian yang telah dilakukan diperoleh kesimpulan : 1. *Finite field* $GF(5^m)$ dikonstruksi dari polinomial primitif. Untuk mengkonstruksi algoritme aritmetik $GF(5^m)$ dipilih polinomial primitif yang bersuku terkecil, hal ini akan mengakibatkan proses komputasi yang dijalankan lebih cepat dibandingkan dengan menggunakan polinomial primitif biasa. 2. Algoritme Reduksi Nol digunakan untuk mengurangi jumlah operasi pada sebuah algoritme. Algoritme ini digunakan pada Algoritme Penjumlahan, Algoritme Penjumlahan digunakan dalam Algoritme Perkalian, Algoritme Penjumlahan dan Algoritme Perkalian digunakan dalam Algoritme Invers, dan seterusnya. 3.Algoritme Geser Satu mampu mempercepat proses kerja suatu algoritme, karena pada algoritme ini hanya mengubah strukturnya saja dan menggeser bersifat konstan. Algoritme ini digunakan pada Algoritme Perkalian, Algoritme Perkalian digunakan dalam Algoritme Invers, dan seterusnya. 4. Algoritme aritmetik hasil konstruksi tidak membutuhkan ruang yang besar sehingga cukup cepat dalam komputasinya karena yang dipakai sebagai modulo digunakan *polinomial primitif* bersuku terkecil, melakukan reduksi nol yang berfungsi mengurangi jumlah operasi dalam algoritme, dan operasi geser satu.

V. Daftar Pustaka

Cohen, Henri and Gerhard Frey.(2006). Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman & Hall/CRC.

Dummit;, D. S. and R. M. Foote. (1999). Abstract Algebra, john Wiley and Sons, Inc.

Gallian, J. A. (1998). Contemporary Abstract Algebra, Houghton Mifflin Company.

Guritman, S. (2004). Struktur Aljabar.

Huffman, W. Carry and Vera Pless (2003). Fundamentals of Error-Correcting Codes. Cambridge University Press.

Koc, K. Cetin, Acar, Tolga. Montgomery Multiplication in *GF(2^k)*. Designs, Codes and Cryptography 14(1), 57-69

Menezes, A., P. v. Oorschot, et al. (1996). Handbook of Applied Cryptography, CRC Press.

Pless, Vera (1990). Introduction to the Theory of Error-Correcting Codes, Second Edition. John Willey & Sons, Inc.

Rosdiana, Sri (2009). *Konstruksi Algoritme Aritmetik* $GF(5^m)$ *Dengan Operasi Perkalian Dibangkitkan Dari Sifat Grup Siklik*. Tesis